Exploring Issues in Lexical Acquisition using Bayesian Modeling

BENJAMIN BÖRSCHINGER, M.A.

thesis submitted for the degree of

Doctor of Philosophy Dr. phil.

at the

Department of Computing
Faculty of Science
Macquarie University

Institut für Computerlinguistik
Neuphilologische Fakultät
Universität Heidelberg

supervised by
Prof. Mark Johnson, PhD
Prof. Dr. Annette Frank

July 2014

dedicated to my parents

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis addresses questions about early lexical acquisition. Four case studies provide concrete examples of how Bayesian computational modeling can be used to study assumptions about inductive biases, properties of the input data and possible limitations on the learning algorithm.

The first study describes an incremental particle filter algorithm for a non-parametric word segmentation models and compares its behavior to Markov Chain Monte Carlo methods that operate in an offline fashion. Depending on the setting, particle filters may be outperformed by or outperform offline batch algorithms. It is argued that the results ought to be viewed as raising questions about the segmentation model rather than providing evidence for any specific algorithm.

The second study explores how modeling assumptions interact with the amount of input processed by a model. The experiments indicate that non-parametric word segmentation models exhibit an overlearning effect where more input results in worse segmentation performance. It is shown that adding the ability to learn entire sequences of words in addition to individual words addresses this problem on a large corpus if linguistically plausible assumptions about possible words are made.

The third study explores the role of stress cues in word segmentation through Bayesian modeling. In line with developmental evidence, the results indicate that stress cues aid segmentation and interact with phonotactic cues; and that substantive constraints such as a Unique Stress Constraint can be inferred from the linguistic input and need not be built into the model.

The fourth study shows how variable phonological processes such as segmental deletion can be modeled jointly with word segmentation by a two-level architecture that uses generative beta-binomial model map underlying to surface forms. Experimental evaluation for the phenomenon of word-final /t/-deletion shows the importance of context in determining whether or not a variable rule applies in context and that naturalistic data contains subtle complexities that may not be captured by summary statistics of the input, illustrating the need to not only pay close attention to the assumptions built into the model but also to those that went into preparing the input.

DECLARATION

As a cotutelle student at Macquarie University and Ruprecht-Karls-UniversityHeidelberg, I certify that the work in this thesis entitled "Exploring Issues in Lexical Acquisition using Bayesian Modeling" has not previously been submitted as part of requirements for a degree to any other university or institution other than Macquarie University and Ruprecht-Karls-University Heidelberg. Under a cotutelle agreement between Macquarie University and Ruprecht-Karls-University Heidelberg, this thesis has also been submitted to Ruprecht-Karls-University Heidelberg.

I also certify that the thesis is an original piece of research and it has been written by me. Any help and assistance that I have received in my research work and the preparation of the thesis itself have been appropriately acknowledged.

The research presented in this thesis did not require approval by the Macquarie University Ethics Review Committee.

In addition, I certify that all information sources and literature used are indicated in the thesis.

Sydney / Heidelberg, July 2014

Benjamin Börschinger

PUBLICATIONS

Chapters 3, 4, 5, and 6 have been published previously as peer-reviewed publications. Chapter 3 has been modified extensively, but the other chapters are close to the form in which they have been published. Only minor edits have been made to fit the layout of the thesis, and where appropriate, cross-references to the other chapters and discussion of the larger context have been added.

The associated publications are:

CHAPTER 3

- Börschinger, B. and Johnson, M. (2011). A particle filter algorithm for Bayesian word segmentation. In *Proceedings of the 2011 Australasian Language Technology Workshop*, pages 10–18, Canberra, Australia. Australasian Language Technology Association
- Börschinger, B. and Johnson, M. (2012). Using rejuvenation to improve particle filtering for Bayesian word segmentation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 85–89, Jeju Island, Korea. Association for Computational Linguistics

CHAPTER 4

- Börschinger, B., Demuth, K., and Johnson, M. (2012). Studying the effect of input size for Bayesian word segmentation on the Providence corpus. In *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*, pages 325–340, Mumbai, India. Coling 2012 Organizing Committee

CHAPTER 5

- Börschinger, B. and Johnson, M. (2014). Exploring the role of stress in Bayesian word segmentation using Adaptor Grammars. *Transactions of the Association of Computational Linguistics*, 2:93–104

CHAPTER 6

- Börschinger, B., Johnson, M., and Demuth, K. (2013). A joint model of word segmentation and phonological variation for English word-final /t/-deletion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1508–1516, Sofia, Bulgaria. Association for Computational Linguistics

1 INTRODUCTION

Except for pathological cases, human infants acquire their native language(s) with remarkable ease and speed, growing from pre-verbal infants to competent speakers of whatever language is spoken in their surrounding environment over the course of just a few years. There is a heated debate about the relative importance in language acquisition of prior biases or innate knowledge on the one hand and linguistic input or experience on the other. Thus, it is common to contrast 'rational' and 'empiricist' views of language acquisition. The former emphasize the role played by innate knowledge which, using a term of Chomsky (1965), is commonly called Universal Grammar; the latter emphasize the role played by linguistic input and experience more generally (for a recent example of the controversy, see Christiansen and Chater, 2009; Pinker and Jackendoff, 2009).

I share, however, Clark (2009)'s assessment that the implied dichotomy between 'empiricist' and 'nativist' approaches is a false one – the scientific question is "what [infants] are born with that is required for this task" (p. 2), and little is gained by giving labels to competing hypotheses. In this thesis, I use computational modeling to address some of these questions as they apply to word segmentation, the problem of "breaking up of the essentially continuous stream of speech into morphemes and words" (Brown, 1973, p. 265).

This introductory chapter introduces the idea of computational modeling and how it can contribute to our understanding of language acquisition. After a brief general motivation for modeling, I outline the *Bayesian* approach taken in this thesis and illustrate how it can contribute to our understanding of word segmentation by giving an overview of the individual chapters of the thesis.

1.1 COMPUTATIONAL MODELING OF LANGUAGE ACQUISITION

In the context of language acquisition, a computational model is a particular instance of a more general theory (Harley, 2008, p. 4). Theories of language acquisition provide high-level explanations of particular phenomena, they tend to rely on the implicit understanding of the reader. In particular, 'verbal-conceptual' theories such as Tomasello (2003)'s usage-based theory or the parameter setting theory of language acquisition (Yang, 2006) are presented in mostly informal language that suffices to convey the core idea of an explanation but leave a great many details unspecified. Consequently, determining the actual implications of such

a theory has to rely on both intuitions and informal reasoning, making it hard to arrive at an objective evaluation of a proposal.

A computational model, in contrast, has to be formally specified and allows for mathematical analysis or the actual implementation as a computer program.[1] This makes it possible to derive the implications of different proposals in a principled way and, in many cases, to perform large scale evaluations of proposals on actual data.

1.1.1 *Computational models as testing proposals*

Thus, the main difference between computational models of and linguistic or psycholinguistic theories about language acquisition is explicitness (Alishahi, 2011; Sun, 2008b). As an example, consider the idea of Parameter Setting as introduced in Chomsky (1981) which, arguably, is the most popular 'rationalist' theory of language acquisition.[2]

Briefly, it suggests that there is a set of universally shared 'parameters' which account for syntactic differences between languages. And that, consequently, language acquisition amounts to learning which parameter setting corresponds to the language spoken in one's environment – "experience is necessary to fix the parameters of core grammar" (p. 8); the parameters themselves, however, need not be acquired. This is a compelling proposal that, to this date, fuels research into language acquisition (for a recent book-length treatment, see Guasti, 2004). Yet, as Chomsky himself points out, left at this level of detail "[t]here are [...] many unspecified details. [...] How many parameters are there? How much exposure to a language and what kind of evidence do children need to set each parameter[?]".

Often, filling in these details in a fully satisfactory way may not be possible and one may refuse to be more specific on the grounds that not all parameters may have been discovered, as well as admitting that the precise way of setting parameters may have yet to be determined. Clearly, intuitive models can be very useful in stimulating research and, in a sense, are required to ever arrive at more fleshed out proposals. Yet, one needs to be cautious that "without detailed theories, most of the details of an intuitive (or verbal-conceptual) theory are left out of consideration, and the intuitive theory may thus be somehow vacuous or internally inconsistent, or otherwise invalid. These problems of an intuitive theory may not be discovered until a detailed model is developed" (Sun, 2008b, p. 6).

A perfect example of this kind of problem is Gibson and Wexler (1994)'s discussion of the trigger learning algorithm which constituted the first fully specified computational model of Chomsky's parameter

1 I do not draw a distinction between mathematical and computational models (as e.g. Sun, 2008b). Arguably, Bayesian models would qualify as mathematical models under this terminology.

2 An accessible introduction is Yang (2006). Interestingly, Chomsky himself seems to no longer promote this idea in his current Minimalist Program (Chomsky, 1995).

setting idea. A trigger is a particular kind of sentence (or, more generally, syntactic construction) which allows a learner to uniquely determine the value a particular parameter has to take to fit the language in its environment. The idea that triggers provide a solution to the parameter setting problem is intuitive and was, indeed, widely accepted; Gibson and Wexler (1994) show, however, that a straight-forward way of actually implementing this idea does not work. Not only is their algorithm unable to identify the correct parameter setting in all circumstances, it can even end up with a wrong grammar.

Obviously, their conclusion that triggering does not work depends on the specific assumptions which they made about how triggers are used in acquisition. A proponent of triggering can always dispute that their computational model faithfully represents the intended interpretation of triggering, and indeed, it is still considered as a strategy for language acquisition (for a recent proposal, see Sakas and Fodor, 2012).

Hence, "[e]ven the most successful computational model can hardly prove that humans exploit a certain strategy or technique when learning a language" (Alishahi, 2011, p. 6). At most, they can "show what type of linguistic knowledge is learnable from what input data" and "give us insights about which representations and processes are more plausible in light of the experimental findings on child language acquisition". This 'limitation', however, is not special to computational modeling. Any scientific result can be rejected on the grounds of rejecting one of the assumptions on which it depends (Quine, 1951; Popper, 1959).

In this respect, then, computational modeling is not fundamentally different from psycholinguistic or, indeed, any kind of empirical experimentation. Conclusions drawn from psycholinguistic experiments also depend on the acceptance of a host of assumptions that connect what is actually measured (reading times, well-formedness judgments, ...) to what is (supposed to be) the underlying psychological process (lexical retrieval, syntactic parsing, ...). Considering this, the fact that a computational model has to make explicit its assumptions ought to be counted as a benefit rather than a limitation.

1.1.2 *Bayesian computational models*

I have argued that computational models provide a means of testing specific proposals about language acquisition. I will now spell out in more detail the particular approach taken in this thesis, namely *Bayesian computational modeling*.

A Bayesian computational model embodies a particular set of assumptions about the inductive biases of a learner (the *prior*), the nature of the input available to the learner (the *data*) and the relationship between any of the possible hypotheses a learner can consider and the input (the *likelihood*). Given such a model, it allows us to answer the

question what the assumptions built into the model imply given any particular set of observations by performing *Bayesian inference*.

This idea is formally explained in Chapter 2. Here, I discuss the conceptual importance of Bayesian inference in the study of language acquisition. I critically discuss the popular idea that Bayesian inference is the 'rational' or 'optimal' way of learning from experience and that Bayesian modeling allows for 'rational analysis' of language acquisition (e.g. Pearl and Goldwater, in press; Griffiths et al., 2008); and introduce my alternative view that emphasizes the idea of *logical probability* and views Bayesian modeling as a principled means of drawing out the logical implications of a specific proposal (this also seems to be the view of Dunbar, 2013).

1.1.2.1 *Bayesian Inference and Rationality*

At the heart of the Bayesian approach is the idea that "Probability is degree of belief" (Hájek, 2012, p. 25). Thus, for a Bayesian the expression "The probability that this coin will land heads is 99%" means that the speaker is 99% certain that this coin will land heads on the next toss. In contrast, for a *frequentist* this statement would mean that the fraction of heads in a long sequence of coin tosses will be 99%. This difference between a subjective Bayesian and a Frequentist or objective view of probability is a topic of heated debate with which we will not concern ourselves here – the interested reader is referred to Gillies (2000) for an excellent philosophical discussion of these different views on the nature of probability.[3]

Treating probability as degree of belief also provides a way of formalizing talk about belief: to represent that somebody is 99% certain that some proposition H is true, write $P(H) = 0.99$. At first blush, this seems to be little more than a convenient shorthand. However, the idea that degrees of belief are represented as probabilities puts substantive constraints on what qualifies as a set of coherent or rational beliefs.

DUTCH BOOK ARGUMENTS AND RATIONALITY Thus, a family of arguments going back to Ramsey (1931) *proves* that, unless one's total set of beliefs satisfies the axioms of probability, one is susceptible to a 'Dutch book' – a (possibly rather complex) bet in which one is *guaranteed to lose money*.

The arguments are involved and I do not want to discuss them in detail – Hacking (2009, chapter 14) provides a very readable exposition. To provide a rough idea, though, consider why being 60% certain that

3 It is worth noting, however, that Chomsky (1968)'s influential criticism against the use of probabilities in the study of language – "the notion 'probability of a sentence' is an is an entirely useless one, under any known interpretation of this term" – presupposes a frequentist interpretation. A Bayesian interpretation, in contrast, fits nicely with the idea that linguistics is about the beliefs speakers have rather than some 'objective' linguistic reality.

it will rain tomorrow and 60% certain that it will not rain tomorrow is problematic. Holding both beliefs is *inconsistent* or *irrational* in that it violates the requirement that the total probability of any event must not exceed 1 as writing $A \vee \neg A$ for "it will rain tomorrow or it will not rain tomorrow", $P(A \vee \neg A) = 0.6 + 0.6 = 1.2$.

The argument starts from the assumption that being 60% certain about something is equivalent to being willing to bet at rate 0.6 on the truth of this. Being 60% certain that it will rain tomorrow is equivalent to being willing to accepting a bet in which one wins 40 dollars if it rains tomorrow and loses 60 dollars if it doesn't.[4] Analogously, being 60% certain that it will not rain tomorrow is equivalent to being willing to accept a bet in which one wins 40 dollars if it does not rain tomorrow and loses 60 dollars if it does.

Then, a problem arises as follows: being 60% certain that it will rain tomorrow and 60% certain that it will not rain tomorrow means that one will accept bets such that – irrespective of the weather on the next day – one is guaranteed to lose 20 dollars: if it does rain, one wins 40 dollars in the first bet but loses 60 in the second; if it does not, one wins 40 dollars in the second bet but loses 60 in the first . A combination of bets which incurs a sure loss is called a "Dutch book", giving the name to the general form of argument.

As alluded to above, what is wrong with being 60% certain about something and its negation is that these certainties sum to 120%, rather than 100%, violating one of the axioms of probability. Through discussing many more cases much more formally, Ramsey (1931) and de Finetti (1990) showed that unless one's beliefs – the totality of probabilities (reflecting one's degree of certainty) one assigns to statements – satisfies *the axioms of probability*, one is subject to a Dutch book and, in this sense, guaranteed to lose money, a result known as the *Ramsey-De Finetti Theorem*.

As accepting bets that ensure one loses money no matter what is hardly rational, only beliefs that satisfy the axioms of probability can be considered 'rational' or, to use the more technical term, coherent (Gillies, 2000, p. 59ff). Note my use of scare quotes around 'rational' – I will not further discuss in what sense our ordinary usage of 'rational' coincides with the technically precise but obviously rather limited idea of 'not susceptible to a Dutch book' but it is important to keep in mind that, in this context, 'rational' has a precisely specified meaning.

1.1.2.2 *Bayesian updating*

The Ramsey-De Finetti theorem also puts a constraint on how beliefs ought to be updated. In particular, the only way to guarantee that a coherent set of beliefs remains coherent after having been 'updated' on

4 This connection between betting and degree of belief is at the heart of Bayesian decision theory. There are good grounds for contesting its adequacy which I will not discuss here – see (Gillies, 2000, p. 55ff) for discussion.

the basis of some observation is to use *Bayesian updating*. Informally, one has to update ones *prior belief* about H which one held before making observation E into the *posterior belief* about H given E using *Bayes' Theorem*:[5]

$$P(H \mid E) \propto P(H)P(E \mid H)$$

The expression $P(E \mid H)$ is called *the likelihood of* H and it quantifies how expected the observation one actually made would be if H were true. In models of language acquisition, it reflects how a learner relates the input E to the possible hypotheses it can consider. $P(H)$ is called the prior and reflects which hypotheses are favored by the learner 'a priori'. Bayes' theorem is often stated informally as

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

Dutch book arguments prove that Bayesian updating is the only 'rational' way of learning from experience.[6]

1.1.2.3 *Bayesian Inference and evidential relations*

Another closely related interesting view interprets the posterior probability $P(H \mid E)$ as "the degree of support or confirmation that a piece of evidence E confers upon a given hypothesis H" (Hájek, 2012, p. 18). Particularly popular in the philosophy of science (see Earman, 1992), following Dunbar (2013) I think that this 'Logic Bayesian' idea can explain very well what kind of contributions Bayesian models make to the study of language acquisition.

Under the 'Logic Bayesian' approach, Bayesian modeling allows us to draw out the 'logical conclusions' of specific assumptions about learners and the input by performing posterior inference, i.e. determining $P(H \mid E)$ which, to repeat, quantifies how strongly the observation supports any particular hypothesis. Again, Bayes' Theorem plays a central role as it is the means by which one can calculate $P(H \mid E)$ for specific cases.

A formal justification for treating posterior probabilities in this way has been given by Cox (1946) who argues that Bayesian updating is the natural extension of deductive (logical) reasoning to situations in which uncertainty is involved. At a very high level, A implies B can be formalized as $P(B \mid A) = 1.0$ – assuming the truth of A, one is 100% certain of the truth of B; cases where A bears on the truth of B

5 I omit the denominator, see chapter 2.1.2 for a proper mathematical discussion of Bayes' Theorem.

6 Strictly speaking, what is required for 'rational' learning is not use of Bayes' Theorem but that one's updated belief be derived by *conditionalization*. The importance of Bayes' Theorem stems from the fact that for many cases, it is what makes possible to perform this conditionalization, although there are alternative ways to calculate the required conditional probability $P(H \mid E)$ on the basis of available probabilities $P(E \mid H)$ and $P(E)$. I gloss over this detail here, see Talbott (2013) and Joyce (2008) for extensive discussion.

without fully determining it can be handled by assigning probabilities of less than 1.0, and the probability calculus takes on the role played by rules of inference in logic. I refer the reader to the discussion of Cox's argument by Jaynes (2003, chapter 2) and Dunbar (2013, chapter 1).

The difference between motivating Bayesian modeling through Dutch book arguments that show the 'optimality' of Bayesian inference and the Logic Bayesian emphasis on the degree of support that evidence confers on particular hypotheses is subtle and, as is usually the case for conceptual issues like these, does not affect the practical steps involved in Bayesian modeling. Indeed, the two views are not mutually exclusive but can be viewed as supplementing each other. For example, in addition to Cox's argument Logic Bayesians might point towards the Ramsey-De Finetti theorem to provide further support to their choice of using Bayesian inference to draw out the logical conclusions of a set of hypotheses.

It is important, however, to be explicit about how experimental findings of Bayesian modeling ought to be interpreted. I view Bayesian modeling as a tool to address the question what – as a matter of principle – can be learned from particular observations given particular assumptions. In logical notions, this is the question of to what extent particular observations support different hypotheses, making it natural to emphasize the Logic Bayesian view. In the context of language acquisition, this is precisely the question that 'Poverty of Stimulus arguments' (Chomsky, 1980) are intended to address. I will elaborate this point in the following section.

BAYESIAN MODELS AND LEARNABILITY 'Poverty of the Stimulus arguments', popularized by Noam Chomsky (Chomsky, 1980), are arguments that, in their general form, derive the necessity of postulating some piece of (linguistic) knowledge as innate on the basis of the observation that a learner would be unable to acquire it from the evidence it has access to. Arguably the most famous example concerns the sensitivity of linguistic rules to hierarchical rather than linear notions: how can infants determine whether the rules of their language should make reference to hierarchical notions such as 'sibling in a tree' rather than linear notions such as 'second word in a sequence'?

An explicit formulation of this argument is given by Perfors et al. (2011), an article that exemplifies to a high degree the approach I argue for. Without going into the specific details, the authors rephrase the *informal* argument outlined by Chomsky as a specific probabilistic model.

This allows them to use Bayesian inference to directly quantify how strongly a small corpus of transcribed child directed speech supports different hypotheses. Surprisingly, they find that – pace Chomsky's informal argument to the contrary – the 'stimulus is rich enough'. The data supports the hypothesis that there are hierarchical structures much

more strongly than that there are linear structures. Crucially, Perfors et al. (2011)'s result is arrived at *not by intuitive reasoning* but by *drawing out the conclusions of a specific proposal in a principled way*, namely Bayesian Inference.

Of course, by spelling out the argument in much more detail their work is easily criticized as simply not having made the right assumptions. Indeed, Dunbar (2013) discusses at length the many ways in which these assumptions are lacking. In a sense, this is the general 'problem' of computational models to which I alluded above – the requirement of being explicit may force one to make questionable assumptions which makes it easy to contest the importance of any findings based on the model. Yet it is important to realize that in the absence of explicit assumptions, there is no real argument to begin with. In the slightly modified words of Partha Niyogi, "for [computational] models the assumptions are more questionable but the conclusions are more reliable – for [intuitive] models, the assumptions are more believable but the conclusions more suspect" (Niyogi, 2006, p. 39).

If Perfors et al. (2011) way of filling in the details is not the one intended by Chomsky and colleagues, the proper reply is not to dismiss Bayesian modeling. Rather, the argument should be made sufficiently precise to be given a proper evaluation.

An alternative way of criticizing this work is to argue that, even if *Bayesian inference* may be able to draw the right conclusions from the input, humans are incapable of performing this kind of reasoning. While this may be the case, in the absence of concrete evidence that humans are, in principle, incapable of this kind of reasoning, it is not a particularly convincing move. In fact, recent work such as Phillips and Pearl (in press) and the work presented in chapter 3 of this thesis demonstrate that *incremental* Bayesian inference is possible, indicating that work in Bayesian modeling can also speak on questions of learnability by comparing the performance of different algorithms. A detailed and critical discussion of possible interpretations along those lines is provided in chapter 3. However, in so far as Poverty of Stimulus arguments are supposed to speak on the *logical* question of language acquisition, i.e. what is learnable *as a matter of principle* rather than actual fact, the question whether or not humans are capable of performing Bayesian inference is irrelevant.

To summarize, I take Bayesian modeling to offer *a principled framework to address learnability issues*, answering questions of the form "what kinds of conclusions can be drawn from what kinds of input".

This highlights an important point of difference between Bayesian computational models and other computational models. Bayesian models are not intended as claims about the actual mechanisms employed by human infants. In the terminology going back to Marr (1982), Bayesian models are at the computational rather than the algorithmic level of description. No assumption about humans actually being Bayesian rea-

soner's are required for these kinds of analyses, as they concern *the logical question* of what provides evidence for what, given specific assumptions, *rather than the psychological question* of how infants actually acquire their language. Another way of putting this is that I view our computational models as *tools to study* how specific assumptions learners can make interact with specific kinds of inputs available to them; not as scientific hypotheses about what human learners actually do.

Yet, it is wrong to conclude from this that Bayesian modeling is completely disconnected from and irrelevant to concrete questions about human learning. Both successes and failures of specific models can be taken as evidence for and against the assumptions built into these models. If something cannot be learned from the input given particular assumptions but we have evidence to believe that it is learned, something about the assumptions or the specification of the input is wrong *as a matter of logic*. On the other hand, if something can be learned from the input given particular assumption, this demonstrates that, *as a matter logic*, the assumptions suffice to acquire this kind of knowledge. This is precisely the logic of Poverty of Stimulus arguments, and one could even argue that – assuming Bayesian inference is the proper extension of logical inference to reasoning under uncertainty (as suggested by Cox's argument) – Bayesian modeling is the framework which has to be used to properly evaluate these arguments. Personally, I advocate the weaker position that Bayesian modeling is one framework which can be used to properly evaluate these kinds of arguments.

1.2 COMPUTATIONAL MODELS OF WORD SEGMENTATION

I now turn to a discussion of the particular problem discussed in this thesis, word segmentation. First, I define the problem and then illustrate how computational modeling can contribute to understanding how infants solve it. To this end, I briefly review prior work and then give a brief summary of the research undertaken for this thesis, providing a high-level overview of its contributions.

1.2.1 *The Word Segmentation problem*

Word segmentation is the problem of 'breaking up [...] the essentially continuous stream of speech into morphemes and words" (Brown, 1973, p. 265). This is among one of the earliest problems human language learners have to address as most aspects of their language such as the morphology, syntax, or semantics presuppose word- or morpheme-like units.

Language acquisition research has made several findings that are highly suggestive of what kind of strategies infants may use to solve the segmentation problem. For example, Saffran et al. (1996) demon-

strated that at the very young age of 8 months infants are sensitive to the transition probabilities between syllables and use these statistics to segment words in artificial language experiments, suggesting that distributional learning plays an important role.

Relatedly, Jusczyk et al. (1993) showed that stressed syllables are treated by English learning infants as cues for the beginnings of words and Mattys et al. (1999) provided evidence that infants are sensitive to the correlation of particular consonant sequences with word beginnings and ends, suggesting that particular kinds of cues are exploited by infants to perform segmentation.

As argued above, the contribution of computational modeling consists in providing a testing ground for proposals made by psycho-linguists by first making a proposal explicit enough to be implemented and then evaluating these models, thus providing evidence in favor or against particular proposals. This idea is quite general to computational modeling, and I review some of the results of prior work before discussing the particular questions addressed in this thesis.

1.2.2 *The computational Word Segmentation problem*

Whereas human infants have to segment actual speech, following previous work (Brent and Cartwright, 1996; Brent, 1999; Goldwater, 2007; Goldwater et al., 2009) I make the simplifying assumption that what is being segmented is a *discrete sequence* of phonemes. For example, I will represent an utterance such as "you want to see the book" as a phoneme sequence

$$\mathrm{y}_{\triangle}\mathrm{u}_{\triangle}\mathrm{w}_{\triangle}\mathrm{a}_{\triangle}\mathrm{n}_{\triangle}\mathrm{t}_{\triangle}\mathrm{t}_{\triangle}\mathrm{u}_{\triangle}\mathrm{s}_{\triangle}\mathrm{i}_{\triangle}\eth_{\triangle}\mathrm{ə}_{\triangle}\mathrm{b}_{\triangle}\upsilon_{\triangle}\mathrm{k}$$

where each triangle indicates a possible boundary position. The goal of word segmentation is to decide which of the possible word boundaries are real word boundaries. The 'correct' solution is

$$\mathrm{y}_{\triangle}\mathrm{u}_{\blacktriangle}\mathrm{w}_{\triangle}\mathrm{a}_{\triangle}\mathrm{n}_{\triangle}\mathrm{t}_{\blacktriangle}\mathrm{t}_{\triangle}\mathrm{u}_{\blacktriangle}\mathrm{s}_{\triangle}\mathrm{i}_{\blacktriangle}\eth_{\triangle}\mathrm{ə}_{\blacktriangle}\mathrm{b}_{\triangle}\upsilon_{\triangle}\mathrm{k}$$

where each black triangle indicates a word boundary. This way of formulating the segmentation problem abstracts away from particular issues such as the acquisition of the phonemic inventory which, to some degree, happens jointly with early segmentation (Clark, 2009, pp. 59–62). Consequently, recent work such as Phillips and Pearl (in press) argues for representing the input to word segmentation in terms of syllables rather than phonemes. This point will be discussed in some detail in chapter 4 on page 141. A criticism that can be raised against both choices, however, is the idealization that the infant's perception of the input is infallible and the pronunciation of a word (whether represented as a sequence of phonemes or atomic syllables) does not depend on the context.

While some work has tried to address some of these simplifying assumptions, the results suggest that at the current stage, making these idealizations is the only way forward. For example, Jansen et al. (2013) found that current unsupervised speech technologies yield very noisy 'categorical percepts' of spoken speech in which *no word is ever transcribed twice in the same way*. Thus, there literally are no distributional cues about words in these transcripts, and applying segmentation models to this output yields essentially chance performance.

On the other hand, recent work tries to address the issue of pronunciation variation, see Elsner et al. (2012), Elsner et al. (2013) and Chapter 6 of this thesis. While this kind of work is important in providing a way how, in theory, variation can be handled in segmentation models, there is a host of unresolved questions as to how well infants do treat different kinds of variation, raising the question how the results of these models ought to be interpreted. We will return to this particular point at the end of Chapter 6.

In summary, then, I believe that at the moment the idealized formulation of the segmentation problem as adopted in virtually all work (Brent and Cartwright, 1996; Brent, 1999; Venkataraman, 2001; Yang, 2004; Goldwater, 2007) provides the most productive setting in which computational models of word segmentation can be studied.

1.3 PRIOR WORK AND MODELING RESULTS

I summarized some core findings about word segmentation as performed by human infants above. Computational modeling has made a host of contributions that relate to different of these proposals. For an excellent review of prior work, I point the reader to Daland (2009, chapter 1). Here, I briefly summarize some results which I take to illustrate nicely the kinds of contributions that can be expected from computational modeling.

1.3.1 *Local statistics*

Saffran et al. (1996) provided experimental support for the idea that young infants are sensitive to *local statistics* such as the transition probability between syllables and seem to segment units from speech according to these statistics. Briefly, the idea is that for syllables which form a word, the conditional probability of the second syllable given the first tends to be much higher than for syllables which do not form part of a word.

This suggests that simple statistics defined over adjacent segments in the input may form the basis for word segmentation. In addition to (or instead of) the probability of the following syllable given the previous one, one could also imagine statistics such as tho pointwise

mutual information between adjacent syllables to work well (Swingley, 2005).

Computational models make it possible to evaluate a wide range of different proposals of this kind on actual child directed speech. To give one example, Yang (2004) implemented a learner that relies on transitional probability as proposed by Saffran et al. (1996) and evaluated it on a huge corpus of child directed speech, comprising 226,178 syllabified words. This evaluation showed that, when applied to actual child directed speech rather than the artificial languages used in Saffran et al. (1996)'s experiments, the segmentation results of the simple transitional probability learner are surprisingly low – only 42% of the words it posited were actual words, and it only identified 23% of the actual words of its input. Of course, this finding does not invalidate the experimental result of Saffran et al. (1996) that infants are sensitive to transitional probabilities. But it suggests, as Yang (2004) argues, that for successful word segmentation of real language more than mere sensitivity to transitional probability is needed.

A different class of local statistics models was motivated by the finding of, e.g., Mattys et al. (1999) that infants are sensitive to *phonotactic* properties of their language. One way to implement a model that exploits phonotactic regularities is by identifying sequences of phonemes – diphones – which occur never or with very low probability inside of any word. To illustrate, Daland and Pierrehumbert (2011) contrast the diphone /p d/ with the diphone /b a/; the latter occurs very frequently inside words whereas the former does not. Word segmentation amounts to identifying diphones which almost never occur inside of words and posit boundaries between them – this is, at a high-level, the idea of their Diphone-Based Segmentation model. On the basis of this concrete implementation, Daland and Pierrehumbert (2011) demonstrate that a phonotactic segmentation strategy is effective for English. Moreover, their model shows that the relevant phonotactic knowledge of which diphones tend to occur at word junctures and which don't can be acquired jointly with segmenting the input – in particular, knowledge of utterance boundaries is sufficient to determine some phonotactic expectations which allow first boundaries to be identified, resulting in an incremental refinement of the phonotactic knowledge.

Interestingly, Daland and Zuraw (2013) found that the same model does not perform well on Korean data, raising the important question what kinds of phonotactics a language needs to exhibit to be useful for segmentation; or, turning the question on its head, which other ways of implementing a model that exploits phonotactic regularities there are, and whether these strategies would perform on both English and Korean (and, of course, any other language).

With this brief review of local statistics learners, I move towards *lexical* models which are also considered in this thesis.

1.3.2 *Lexical models*

Unlike the models just discussed, lexical models do not simply try to identify word boundaries – they also attempt to explicitly build a *lexicon*, that is, a list of words of the language.

In these kinds of models, utterances are segmented by trying to match words already in the lexicon against the unsegmented input, thus providing a segmentation in terms of known and novel words – the novel words simply are those parts of the utterance which could not be matched. To illustrate, imagine you already know that "dog" is a word. Then you can 'segment' the utterance "thedogbarks" by spotting "dog" which yields "the dog barks". In the process, you have learned two novel words, "the" and "barks", which will be added to the lexicon and can be used to segment novel utterances.

This seems to raise a chicken-and-egg problem as, initially, one cannot assume knowledge of any words. Yet, the 'Recession segmentation' algorithm proposed by Yang (2004) and further refined by Lignos (2012) demonstrates that a simple strategy can solve this problem in practice. When incrementally processing the input their algorithm initially treats every utterance as a word. At some point, it will observe a short utterance which will occur again as part of a longer utterance; for example, a single word utterance such as "doggie!" or a short phrase such as "stop it!". At this point, the larger utterance will be broken into smaller units which, themselves, can be used to analyze more novel utterances and identify more short units.

By implementing this algorithm and applying it to huge corpora of child directed speech, Yang (2004) and Lignos (2012) were able to demonstrate that this simple strategy performs segmentation surprisingly well.

An alternative line of work treats lexicon identification and segmentation as a joint inference problem in a probabilistic setting. This approach originated in Brent (1999) and, in the Bayesian formulation of Goldwater (2007), is the approach taken in this thesis. The specifics of these kinds of models will be discussed in Chapter 2, here I merely point out some of the interesting findings made in this approach.

A core finding of Goldwater (2007) which will be discussed in more depth again in Chapter 4 is that unless dependencies that exist between the actual words in an utterance are taken into account, a segmentation model will tend to undersegment the input. To illustrate, in an utterance such as "the doggie barks" the words are not independent – intuitively, the probability of "doggie" following "the" is much higher than that of "the" following "doggie", as is evident from noting that "doggie the barks" is not an English sentence.

What Goldwater (2007) demonstrated is that *if* a model embodies the assumption that words are independent, *then* it will prefer segmentations in which sequences of words which exhibit strong dependencies

will be analyzed as single words. In the example, the model is likely to segment "thedoggie barks" rather than "the doggie barks". Moreover, Goldwater (2007) showed that a model that explicitly tries to not only learn a lexicon and infer segmentations but also tries to learn the probabilities with which each word follows each other word (Bigram probabilities) is less prone to undersegmentation. This illustrates how modeling results can establish a relationship between an error pattern attested in infant segmentations such as *undersegmentation* (Brown, 1973; Peters, 1983) and *independence assumptions about words*, a particular kind of assumptions a learner can make. This does not commit us to the idea that infants actually perform Bayesian inference – rather, we use the Bayesian model to determine what 'follows' in a logical sense from particular assumptions given data.

Another interesting result is that of Fourtassi and Dupoux (2014) which directly addresses the question why the performance of a segmentation model might differ between languages. They find that Japanese exhibits a high degree of 'segmentation ambiguity': many of the 'gold' words in the Japanese input can occur as parts of other words. Consequently, even considering only segmentations in which actual Japanese words occur there are thousands of possible segmentations for utterances of medium length, a striking contrast to English. Consistent with this observation, they show that a successful Japanese segmentation model requires the ability to handle subtle word-to-word dependencies to distinguish 'correct' from 'incorrect' segmentations. In addition, they suggest that additional cues such as prosodic breaks may play a more important role in languages such as Japanese than for languages like English, again raising a concrete question that can empirically be tested through psycholinguistic experimentation.

What is common to all findings reported is that they raise questions for and bear on findings of psycholinguistic experiments. Yang (2004) demonstrated that, by themselves, transitional probabilities do not perform well on real language, highlighting the importance of additional cues; Daland and Pierrehumbert (2011) and Daland and Zuraw (2013) raise the question which strategies of exploiting phonotactic regularities may be viable cross-linguistically; Goldwater et al. (2009) illustrates the importance of taking into account word-to-word dependencies in segmentation, raising the question what kind of assumptions (if any) infants are making; and Fourtassi and Dupoux (2014) suggest that segmentation ambiguity of a language is a good indicator for how well a lexical segmentation strategy performs on it, raising the question what additional cues are used by infants to solve the ambiguity problem.

With this discussion, I conclude my introductory brief review of prior work and conclude the chapter by providing an outline of the remaining chapters, giving a high-level overview of the contributions of the thesis.

1.4 OUTLINE OF THE THESIS AND CONTRIBUTIONS

The thesis comprises by and large the papers on word segmentation which I published as a first author during my candidature.[7] The papers have been edited to form a single narrative that addresses a set of related questions about word segmentation and Bayesian computational models of word segmentation; thus, cross-references to different chapters have been added and at several points, additional discussion and explanation has been included. To enable independent reading of the content chapters, I have kept high-level introductory exposition contained in each paper; rather than providing more detailed explanation of this material in each individual chapter, I include a detailed discussion of the mathematical and formal background in Chapter 2.

Chapter 2 is not based on a previously published paper, and it provides an extensive and in depth review of the Unigram model and Bigram model of Goldwater et al. (2009), as well as an introduction to the Adaptor Grammar framework used in this thesis that emphasizes their connection to models defined without using any kind of formal grammar. While largely reviewing and presenting prior work, this chapter makes several novel contributions.

Thus, the presentation of the models emphasizes the difference between the *collapsed* representation used for inference and the model which is defined in terms of (draws from) Dirichlet Process, providing a clearer understanding of non-parametric word segmentation models. Also, I provide a more detailed discussion of the popular Gibbs sampling algorithm of Goldwater et al. (2009) than currently exists in the literature and extend the models by adding a *possible word constraint* and discuss hyper parameter inference. These extensions allow me to demonstrate that, pace Goldwater et al. (2009), the choice of the 'lexical model' which encodes prior expectations about words has a huge impact on segmentation performance if hyper parameters are inferred rather than manually set.

Chapter 3 proposes two incremental Particle Filter inference algorithms for the Unigram and the Bigram model of Goldwater et al. (2009). I demonstrate that and explain why incremental inference for word segmentation models is challenging and that, in general, performance of incremental and batch algorithms differs. Thus, the possible word constraint discussed in Chapter 2 proves to be more important in incremental than batch inference; and particle filters that are allowed to 'revise' earlier analyses do not only perform better than particle filters that do not; in some circumstances they identify more accurate segmentations than the batch algorithms.

7 I do not include discussion of my work on semantic parsing (Börschinger et al., 2011) and joint work on word segmentation to which I contributed as second or third author (Fourtassi et al., 2013; Jansen et al., 2013; Synnave et al., 2014).

A critical discussion of my findings cautions against a currently popular "less is more" (Newport, 1990) reading of these kinds of results (e.g. Pearl et al., 2010; Phillips and Pearl, in press). I propose a different interpretation in line with the Bayesian approach outlined in the introduction and show how they raise questions about *models* rather than provide answers about *mechanisms*, leading to the questions addressed in chapter 4.

Chapter 4 examines the relationship between input size and performance of a variety of models on a large amount of longitudinal data. It performs a large scale evaluation of a variety of models that make different assumptions about the relations between words in an utterance and the internal structure of words. The experiments identify a previously unnoticed 'overlearning' property of Bayesian word segmentation models: counter-intuitively, having access to more input results in a degradation of segmentation quality due to undersegmentation. I discuss which aspects of the input and the model are responsible for this phenomenon which, to some extent, also explains chapter 3's finding that certain incremental algorithms perform better segmentation than batch algorithms. I show that Johnson (2008b)'s idea of collocations virtually solves the overlearning problem for a large corpus of roughly 25,000 utterances and that, in line with the findings of chapter 3, stronger constraints on possible words are important for models that capture word-to-word dependencies to perform well.

Chapter 5 explores how stress cues which have been argued to play an important role in infant segmentation of English can be added to Bayesian word segmentation models. In line with developmental evidence, the results indicate that stress cues aid segmentation. Going beyond previous modeling work in this direction, it demonstrates that phonotactic and stress cues as well as overall amount of input interact. The results also show that a substantive constraint on possible words previously argued to explain the usefulness of stress cues Yang (2004) can be acquired jointly with performing segmentation rather than having to be built in; and that the models I discuss correctly identify the dominant stress pattern from the data.

Chapter 6 suggests a way of adding phonological rules to Bayesian word segmentation models. In particular, it studies the phenomenon of /t/-deletion and presents a two-level model that infers underlying and surface forms of segmented words. Experimental evaluation shows the importance of context in determining whether or not a variable rule applies in context, in line with linguistic work on the phenomenon; and that naturalistic data contains subtle complexities that may not be captured by summary statistics of the input. This illustrates the need to not only pay close attention to the assumptions built into the model but also to those that went into preparing the input on which models are evaluated.

The final chapter concludes and suggests several directions along which the research presented in the thesis can be extended.

2

BACKGROUND

This background chapter provides a detailed review of the mathematics for the kind of models explored in this thesis. To this end, I briefly review core ideas from probability theory and define the notation used in this thesis. I will then introduce generative probabilistic models, leading up to a detailed recapitulation of the Unigram model and Bigram model for word segmentation (Goldwater, 2007; Goldwater et al., 2009).

The presentation of these models also discuses the relationship between the collapsed representation under which inference is performed and the generative model in terms of the Dirichlet Process. In addition to explaining the standard Gibbs sampling algorithm originally introduced by Goldwater (2007) in detail, I show how hyper parameter inference can be performed jointly with segmentation. This leads to a novel finding about the role of the base distribution which encodes prior assumptions about possible words: contrary to previous experimental results by Goldwater et al. (2009), the base distribution does make a marked difference on segmentation performance when the hyper parameters are inferred rather than manually set.

Finally, I present the Adaptor Grammar framework (Johnson, 2007) which is used in Chapters 4 and 5. Adaptor grammars allow for the easy specification of a huge class of Bayesian non-parametric probabilistic models through (probabilistic) context-free grammars.

2.1 BASIC REVIEW OF PROBABILITY THEORY

A random variable X is a variable that takes on values on some set $\mathscr{X}$, called the range of the random variable.[1] I write $P(X = x)$ for the probability with which X takes on value $x \in \mathscr{X}$ and $P(X)$ for the distribution function of X. If the context makes clear the random variable, I simply write $P(x)$ instead of $P(X = x)$. If the range of a random variable X is finite or countably infinite, it is called a *discrete random variable*. In this case, the distribution function $P(X)$ has to satisfy

$$0 \leqslant P(X = x) \leqslant 1$$

$$\sum_{x \in \mathscr{X}} P(X = x) = 1$$

1 This brief review closely follows that of Murphy (2012, chapter 2.2).

If the range of a random variable Θ is uncountably infinite, we call it a continuous random variable and require of $P(\Theta)$ that

$$0 \leqslant P(\Theta = \theta)$$
$$\int P(\Theta = \theta) d\theta = 1$$

where the integral is over the range of Θ.

I use capital letters $X, Y, \ldots$ for discrete random variables, capital greek letters $\Theta, \Phi, \ldots$ for continuous random variables, and small letters $x, y, \theta, \phi, \ldots$ for the concrete values a random variable can take.

I write $X \sim F$ to indicate that a random variable X is distributed according to some F. For example, from

$$X \sim \mathsf{Bern}(0.3)$$

it follows that

$$P(X = x_1) = 0.3$$
$$P(X = x_2) = 0.7$$

where x_1, x_2 are the two different values X can take. The '$\sim$' abbreviates "is distributed according to", thus the entire expression $X \sim F$ can be read as "X is distributed according to F". An overview of several standard distributions which will be used throughout this thesis are given in Table 2.1 which also provides some more examples for the "distributed-according-to" notation.

2.1.1 *Marginal, joint and conditional distributions*

A distribution over a single random variable is called its *marginal distribution*. A distribution over multiple random variables is called their *joint distribution*. For discrete random variables X, Y, their joint distribution has to satisfy

$$1 \leqslant P(X = x, Y = y) \leqslant 1$$
$$\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} P(X = x, Y = y) = 1$$

and for continuous random variables Θ, Φ

$$0 \leqslant P(\Theta = \theta, \Phi = \phi)$$
$$\int\int P(\Theta = \theta, \Phi = \phi) d\theta d\phi = 1$$

This extends in the obvious way to distributions over more than two random variables. Joint distributions over both discrete and continuous

random variables can also be defined. To illustrate, a distribution over discrete X, Y and continuous Θ has to satisfy

$$0 \leqslant P(X = x, Y = y, \Theta = \theta)$$
$$\int \sum_{x \in \mathscr{X}, y \in \mathscr{Y}} P(X = x, Y = y, \Theta = \theta) d\theta = 1$$

Given any joint distribution $P(X, Y)$, the marginal distribution of any of the random variables can be derived through *marginalization*:[2]

$$P(Y = y) = \begin{cases} \sum_{x \in \mathscr{X}} P(X = x, Y = y) & \text{if } X \text{ is discrete} \\ \int P(X = x, Y = y) dx & \text{if } X \text{ is continuous} \end{cases}$$

The *conditional distribution* of X given Y is defined as

$$P(X = x \mid Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}$$

One can decompose any joint distribution $P(X_1, \ldots, X_n)$ into a product of a marginal and several conditional distributions using the *chain rule*:

$$P(X_1, \ldots, X_n) = P(X_1) \prod_{i=2}^{n} P(X_i \mid \mathbf{X}_{1:i-1})$$

where I write $\mathbf{X}_{j:k}$ to abbreviate a sequence of random variables $X_j, X_{j+1}, \ldots, X_k$ and use boldface to indicate sequences.

2.1.2 *Bayes' Theorem*

Using the ideas introduced so far, the following equality can be derived:[3]

$$\begin{aligned} P(H = h \mid D = d) &= \frac{P(H = h, D = d)}{P(D = d)} \\ &= \frac{P(H = h) P(D = d \mid H = h)}{\sum_{h' \in \mathscr{H}} P(H = h') P(D = d \mid H = h')} \end{aligned}$$

This is known as *Bayes' Theorem* and is central to this thesis because it relates *D*ata and *H*ypotheses. Its importance stems from the fact that it allows us to express $P(H \mid D)$, called the *posterior probability distribution* over H given D, in terms of $P(H)$, the marginal or *prior distribution* of H, and $P(D \mid H)$, the conditional probability of the data given the hypothesis also called the *likelihood* of H.[4]

2 For joint distributions over more than two random variables, marginalization also allows one to derive the joint distribution of any subset of the variables.

3 For simplicity, I only consider the case of discrete random variables.

4 It is common to refer to $P(D \mid H)$ as the likelihood of the data. However, we follow MacKay's recommendation to "[n]ever say 'the likelihood of the data'. Always say 'the likelihood of the [hypothesis]'. The likelihood function is not a probability distribution" (MacKay, 2003, p. 29).

Distribution	Support	Notation	Parameters	Distribution Function	mean and mode
Beta	'probability' $\phi \in [0,1]$	$\Phi \sim \text{Beta}(a,b)$	'prior sample sizes' $a>0, b>0$	$P(\phi) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\phi^{a-1}(1-\phi)^{b-1}$	mean=$\frac{a}{a+b}$ mode=$\frac{a-1}{a+b-2}$
Dirichlet	'probability vector' $\theta \in \Delta^k$	$\Theta \sim \text{Dir}(\alpha_1,\ldots,\alpha_k)$	'prior sample sizes' $\alpha_j > 0$	$P(\theta_1,\ldots,\theta_k) = \frac{\Gamma(\sum_{i=1}^k(\alpha_i))}{\prod_{i=1}^k\Gamma(\alpha_i)}\prod_{i=1}^k\theta_i^{\alpha_i-1}$	mean(θ_i)=$\frac{\alpha_i}{\sum_{j=1}^k\alpha_j}$ mode(θ_i)=$\frac{\alpha_i-1}{\sum_{j=1}^k\alpha_j-1}$
Bernoulli	two outcomes $\mathscr{X} = \{x_1, x_2\}$	$X \sim \text{Bern}(\phi)$	'success probability' $\phi > 0$	$P(x_1) = \phi$ $P(x_2) = (1-\phi)$	
Categorical (Discrete)	k outcomes $\mathscr{X} = \{x_1,\ldots,x_k\}$	$X \sim \text{Cat}(\theta_1,\ldots,\theta_k)$	'probabilities' $\theta_j \in [0,1],\ \sum_{i=1}^k\theta_i = 1$	$P(X = x_k) = \theta_k$	
Multinomial	count vector of outcomes	$\langle N_1,\ldots,N_k\rangle \sim \text{Mult}(n;\theta_1,\ldots,\theta_k)$	'sample size' n 'probabilities' $\theta_j \in [0,1], \sum_{i=1}^k\theta_i = 1$	$P(n_1,\ldots,n_k) = \binom{n}{n_1\ \ldots\ n_k}\prod_{i=1}^k\theta_i^{n_i}$	mean(n_j)=$n\theta_j$
Geometric	positive integers	$N \sim \text{Geom}(\theta)$	'stopping probability' $\theta \in [0,1]$	$P(n) = (1-\theta)^{n-1}\theta$	mean(n)=$\frac{1}{p}$
Gamma	positive reals	$\Theta \sim \text{Gamma}(\alpha,\beta)$	shape $\alpha > 0$ rate $\beta > 0$	$P(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{\alpha+1}e^{-\beta\theta}$	mean=$\frac{\alpha}{\beta}$ variance=$\frac{\alpha}{\beta^2}$

Table 2.1: Several common distributions, adapted from Table A.1 in Gelman et al. (2014).

The denominator $P(D = d)$ is the marginal probability of d, also called the *evidence*. We can calculate it using marginalization although, in most cases, performing the required sums or, for continuous random variables, integrals is infeasible. Luckily, $P(D = d)$ is independent of h as it marginalizes over all possible values for H and we can ignore it if we are only interested in the posterior distribution up to proportionality. Thus, one often sees Bayes' Theorem abbreviated as

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}$$

2.1.3 *Dependencies between random variables*

Two random variables X and Y are said to be *independent* if and only if

$$P(X, Y) = P(X)P(Y)$$

A joint distribution over independent random variables can be defined through their marginal distributions. This simplifies the definition as, rather than specifying $|\mathscr{X}| \times |\mathscr{Y}|$ probabilities for all possible assignments of values to X and Y, we only have to define $|\mathscr{X}| + |\mathscr{Y}|$ probabilities.

A weaker version of independence is that of *conditional independence*. X and Y are said to be *conditionally independent given* Z if and only if

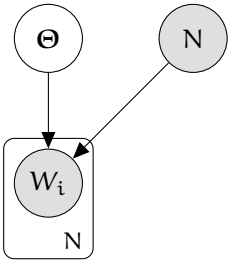
$$P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z)$$

Again, such an assumption allows us to specify a conditional joint distribution over two variables using fewer parameters. Thus, conditional independence and independence assumptions allow us to define joint distributions in terms of "small pieces" (Murphy, 2012, p. 32), i.e. several distributions over (possibly singleton) subsets of all variables. This forms the basis of *generative probabilistic models*.

2.2 GENERATIVE PROBABILISTIC MODELS

A generative probabilistic model (Koller and Friedman, 2009) defines a joint distribution over a set of random variables by specifying a *generative process* through which an assignment of values to all random variables can be generated. Intuitively, a generative process is an algorithm that gives rise to some data by randomly determining the value of the random variables in a way that reflects a specific set of conditional independence assumptions.

Concretely, let us imagine that we want to generate a sequence of words coming from some finite alphabet $\mathscr{W}$. We view this as determining the value of a sequence random variable $\mathbf{W} = W_1, \ldots, W_n$ and assume the following generative process. First, sample parameters Θ to get a specific categorical distribution over $\mathscr{W}$. Then, sample the length of



$$\Theta \sim \mathsf{Dir}(\boldsymbol{\alpha}) \tag{2.1}$$

$$N \sim \mathsf{Geom}(0.5) \tag{2.2}$$

$$W_i \mid N \sim \mathsf{Cat}(\theta) \qquad \text{for } 1 \leqslant i \leqslant N \tag{2.3}$$

Figure 2.1: A probabilistic model for a sequence of words over some finite vocabulary. N is the number of words in the sequence, Θ a concrete distribution over words, and each W_i on of the words in the sequence.

the sequence N that is being generated. Finally, generate the value for the sequence random variable by generating the value for each of its elements W_i by making an independent draw from $\mathsf{Cat}(\Theta)$. Using my notation, this generative process can be expressed by equations 2.1 to 2.3 in Figure 2.1.

A visual representation of the steps in a generative process is given by a *directed graphical model* (Koller and Friedman, 2009). The directed graphical model corresponding to equations 2.1 to 2.3 is also given in Figure 2.1. It has a node corresponding to each of the random variables, and an arrow going from a node labeled X to a node labelled Y if and only if, according to the generative process, we sample the value of Y conditional on a specific value of X. Thus, there are arrows going from Θ to W_i and, in so far as the number of W_i depends on the value of N, from N to W_i. I am using *plate notation* (Koller and Friedman, 2009, p. 216ff) to compactly represent all the $W_1, \ldots, W_N$ that make up the sequence random variable $\boldsymbol{W}$. Briefly, a plate is box with subscript n in the lower right corner which indicates that there are n copies of the nodes inside the box.

The directed graphical model represents the steps in the generative process in the sense that we can determine values for all nodes by beginning 'at the top', sampling values for the nodes that have no parents, and move our way down to the nodes that have no children, at each step sampling values for all the nodes whose parents' values have already been determined.[5] Thus, we can read off the graphical model that we first generate N and Θ before generating each W_i conditional on Θ.

Coming back to the goal of defining a joint distribution, the generative process provides a way of breaking up the joint probability dis-

5 This is also known as *ancestral sampling*.

tribution over all the variables N, Θ and $\boldsymbol{W} = W_1, \ldots, W_N$ using the distributions that define the generative process. Here,[6]

$$\begin{aligned} P(\Theta = \theta, \boldsymbol{W} = \boldsymbol{w}) &= P(\Theta = \theta) P(N = |\boldsymbol{w}|) \prod_{i=1}^{|\boldsymbol{w}|} P(w_i \mid \theta) \qquad (2.4) \\ &= \frac{\Gamma(\sum_{i=1}^{|\boldsymbol{\alpha}|} \alpha_i)}{\prod_{i=1}^{|\boldsymbol{\alpha}|} \Gamma(\alpha_i)} (\prod_{i=1}^{|\boldsymbol{\alpha}|} \theta_i^{\alpha_i - 1}) 0.5^{2|\boldsymbol{w}|} \prod_{i=1}^{|\boldsymbol{w}|} \theta_{w_i} \end{aligned}$$

I call an assignment of values to all random variables of a model a model state. The state space of a model is the set of all possible states.

From the joint distribution over all random variables defined by a model, one can derive "the posterior probability of some variables given evidence on others" (Koller and Friedman, 2009, p. 5). This is called *performing posterior inference* and, essentially, corresponds to learning about plausible values for some variables from the evidence.

To illustrate, we treat $\boldsymbol{W}$ as observed and compute $P(\Theta \mid \boldsymbol{W})$ which is the posterior distribution of Θ given $\boldsymbol{W}$. To explicitly indicate which variables one wants to infer posterior distributions for and which variables one considers as observed, the nodes corresponding to the observed variables are shaded in a graphical model as is done for W_i and N in Figure 2.1. I call the shaded random variables *observed* and the unshaded *latent* variables. Generally speaking, then, *Bayesian inference* is the task of inferring the posterior distribution over latent variables given values for the observed ones.

2.2.1 *Inference under a model*

We illustrate the concrete inference task of determining $P(\Theta \mid \boldsymbol{W} = \boldsymbol{w}_{1:n})$, the *posterior distribution over* Θ *given a specific sequence of observed words* $\boldsymbol{w}_{1:n}$. Unlike the joint distribution, we cannot directly read off this posterior distribution from the model definition but we can derive it using Bayes' Theorem:

$$\begin{aligned} P(\Theta = \theta \mid \boldsymbol{W} = \boldsymbol{w}_{1:n}) &\propto P(\boldsymbol{w}_{1:n} \mid \theta) P(\theta) \\ &= (\prod_{i=1}^{n} P(w_i \mid \theta)) P(\theta) \\ &\propto (\prod_{w' \in \mathcal{W}} \theta_{w'}^{C(w', \boldsymbol{w})}) (\prod_{w' \in \mathcal{W}} \theta_{w'}^{\alpha_{w'} - 1}) \\ &= (\prod_{w' \in \mathcal{W}} \theta_{w'}^{C(w', \boldsymbol{w}) + \alpha_{w'} - 1}) \\ &\propto \mathrm{Dir}(\boldsymbol{\alpha} + C(\boldsymbol{w})) \end{aligned}$$

where I write $C(x, \boldsymbol{w})$ for the number of times with which the specific value x occurs in a sequence of values $\boldsymbol{w}$. $C(\boldsymbol{w})$ is short for an entire

6 As N has to be identical to the length of $\boldsymbol{W}$, I do not explicitly mention it on the left-hand side.

vector of counts that for every $w' \in \mathscr{W}$ gives the number of times with which w' occurs in $\boldsymbol{w}$.

The first line is an application of Bayes' Theorem and every following line uses the independence assumptions of the model, the definition of the resulting distributions (see Table 2.1) and algebraic manipulation to simplify the expression. The end result is, up to proportionality, a Dirichlet distribution with parameters $\boldsymbol{\alpha} + \mathrm{C}(\boldsymbol{w})$. This is because this is a *conjugate model* which "informally [means] that the prior [...] has the same form as the likelihood" (Murphy, 2012, p. 287).

Concretely, the Dirichlet distribution over Θ and the categorical likelihood have the same form; hence, the posterior distribution will also have the same form and, just like the prior, is a Dirichlet distribution. This can be written as

$$\Theta \mid \boldsymbol{w} \sim \mathsf{Dir}(\boldsymbol{\alpha} + \mathrm{C}(\boldsymbol{w}))$$

which states that, conditional on $\boldsymbol{w}$, Θ is distributed to a Dirichlet distribution whose parameters are $\boldsymbol{\alpha} + \mathrm{C}(\boldsymbol{w}$, i.e. the sum of the prior parameters and the count vector for the observed data. One also says that the count vector is a *sufficient statistics* for the data: if one knows the count vector corresponding to a sequence of observations, that completely sums up all the information contained in these observations.

For a rigorous but accessible discussion of conjugacy, see Murphy (2012, chapter 9).

2.2.1.1 *Point estimates and posterior predictive distribution*

The posterior distribution sums up everything that, under the particular assumptions built into a model, can be learned about Θ from any given sequence of words $\boldsymbol{w}$. Yet, it is often useful to summarize a posterior distribution by a single representative value. Two *point estimates* which are commonly used for this.

The single most probable value according to the posterior distribution is called the maximum a posteriori or MAP hypothesis and, staying with our current example, is defined as

$$\hat{\theta} = \arg\max_{\theta} \mathsf{P}(\theta \mid \boldsymbol{w})$$

While the MAP estimate has an intuitive interpretation as the most probable value for a random variable according to its posterior distribution, it also has shortcomings. For one thing, some distributions do not have a mode in which case the MAP is undefined. Secondly, the MAP may be uncharacteristic of the full posterior distribution if most of the probability mass is spread over a very large range of plausible values (see Murphy, 2012, p. 150). For most cases, these problems are addressed by considering the *expected value* of a random variable according to its posterior distribution, the posterior mean which is defined as

$$\tilde{\theta} = \int_{\Delta} \theta \mathsf{P}(\theta \mid \boldsymbol{w}) \mathrm{d}\theta$$

This is, essentially, a weighted sum in which each particular value is considered according to its posterior probability. The big advantage of the posterior mean is that it *averages* over the posterior and, in cases where the posterior is very flat, provides a more representative estimate of what a 'typical' value looks like.

Even though identifying the MAP and mean of a posterior distribution generally involves solving non-trivial maximization or integration problems, in cases where the posterior distribution has a known analytical form such as the Dirichlet we can often calculate these values analytically, see Table 2.1. In particular, for a Dirichlet distribution with parameters $\boldsymbol{\alpha}$, the single most probable value $\hat{\theta}_x$ for the probability of outcome x and its expected value $\tilde{\theta}_x$ are

$$\hat{\theta}_x = \frac{\alpha_x - 1}{\sum_{x'}(\alpha_{x'} - 1)}$$
$$\tilde{\theta}_x = \frac{\alpha_x}{\sum_{x'} \alpha_{x'}}$$

The expected value arises naturally in the *posterior predictive distribution* which is used to predict the probability of the $(n+1)^{th}$ observation on the basis of the n previously made observations:

$$\begin{aligned} P(W_{n+1} = w \mid \boldsymbol{w}_{1:n}) &= \int_\Delta P(w \mid \theta) P(\theta \mid \boldsymbol{w}_{1:n}) d\theta \\ &= \frac{C(w, \boldsymbol{w}_{1:n}) + \alpha_w}{\sum_{w'}(C(w', \boldsymbol{w}_{1:n}) + \alpha_{w'})} \end{aligned} \tag{2.5}$$

Using the posterior predictive distribution corresponds to first determining the posterior distribution of the parameter that governs the sequence, here, $P(\Theta \mid \boldsymbol{w}_{1:n})$; and then using the expected value of this posterior distribution as the basis of your next prediction.

2.2.2 *Integrating out a random variable*

The posterior predictive distribution also arises when a random variable is integrated out or *collapsed* from a model (Liu, 1994). An obvious reason for doing this is if some variables are not of direct interest and are only required for the definition of the model.[7] Thus, we may be interested in the model only in so far as it generates word sequences – that is, rather than the full joint distribution $P(\Theta, \boldsymbol{W})$ we may primarily

7 In statistics, these kinds of variables are known as 'nuisance' variables: "In many problems there is no interest in making inferences about many of the unknown parameters, although they are required in order to construct a realistic model. Parameters of this kind are often called *nuisance parameters*" (Gelman et al., 2014, p. 63).

be interested in a marginal distribution $P(\boldsymbol{W})$. Recall that one can derive this from equation 2.4 by marginalization:[8]

$$P(\boldsymbol{w}_{1:n}) = P(N = n) \int_{\Delta} \prod_{i=1}^{n} P(w_i \mid \theta) P(\theta \mid \boldsymbol{w}_{1:i-1}) d\theta \tag{2.6}$$

In this case, the integral has a closed form solution which can be derived by decomposing the left hand side using the chain rule and repeatedly applying the posterior predictive distribution (Murphy, 2012, p.160):

$$\begin{aligned} P(\boldsymbol{w}_{1:n}) &= P(w_1) \prod_{i=2}^{n} P(w_i \mid \boldsymbol{w}_{1:i-1}) \\ &= \frac{\Gamma(\sum_{x \in \mathcal{W}} \alpha_x)}{\Gamma(\sum_{x \in \mathcal{W}} C(x, \boldsymbol{w}_{1:n}) + \alpha_x)} \prod_{x \in \mathcal{W}} \frac{\Gamma(C(x, \boldsymbol{w}_{1:n}) + \alpha_x)}{\Gamma(\alpha_x)} \end{aligned} \tag{2.7}$$

where Γ is the Gamma-function, the real valued extension of the factorial function $n! = \prod_{i=1}^{n} i$.

As we no longer can factor $P(\boldsymbol{w}_{1:n}) = \prod_{i=1}^{n} P(w_i)$ but have to use, instead, equation 2.7, integrating out Θ has *introduced dependencies between the* W_i *variables* – this reflects the fact that the W_i were only conditionally independent given Θ. However, equation 2.7 also shows that despite these dependencies, the entire sequence of words is *exchangeable* – every possible permutation of words will be assigned the same probability. I return to this point in section 2.3.3.

2.2.3 *Collapsed and uncollapsed model*

Equation 2.7 shows how one can express equation 2.6 without explicitly mentioning Θ at all. Hence, one can consider 2.6 as corresponding to a model whose state space only consists of all possible sequences of words, i.e. which does not include a random variable Θ to begin with. In this sense, integrating out Θ from the model in Figure 2.1 gives rise to a 'collapsed model' whose state-space is strictly smaller than that of the 'uncollapsed model'.

Every state of the collapsed model corresponds to a set of states under the uncollapsed model. Here, a particular sequence of words $\boldsymbol{w}$ which is a single state for the collapsed model corresponds to the set of states $\{\langle \boldsymbol{w}, \theta \rangle\}$ where θ ranges over all possible values of Θ.

While this is a rather conceptual point, it will be useful in understanding the relation between the non-parametric word segmentation models that I now introduce and the Chinese Restaurant representation under which the inference algorithms for these models operate, and I come back to it in section 2.3.3.

8 We use Δ as shorthand for the set of all probability vectors that the variable we integrate over, here Θ, can take on.

2.3 WORD SEGMENTATION MODELS

Word segmentation is the task of identifying a latent sequence of words $\boldsymbol{w} = w_1, \ldots, w_m$, $w_i \in \Sigma^+$ that can account for an observed sequence of segments $\mathbf{s} = s_1, \ldots, s_n, s_i \in \Sigma$.

For example, letting $\mathbf{s} = \langle t, h, e, d, o, g \rangle$ we want identify sequences of words such as $w_1 = \text{the}, w_2 = \text{dog}$ which, when concatenated, yield the observed sequence. This can be cast as a probabilistic inference task by defining a probabilistic model that generates sequences of words and calculate the posterior distribution over these sequences given the observed sequence of unsegmented segments.

If we assumed the words come from some finite vocabulary, the model we just defined could be applied to this task: it defines a probability distribution over all possible sequences of words over some finite vocabulary, and consequently it assigns a probability to every possible sequence of words that yields the observed sequence when concatenated. The problem we are interested in, however, does not assume that we know the lexicon beforehand, mimicking the kind of problem human infants have to tackle when they acquire their first language: solving the problem of identifying the words which make up the lexicon of a language jointly with identifying the "correct" segmentations of the unsegmented input.

One possibility is to assume a very large but still finite vocabulary $\mathscr{W}$, for example all elements of Σ^+ up to a certain length. Then, we can try to infer a joint posterior over word sequences and the probability distribution over these words, hoping that most of these 'words' will be assigned probabilities close to 0 by all θs that have high posterior probability.[9]

A more elegant (and ultimately more efficient) approach is to account for the infinity of the space of possible words and use a *non-parametric* model, that is, a model that is defined by an infinite number of parameters. Goldwater (2007) proposed models based on the *Dirichlet Process* (Ferguson, 1973), a generalization of the Dirichlet distribution that can act as a prior on distributions with countably infinite support in the following sense: every draw from a Dirichlet Process (DP) is a distribution over a (possibly infinite) discrete set.

2.3.1 *The Dirichlet Process*

A DP is defined by a *base distribution* H which defines the possible support for draws from the DP and a *concentration parameter* α that

9 Essentially, this is the approach taken by Goldwater (2007, chapter 4) where the distribution over possible morphological stems and suffixes is modeled as categorical distributions over the 22,396 unique prefix and 21,544 unique suffix strings determined that occur in the data set she considers. This is also the strategy underlying *variational* inference for non-parametric models as in Cohen (2011).

controls how many distinct outcomes will be assigned non-negligible probability by a draw from a DP. I write

$$\begin{aligned} G &\sim \mathsf{DP}(\alpha, H) \\ X_i \mid G &\sim G \end{aligned}$$

to indicate that G is drawn from a DP and that several X_i variables are distributed according to G. If H is a distribution with support $\mathscr{H}$, G is a distribution whose support is a (possibly infinite) discrete set $\mathscr{G} \subseteq \mathscr{H}$.[10]

A constructive way of characterizing G is through the *stick-breaking construction* (Sethuraman, 1994). First, generate an infinite sequence of probabilities $\theta_{1:\infty} \sim \mathsf{GEM}(\alpha)$. The GEM distribution is defined over infinite probability vectors and favors vectors in which all but the first few k components of θ are so close to 0 as to be negligible. The definition of the GEM distribution makes use of the idea of unit stick from which smaller and smaller pieces are broken off, giving the construction its name (Buntine and Hutter, 2010).

Then, draw an infinite number of *atoms* $\psi_{1:\infty}$ from H, yielding the possibly infinite but discrete set $\mathscr{G}$ which is the support of G. Then, the distribution $P(X)$ for any $X \sim G$ is

$$P(X = x) = \sum_{i=1}^{\infty} \theta_i \mathbb{1}[\psi_i = x] \tag{2.8}$$

where $\mathbb{1}[x = y]$ is 1 iff $x = y$ and 0 otherwise. This is similar to a categorical distribution except that we have to consider an infinite number of possible outcomes and account for the possibility that multiple atoms correspond to the same outcome. In this sense, the $\mathrm{DP}(\alpha, H)$ can be thought of as a generalization of the Dirichlet prior for infinite distributions.

This constructive definition shows that G is characterized in terms of an infinite number of parameters $\theta_{1:\infty}, \psi_{1:\infty}$, raising the question how to handle we can practically represent any individual G.[11] One idea is to *truncate* the infinite vectors θ and ψ as is done, for example, in variational approaches such as Cohen (2011, Chapter 6) and is very similar to the idea of using a categorical distribution with a very large but finite support which I briefly discussed above. I follow Goldwater (2007) and Johnson et al. (2007b) and use an alternative solution that *integrates out* G and uses the *Chinese Restaurant* representation.

10 Equality holds in cases where $\mathscr{H}$ is finite or countably infinite. If $\mathscr{H}$ is finite, the model can equivalently be defined in terms of a Dirichlet distribution.

11 Incidentally, note that the stick breaking construction is *not* a constructive characterization of the DP itself but only for draws from a DP.

2.3.2 *The Chinese Restaurant Process*

Consider the model defined by

$$G \sim DP(\alpha, H) \tag{2.9}$$

$$X_i \mid G \sim G \tag{2.10}$$

that defines the joint distribution

$$P(G, \mathbf{X}_{1:n}) = P(G) \prod_{i}^{n} (P(X_i \mid G)) \tag{2.11}$$

This is similar to equation 2.4. However, in this case we cannot directly work with this joint distribution because G is an infinite object. Thus, we integrate out G from equation 2.11, just as we integrated out Θ in equation 2.6 to induce a distribution over only the X_i:

$$P(X_1, \ldots, X_n) = \int_{\Delta} P(G) \prod_{i=1}^{n} P(X_i \mid G) dG$$

As before, this can be simplified through successive application of the *posterior predictive distribution* of X_i given $\mathbf{X}_{1:i-1}$ which takes the following form:

$$P(X_i = x \mid \boldsymbol{x}_{1:i-1}) = \\ = \frac{C(x, \boldsymbol{x}_{1:i-1}) + \alpha H(x)}{\alpha + i - 1} \tag{2.12}$$

The generative process induced by sequentially generating values according to equation 2.12 is called the *Chinese Restaurant Process* (CRP) and is commonly defined using the following metaphor:[12] consider the X_i as customers waiting in line for a Chinese Restaurant with an infinite number of tables. The assignment of each customer X_i to a table is recorded by a random variable Z_i and associated with each table j is a dish or label L_j.

The values for the sequences of random variables $\mathbf{X}, \mathbf{Z}, \mathbf{L}$ are determined by this algorithm:

- for each customer X_i
  1. if
     a) $i = 1$, $Z_i = 1$, that is, the first customer sits at the first empty table;
     b) else, sample a value for Z_i from

$$P(Z_i = k \mid \boldsymbol{z}_{1:i-1}) = \begin{cases} \frac{n_k^{z_{1:i-1}}}{\alpha + i - 1} & \text{if } k \leqslant K(\boldsymbol{z}) \\ \frac{\alpha}{\alpha + i - 1} & \text{if } k = K(\boldsymbol{z}) + 1 \end{cases} \tag{2.13}$$

12 We are using the *labeled Chinese Restaurant Process* exclusively here. The original CRP directly takes the number of a table as its label, generating distributions over the natural numbers rather than arbitrary discrete sets.

where $K(z_{1:i-1}) = \max(z_{1:i-1})$, i.e. the number of tables occupied by the first $i-1$ customers; and $n^z_k = |\{j \mid z_j = k\}|$, i.e. the number of customers already sitting at the k^{th} table according to the assignments in z. Thus, X_i sits at an already occupied table with probability proportional to the number of customers sitting at this table, and at the next empty table with probability proportional to α;

2. if X_i sits at empty table j, sample a value for L_j from H;
3. set the value of X_i to the value of L_{Z_i}, that is, the dish served at the table at which X_i was seated. Hence all customers sitting at a table have the same value.

The sequential sampling scheme induces a rich-get-richer dynamic for table occupations: tables which already have many customers sitting at them will more strongly attract new customers than tables with fewer customers. α controls the overall number of tables that will be occupied, with large values favoring many and small values favoring few occupied tables.

A detailed illustration is given in Figure 2.2. In addition to the generated values $\mathbf{x}$, it shows the table indicators $\mathbf{z}$ and the table labels $\mathbf{l}$ (drawn directly into the respective table). The probability of each seating choice is given below each generated value, and the probability of the entire seating arrangement is the product of the individual seating probabilities.

Note that the CRP does *not* define a distribution directly on sequences of values $\mathbf{x}$; rather, it defines a joint distribution over the table indicator variables $\mathbf{Z}$ and the table label variables $\mathbf{L}$, and each assignment $\mathbf{z}, \mathbf{l}$ uniquely determines a sequence of values $\mathbf{x}$.[13]

The mapping from $\mathbf{z}, \mathbf{l}$ is generally many-to-one: to illustrate, in addition to the specific seating arrangement in Figure 2.2 the customers corresponding to X_1 and X_3 could have sat at their own tables or the customer corresponding to X_4 could have joined the other two customers. All of these different seating arrangements correspond to the same observation sequence $\mathbf{x}$. To my knowledge, there is no closed form expression for the probability of an observed sequence of values $\mathbf{x}$ that sums over all

13 For models in which the base distribution of the Dirichlet Process is fixed, the $\mathbf{Z}$ and $\mathbf{L}$ can be ignored and everything can be described only in terms of the $\mathbf{X}$ (see Goldwater, 2007)). As this does only hold for one particular kind of model, I limit discussion to the more general if slightly more complicated case. In particular, note that even a Unigram model in which inference for the parameters of the base distribution is performed needs to be described in terms of $\mathbf{Z}$ and $\mathbf{L}$, see section 2.3.4.1.

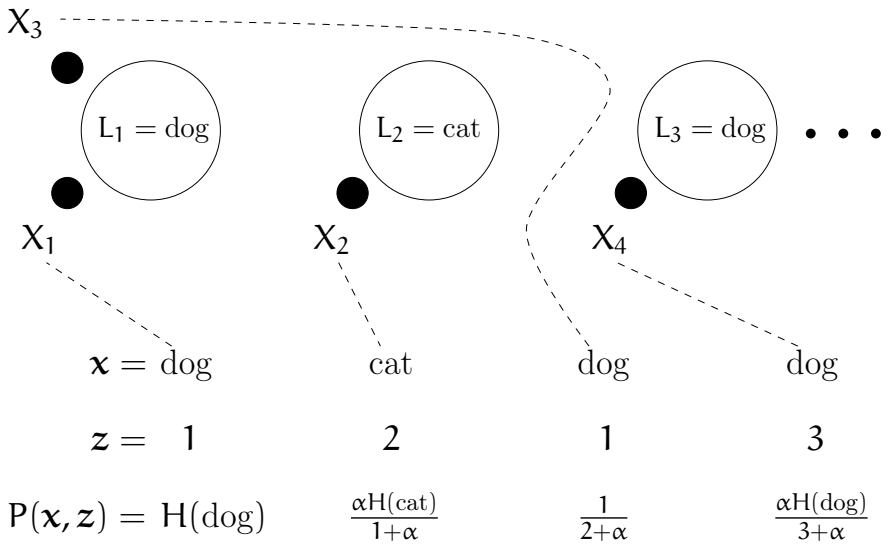


Figure 2.2: Illustration of a Chinese Restaurant Process (CRP). There is an infinite number of tables but only a finite number of them is occupied at any point in time. Every table 'serves' a possible value sampled from H, and all customers (black dots, corresponding to random variables) sitting at a table take on the same value. The first customer who sits at a table 'orders' by sampling from the base distribution H, every consecutive customer does not need to order again. $\mathbf{x}$ is the sequence of generated values, and $\boldsymbol{z}$ captures the 'seating-arrangement', that is, it records for every random variable at which table it was seated. Note that the CRP defines a joint distribution over values and seating arrangements, not just over sequences and values.

possible seating arrangements, but the joint distribution of any seating arrangement can be analytically calculated as[14]

$$P(\mathbf{Z} = \boldsymbol{z}, \mathbf{L} = \boldsymbol{l}) = \frac{\Gamma(\alpha)}{\Gamma(|\boldsymbol{z}| + \alpha)} \prod_{k=1}^{K(\boldsymbol{z})} \left(\alpha H(l_k)(n_k - 1)!\right) \tag{2.14}$$

This shows that the $\boldsymbol{z}, \boldsymbol{l}$ and, consequently, the $\mathbf{x}$ are *exchangeable* – no matter how we permute the sequence of generated values, the probability assigned to the entire sequence is unaffected.

2.3.3 *Relationship between the CRP, the DP, and De Finetti's theorem*

The CRP arises because rather than actually representing the distribution $G \sim \mathrm{DP}$, we choose to work with a *collapsed model* in which $X_i \sim G$ are not independent but exchangeable. We saw that this introduces additional random variables Z_i and L_j which represent a seating

14 Teh (2006a) does, however, derive a closed form expression for the marginal probability of a restaurant with c customers and t tables, summing over all possible ways of distributing customers across tables.

arrangement in place of the collapsed G. Thus, it is important to keep in mind that one can study this model without using the CRP at all, for example by using a truncated stick-breaking representation (Gelman et al., 2014, p. 552f).

To conclude this discussion, it is worth briefly mentioning De Finetti's theorem (de Finetti, 1990) which provides an additional view on the relationship between the CRP and the DP. The mathematical details of this theorem go well beyond the scope of this thesis but, at a very high level, De Finetti's theorem "establishes that any collection of exchangeable random variables has a representation as a mixture distribution – in general an infinite mixture" (Blei et al., 2003, p. 994).

Here, the sequence of exchangeable random variables are the X_i variables, and the (in fact infinite) mixture distribution is G as defined in equation 2.8.

In some more detail, de Finetti's theorem states that for every exchangeable sequence of random variables $X_1, \ldots, X_n$ there exists a (possibly infinite) mixture distribution G such that, conditional on G, the X_1 are independent and such that(see Bernardo, 1996)

$$P(X_1, \ldots, X_n) = \int \prod_{i=1}^{n} P(X_i \mid G) P(G) dG$$

The left-hand side corresponds to the joint distribution over $X_1, \ldots, X_n$ according to the CRP; the right hand side explicitly mentions the draw from the DP which, however, is integrated out.

Note that de Finetti's theorem also applies to the discussion of collapsing in the simple model in Figure 2.1. In this case, equation 2.7 is the result of integrating over the distribution Θ which is distributed according to a Dirichlet distribution.

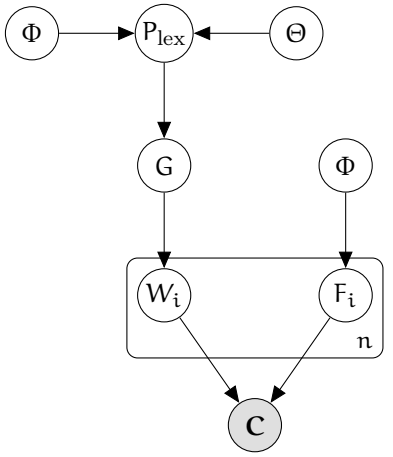
2.3.4 *The Unigram model*

The Unigram model (Goldwater, 2007) assumes that a sequence of W_i variables is generated from a probability distribution over Σ^+ (a probabilistic lexicon G) that is drawn from a Dirichlet Process. The model is defined in Figure 2.3, and the base distribution P_{lex} is defined in Figure 2.4 and discussed below.

Associated with each word W_i is a variable F_i which is drawn from a Bernoulli distribution over $\{s, c\}$ and which indicates whether W_i terminates an utterance (stop) or is non-final (continue). This is equivalent to assuming a geometric distribution over utterance lengths with parameter Φ as in Figure 2.1.[15]

Integrating out G induces a CRP. This introduces additional random variables Z_i which indicate at which table the i^{th} word sits and L_j

15 Using a sequence of Bernouli-distributed indicator variables rather than explicitly drawing the length of each utterance from a geometric distribution makes derivation of the required sampling equations easier.



$$
\begin{aligned}
\Theta &\sim \mathrm{Dir}(\boldsymbol{\alpha}) \\
\Phi &\sim \mathrm{Beta}(a, a) \\
G &\sim \mathrm{DP}(\alpha_0, P_{lex}) \\
R &\sim \mathrm{Beta}(a_0, a_0) \\
W_i \mid G &\sim G \\
F_i \mid \Phi &\sim \mathrm{Bern}(\Phi) \\
\mathbf{C} \mid \boldsymbol{W}, \mathbf{F} &= \textsc{Concat}(\boldsymbol{W}, \mathbf{F})
\end{aligned}
$$

Figure 2.3: Description of the Unigram model for word segmentation. All latent random variables except for W_i and F_i will be integrated out, resulting in a Chinese Restaurant Process representation of the model. F_i indicates whether or not W_i is utterance-final. $\mathbf{C}$ is the actually observed sequence of unsegmented utterances which are the output of simply concatenating all words without spaces and, at the end of utterances, indicating the presence of a boundary.

which indicate the type of all words that sit at the j^{th} table. As was pointed out by Teh (2006a), for inference purposes one does not need to keep track of the actual table to which any individual customer was assigned because "given the dish a customer eats, the actual identity of the table at which the customer sits has no effect on the likelihood of the data" (p. 17f). This is easy to see from equation 2.14 which only depends on the number of customers sitting at each table.

Hence, I follow Teh (2006a) and define the probabilities required for the inference algorithm in terms of the counts given in Table 2.2 instead of explicit values for the Z_i and L_i variables. I write h as shorthand for all sufficient statistics for the conditional distribution in the Unigram model under its CRP representation.

I now provide the conditional distributions that define the generative process in terms of these counts.

$$
P(W_i = x, Z_i = k \mid h) = \begin{cases} \frac{n^h_{x,k}}{\alpha_0 + i - 1} & \text{if } k \leqslant K^h \\ \frac{\alpha_0 P_{lex}(x)}{\alpha_0 + i - 1} & \text{if } k = K^h + 1 \end{cases} \tag{2.15}
$$

As explained by Table 2.2, $n^h_{x,k}$ is the number of customers of type x sitting at table k, and K^h is the total number of currently occupied tables. Marginalizing over Z_i in this equation, we can directly derive the predictive probability for the i^{th} word which is just equation 2.12 using the slightly different notation:

$$
P(W_i = x \mid h) = \frac{n^h_{x,\cdot} + \alpha_0 P_{lex}(x)}{\alpha_0 + i - 1} \tag{2.16}
$$

K^h	$\max(z)$	number of occupied tables in h
$n^h_{w,k}$	$\begin{cases} \lvert\{i \mid z_i = k\}\rvert & \text{if } l_k = w \\ 0 & \text{if } l_k \neq w \end{cases}$	w-customers sitting at table k
$n^h_{\cdot,k}$	$\lvert\{i \mid z_i = k\}\rvert = \sum_w n^h_{w,k}$	customers sitting at table k
$n^h_{w,\cdot}$	$\sum_{i=1}^{K^h} n^h_{w,i}$	total number of w-customers
$n^h_{\cdot,\cdot}$	$\sum_{i=1}^{K^h} n^h_{\cdot,i}$	total number of customers
$n^h_{\#}$	$c(s, \mathbf{f})$	number of utterance boundaries

Table 2.2: Counts used to represent seating arrangements, and their relation to table indicators $\mathbf{z}$ and table labels $\mathbf{l}$. Note that we can completely characterize a seating arrangement in terms of K^h, the number of tables, and $n^h_{w,k}$, the number of word customers sitting at each table, as all other counts we require can be defined in terms of these two counts. Also note that, for simplicity, we consider the counts of word-boundaries to be part of h even though, strictly speaking, in the Unigram model these counts do not correspond to customers.

Finally, the predictive probability for the utterance boundary indicator is

$$P(F_i = s \mid h) = \frac{n^h_{\#} + a_0}{|\mathbf{f}| + 2a_0} \tag{2.17}$$

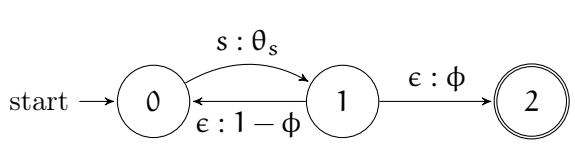
$$P(F_i = c \mid h) = 1 - P(F_i = s \mid h) \tag{2.18}$$

By sequentially sampling values for W_i and F_i using these equations we can generate a sequence of utterances. Finally the observed sequence of segments $\mathbf{C}_{1:m}$ (which also includes observed utterance boundaries) is generated by concatenating all words without spaces. This is achieved by a deterministic function of the W_i and F_i random variables which I abbreviate with Concat.

2.3.4.1 *A base distribution over words*

To complete the definition of the model, we need to specify the base distribution P_{lex} over $\Sigma+$. I use a slight extension of Goldwater (2007)'s Unigram phoneme distribution that makes it possible to perform inference for phoneme probabilities, rather than assuming a uniform distribution over phonemes.

Figure 2.4 shows a Probabilistic Finite State Automaton for this distribution which is defined in terms of a distribution over segments P_{seg} (corresponding to state 0 in Figure 2.4) and a distribution P_{stop} over



$$\Theta \sim \mathsf{Dir}(\boldsymbol{\alpha})$$
$$\Psi \sim \mathsf{Beta}(a, b)$$
$$P_{\text{seg}} = \mathsf{Cat}(\Theta)$$
$$P_{\text{stop}} = \mathsf{Bern}(\Phi)$$

Figure 2.4: Unigram phoneme distribution as Probabilistic Finite State Automaton. State 0 generates a phoneme s according to a categorical distribution $\mathrm{Cat}(\theta)$. State 1 decides whether to generate the end of the word with probability ϕ or whether to generate another segment with probability $1 - \phi$.

$\{\text{S}, \text{C}\}$ that determines the length of words (corresponding to state 1).[16] The base distribution used in Goldwater (2007) and Goldwater et al. (2009) is a special case of this base distribution where, rather than inferring P_{seg} and P_{stop} uniform distributions are used.

We collapse the random variables Φ and Θ, conditioning on the previously generated segments and words instead. Writing $\mathbf{s}$ for all previously generated segments and n_w for the previously generated total number of words, the predictive versions of P_{seg} and P_{stop} are:

$$P_{seg}(s \mid \mathbf{s}) = \frac{C(s, \mathbf{s}) + \alpha_s}{|\mathbf{s}| + \sum_{s' \in \Sigma} \alpha_{s'}} \tag{2.19}$$

$$P_{stop}(\text{S} \mid \mathbf{s}, n_w) = \frac{n_w + a}{|\mathbf{s}| + a + b} \tag{2.20}$$

$$P_{stop}(\text{C} \mid \mathbf{s}, n_w) = \frac{|\mathbf{s}| - n_w + b}{|\mathbf{s}| + a + b} \tag{2.21}$$

Both $\mathbf{s}$ and n_w can be derived directly from h – they only depend on $\mathbf{l}$, the table labels which are generated by the base distribution. This is an instance of a hierarchical model in which the base distribution probabilities are estimated not directly from the generated word tokens W_i but through 'interpolating' type and token frequencies Goldwater et al. (2006).

2.3.4.2 *A linguistically informed base distribution*

While Goldwater et al. (2009) argue that the choice of the base distribution only has a minor impact on segmentation performance, a general recommendation for Dirichlet Process models is to "construc[t] an informative [base distribution] placing high probability on introducing cluster near the support of the data" (Gelman et al., 2014, p. 552). Even though word segmentation differs from typical applications of Dirichlet Process models in that the relevant data – the words that make up a

16 As is common, we describe Probabilistic Finite State Automata as Mealey Machines. If the probability of transitioning from state t to t' while emitting symbols s is p, the arc going from t to t' will be labeled with $s : p$.

segmentation – is latent, previous work in both psycholinguistics (Norris et al., 1997) and computational modeling (Blanchard et al., 2010) has argued for a simple yet substantive possible word constraint to aid word segmentation. Indeed I will show that Goldwater et al. (2009)'s observation needs to be qualified with respect to the Bigram model below.

The simple constraint requires a possible word to contain at least a single syllabic element (commonly a vowel, although in some languages, consonants may function as syllabic elements), thus excluding many possible segmentations that contain 'impossible' words from the models state space. Adding this constraint in a principled way is surprisingly subtle. For example, the strategy of Blanchard et al. (2010) to simply assign probability 0 to sequences that lack a syllabic element will result in an improper probability distribution. Performing the required re-normalization is trivial if the phoneme and stopping probabilities are considered to be fixed but one no longer can integrate over these parameters nor analytically determine their posterior distributions, making inference for them more complicated.

As a mathematically well-defined model that is convenient to work with, I use the distribution defined in Figure 2.5 which has the same number of parameters as the base distribution in Figure 2.4.[17] Basically, the automaton defining the distribution cannot transition into the end state before at least a single syllabic element has been generated. There are other ways of implementing this kind of constraint but I chose this particular formulation because it introduces no additional parameters to the simpler base distribution and still allows for analytic collapsing of the parameters of P_{seg} and P_{stop}.

2.3.5 *Inference for the Unigram model*

The goal of posterior inference is to identify the marginal posterior over segmentations, that is, sequences of words that yield the observed sequence of segments when Concat is applied to them.

$$\mathsf{P}(\boldsymbol{w} \mid \mathbf{C} = \boldsymbol{c}) \propto \mathsf{P}(\boldsymbol{c} \mid \boldsymbol{w})\mathsf{P}(\boldsymbol{w}) \tag{2.22}$$

I briefly discuss why a Dirichlet Process model encourages linguistically meaningful segmentations before discussing in more detail how inference for the marginal posterior can be performed.

SEGMENTATION INTUITION The Dirichlet Process prior encourages distributions in which a few outcomes account for most of the

17 Stronger constraints, for example requiring a syllabic parse, have been used in adaptor grammar based models (Johnson, 2008b; Johnson and Goldwater, 2009). For the Unigram and Bigram model, however, I will rely on this simpler constraint which lends itself to an easy implementation that does not require the use of a parser at any point.

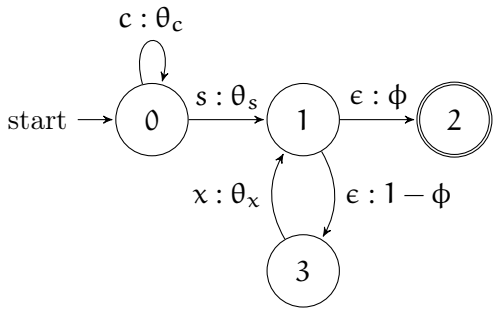


Figure 2.5: Base distribution with a possible word constraint. The parametrization is identical to that of the distribution in Figure 2.4, the sole difference being that transitioning from state 0 to 1 requires generation of a syllabic segment (abbreviated as s) instead of a non-syllabic segment (abbreviated as c). Once a syllabic segment has been generated, the generative process is identical to that of Figure 2.4.

probability mass. In terms of the stick-breaking construction, for most draws from a DP all but the first few elements of $\theta_{1:\infty}$ will be noticeably different from 0. Thinking about this in terms of sequences of words generated by the Chinese Restaurant Process, sequences which comprise *small number of types* that are *used frequently* are preferred although, as is common for probabilistic generative processes, sequences with overall fewer tokens are preferred to those with more tokens.

Taken together, this leads to a trade-off between the number of tokens in an analysis and the number of types in an analysis. A segmentation in which every utterance is treated as a single long 'word' uses too many types and a solution that treats every phoneme as a word too many tokens to attain a high probability by a Bayesian segmentation model. Segmentations that properly balance types and tokens, on the other hand, tend to be both linguistically meaningful and preferred by these models.

MONTE CARLO INFERENCE Calculating the posterior by exhaustive enumeration is infeasible for all but toy examples as the number of possible segmentations grows exponentially with the length of the input.[18] To combat this problem, *Monte Carlo algorithms* (Metropolis and Ulam, 1949) are commonly used to perform (approximate) inference instead. The key insight of Monte Carlo inference is that one can approximate any distribution P by a a set of samples from this distribution $\{s_i\}_{i=1}^{n}, s_i \sim P$ and treat the relative frequency with which each

18 And there is the added complexity of summing over possible seating arrangements. See Figure 3.1 for an illustration of exhaustive enumeration.

value occurred in this sample as its approximate probability according to the distribution:

$$\hat{P}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[s_i = x] \tag{2.23}$$

This intuitively makes sense, and there is indeed a mathematical guarantee by way of the central-limit theorem that the accuracy of these relative frequency approximations increases with the number of samples (Murphy, 2012, p.54f). To generate these samples, we will use *Markov Chain Monte Carlo* (or MCMC) methods.[19]

For an explanation the relevant Markov Chain theory on which MCMC builds, see the excellent discussion of Markov Chains in Murphy (2012, p. 596–600); for accessible discussions of MCMC methods see Murphy (2012, chapter 24), MacKay (2003, chapter 29) and Gelman et al. (2014, chapter 11). Here, I restrict myself to reviewing the specific *Gibbs samplers* of Goldwater (2007).

2.3.6 *Gibbs sampling*

Gibbs sampling Geman and Geman (1984) is an algorithm that generates samples from some distribution by making small random changes to a randomly chosen initial sample. Here, our goal is to generate samples from the posterior over segmentations and seating arrangements given unsegmented utterances. We explain this idea in more detail by considering the concrete case of segmenting the single utterance abab using the Unigram model.

2.3.6.1 *State space of the collapsed model*

Each state of the collapsed Unigram model is a specific seating arrangement h which corresponds to a segmentation $\boldsymbol{w}$. Conditional on the observed sequence $\mathbf{c}$ = abab, the state space consists of all seating arrangements such that concatenating the words in their associated $\boldsymbol{w}$ yields abab. It is important to realize that even though there are only 8 possible segmentations abab, ab ab, a b a b, ab a b, a b ab, a ba b, aba b, a bab, the state space is larger because some of these segmentations correspond to multiple seating arrangements. For example, ab ab can arise form a seating arrangement with either a single table that has two customers or two tables with one customer each.

Rather than thinking about states directly in terms of a seating arrangement, we will think of them in terms of a fixed-length *binary vector* that indicates for every possible position whether or not a word boundary occurs at this position. Concretely, for our example we have

19 In Chapter 3, I discuss an alternative inference algorithm based on a *Sequential Monte Carlo Method* called Particle Filter.

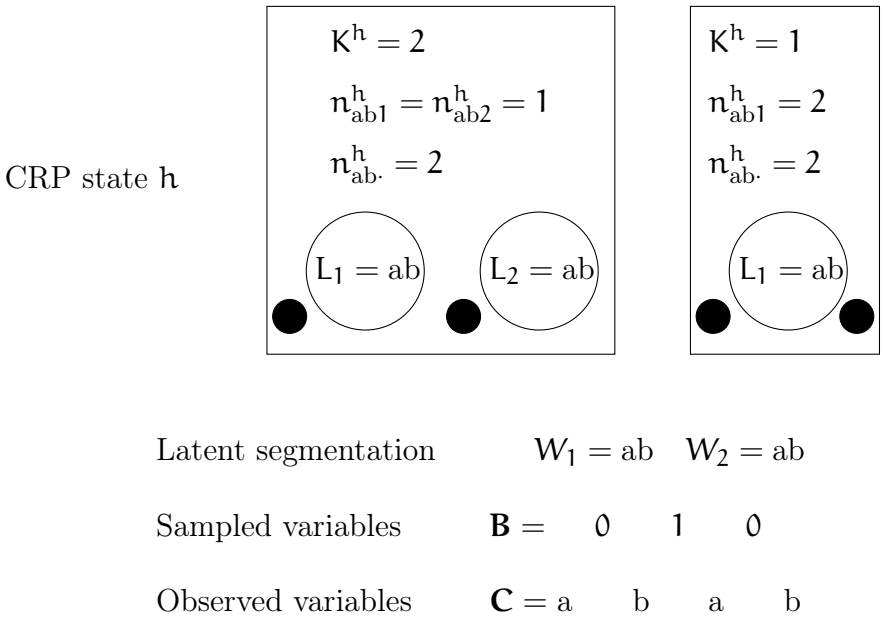


Figure 2.6: Relationship between the fixed number of boundary indicator random variables (Sampled variables), the observed variables C_i, the variables W_i which define the segmentation, and the underlying Chinese Restaurant Process state h in terms of the counts defined in Table 2.2.

three boundary variables B_1, B_2, B_3 that indicate whether or not there is a boundary between the first occurrence of ab, between ba and between the second occurrence of ab, respectively. Figure 2.6 illustrates the relationship between this binary vector representation over which we perform sampling, the latent segmentation $\boldsymbol{w}$ and the seating arrangement introduced by the CRP as a result of collapsing out G. Considering states as sequences of boundary indicators $\mathbf{B}$ has the advantage of providing *a fixed number of random variables* whereas different segmentations may contain different numbers of words.

As discussed above, there is usually a one-to-many relationship between sequences of words and seating arrangements h; Figure 2.6 illustrates two possible underlying CRP states defined in terms of the counts of Table 2.2. Seating arrangements are handled implicitly in the sampler through the addCustomer and removeCustomer functions discussed below, following the idea of Teh et al. (2006).

Relation between collapsed and uncollapsed model

Before I discuss the Gibbs sampler in detail, let us briefly consider how the collapsed representation under which we sample relates to the original model. Recall that even though the state space considered by the sampler does not include a distribution over words such as G as part of its states, the word segmentation model is defined in terms of a draw

from a Dirichlet Process G; and that de Finetti's theorem establishes the equivalence between the CRP and the DP model.

We can recover the joint posterior distribution over $\boldsymbol{W}$ and G from the marginal posterior distribution over segmentations from which the Gibbs sampler produces sampler, and doing this provides a deeper understanding of relationship between the collapsed and the uncollapsed model.

Note that *given a particular sequence of observed words* $\boldsymbol{w}_{1:n}$ which were sampled i.i.d. from $G \sim \mathsf{DP}(\alpha, P_{\text{lex}})$, the posterior distribution for G is (Gelman et al., 2014, p. 547)

$$G \mid w_1, \ldots, w_n \sim \mathsf{DP}(\alpha + n, \frac{P_{\text{lex}} + \sum_{i=1}^{n} \delta_{w_i}}{\alpha + n}) \tag{2.24}$$

where δ_x is the Dirac measure which assigns all mass to x, i.e. $\delta_x(y) = 1$ if and only if $y = x$. In other words, if we know the segmentations we can analytically derive the posterior distribution over G. This is, in essence, identical to how we could analytically calculate the posterior of Θ for the model in Figure 2.1 by simply adding the hyper parameters of the prior and the count vector of the observations.

With this, we can derive the joint posterior distribution over segmentations and G from $P(\boldsymbol{W} \mid \mathbf{C})$, the marginal posterior over segmentations:

$$P(G, \boldsymbol{W} \mid \mathbf{C}) = P(G \mid \boldsymbol{W}, \mathbf{C}) P(\boldsymbol{W} \mid \mathbf{C}) = P(G \mid \boldsymbol{W}) P(\boldsymbol{W} \mid \mathbf{C})$$

where $P(\boldsymbol{W} \mid \mathbf{C})$ is the posterior distribution over segmentations we approximate by sampling under the collapsed representation, and $P(G \mid \boldsymbol{W})$ will be another Dirichlet Process. Taking this one step further, we can derive the marginal posterior distribution over G through marginalization and get

$$P(G \mid \mathbf{C}) = \sum_{\boldsymbol{w}} P(G \mid \boldsymbol{w}) P(\boldsymbol{w} \mid \mathbf{C})$$

where we sum over all possible segmentations of $\mathbf{C}$. This shows that the posterior distribution over G will be a *mixture of Dirichlet Processes*, with one mixture component per latent segmentation. The weight of each mixture component is the marginal posterior probability of the corresponding latent segmentation conditional on just the observed data which, up to proportionality, can be calculated analytically using equation 2.14. Thus, we can write the posterior distribution in the following explicit form:

$$G \mid \mathbf{C} \sim \sum_{\boldsymbol{w}} \mathsf{DP}\left(\alpha + |\boldsymbol{w}|, \frac{\sum_{i=1}^{|\boldsymbol{w}|} \delta_{w_i} + \alpha P_{\text{lex}}}{\alpha + |\boldsymbol{w}|}\right) \tag{2.25}$$

Of course, this result cannot be used for inference as the set of possible segmentations over which the sum would need to be evaluated is infeasibly large and grows exponentially with the size of the corpus.

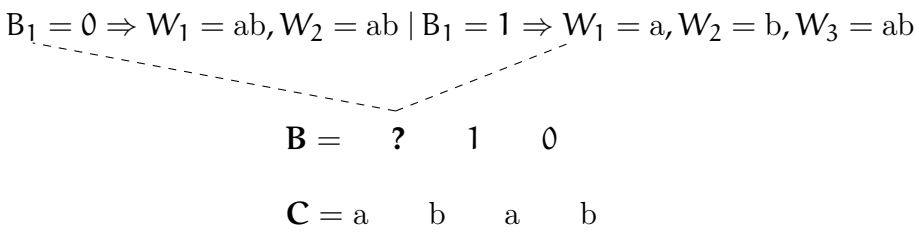


Figure 2.7: Illustration of the local changes involved in resampling the value for a single boundary variable.

Incidentally, this also shows that in the posterior distributions grow more 'complex' as a function of the size of the corpus as the number of mixture components just is the number of possible segmentations. This highlights the *non-parametric* character of the model.

The point of this brief discussion is to highlight that, even though all inference algorithms considered in this thesis (and the literature on Bayesian word segmentation at large) operate in collapsed representations which integrate out the actual distributions in terms of which the model is defined, it is always possible to recover a posterior distribution over these distributions from the marginal posterior distribution over latent structures.

2.3.6.2 *Gibbs sampling for the Unigram model*

A single iteration of Gibbs sampling resamples the value for every boundary variable once, thus generating a new sample (a predicted segmentation $\boldsymbol{w}$ plus an underlying seating arrangement h). To ensure that these samples will be distributed according to the posterior over segmentations, we have to resample B_i according to the *conditional distribution* given the current values for all variables but B_i.

Consider how changing a single boundary variable affects the overall segmentation. Figure 2.7 illustrates the two different segmentations that can arise in our specific example when we go on to resample B_1. The value of this boundary variable only affects the span of the input that reaches from the first boundary to the left of B_1 to the first boundary to its right, here ab. If a boundary is posited, this stretch will be analyzed as a sequence of words $w_1 = \text{a}, w_2 = \text{b}$, if no boundary is posited it will be analyzed as a single long word $w_{1.2} = \text{ab}$. Crucially, the value of B_1 does not affect the identity of any other words in the previous segmentations. The last word in the segmentation will be ab, irrespective of what value will be sampled for B_1.

Because the overall sequence of random variables is exchangeable, we can 'move' all the words on which the two hypotheses agree (i.e., everything except for the small affected part of the input) to the front of the sequence, treating the stretch that is affected by the variable under consideration as if it were the last part of the observed sequence of segments that was generated. Then, we can use the predictive probabil-

ities defined in equations 2.16 and 2.17. This is done by the algorithm in Figure 2.8.

The first step is to determine the affected span and its current analysis. If $B_i = 1$, we need to remove both a w_1 and a w_2 customer from h, otherwise we remove a single $w_{1.2}$ customer. In the example in Figure 2.7, $w_1 = a, w_2 = b$ and $w_{1.2} = ab$.

For these removal operations we use the add/remove-customer functions of Teh (2006b) which are defined in Figure 2.9. ADDCUSTOMER(h, w) randomly samples the table k at which a novel w customer sits in seating arrangement h according to equation 2.13. REMOVECUSTOMER(w) removes a customer from a randomly sampled table k that is labeled w with probability proportional to $n^h_{w,k}$. In Chapter 3, these functions are also used and the probabilities of sitting or removing a customer are required. For the Gibbs sampler described here, the return values of the functions can be ignored.

We then calculate the probability of the two possible values B_i can take. $B_i = 1$ corresponds to positing a boundary and its probability is thus proportional to $P(w_1 \mid h)P(w_2 \mid h, w_1)$, i.e. generating the two words w_1 and w_2 from seating arrangement h. $B_i = 0$ corresponds to generating only a single word $w_{1.2}$ and its probability is thus proportional to $P(w_{1.2} \mid h)$. These probabilities can be calculated using formula 2.16 but there is a minor complication for $B_i = 1$.

This complication is that $P(w_2 \mid h, w_1)$ is the probability of generating w_2 from h after w_1 has been generated. In Goldwater (2007)'s and Goldwater et al. (2009)'s presentation of the sampling algorithm

$$P(w_2 \mid w_1, h) = \frac{n^h_{w_2,\cdot} + \mathbb{1}[w_1 = w_2] + \alpha P_{\text{lex}}(w_2)}{n^h_{\cdot,\cdot} + 1 + \alpha}$$

This formula holds because having generated w_1 increases the number of total customers in h by 1, captured by the additional $+1$ in the denominator. And if $w_1 = w_2$, $n^h_{w_2}$ will also have increased, hence the conditional $+1$ in the numerator. This holds, however, only if the base distribution is fixed as in Goldwater (2007) and Goldwater et al. (2009). If P_{lex} is estimated from the words that have been generated – as in the base distribution in Figure 2.4 – having generated w_1 can also affect the probability of w_2 through P_{lex} and the update formula does not hold.

Instead, we calculate $P(w_2 \mid h, w_1)$ by performing a 'temporary update' of h. We call ADDCUSTOMER(w_1, h) and then just calculate $P(w_2 \mid h)$, calling REMOVECUSTOMER(w_1, h) after the calculation to undo the change to h.[20] The probability of $B_i = 0$ is proportional $P(w_{1.2} \mid h)$.

20 A subtle detail is the fact that we treat the table label indicators as latent rather than explicitly tracking them, following the idea described in Teh (2006b). Thus, we cannot guarantee that the intermediate update step will be exactly undone by calling REMOVECUSTOMER as the identity of the table from which a customer will be removed is determined randomly. Similarly, our update operation will only consider one of the potentially many possible ways in which w_1 might have been added to h.

procedure SAMPLEBOUNDARY(i, h)
  determine affected span $w_{1..2}$ and w_1, w_2
  if $B_i = 1$ **then**
    REMOVECUSTOMER(w_1, h)
    REMOVECUSTOMER(w_2, h)
  else
    REMOVECUSTOMER($w_{1..2}, h$)
  end if
  $p_b = P(w_1 \mid h) P(\text{C} \mid h)$
  ADDCUSTOMER(w_1, h) ▷ temporary update
  $p_b = p_b P(w_2 \mid h) \begin{cases} P(\text{S} \mid h) & \text{if } w_2 \text{ is utterance final} \\ P(\text{C} \mid h) & \text{if } w_2 \text{ else} \end{cases}$
  REMOVECUSTOMER(w_1, h) ▷ Undo temporary update
  $p_{\neg b} = P(w_{1..2} \mid h) \begin{cases} P(\text{S} \mid h) & \text{if } w_2 \text{ is utterance final} \\ P(\text{C} \mid h) & \text{if } w_2 \text{ else} \end{cases}$
  if NEXTDOUBLE $\times (p_b + p_{\neg b}) \leqslant p_b$ **then** ▷ put boundary
    $B_i = 1$
    ADDCUSTOMER(w_1, h)
    ADDCUSTOMER(w_2, h)
  else ▷ don't put boundary
    $B_i = 0$
    ADDCUSTOMER($w_{1..2}, h$)
  end if

end procedure

Figure 2.8: Algorithm to resample word-boundary B_i given a current seating arrangement h. Whether or not any word is utterance-final is known as utterance-boundaries are considered to be observed.

Another detail is that we need to account for the probability of generating another word ($P(\text{C} \mid h)$) or generating an utterance boundary ($P(\text{S} \mid h)$), respectively, using equations 2.17 and 2.18.

After having calculated both probabilities, we sample the value of B_i according to them and update h accordingly: if $B_i = 1$, we add w_1 and w_2 using ADDCUSTOMER. Otherwise, we only add $w_{1..2}$. Note that ADDCUSTOMER and REMOVECUSTOMER automatically take care of the book-keeping involved required to track the CRP seating arrangement.

To ensure that a Gibbs sampler has converged on the target posterior, it is common to ignore the initial iterations (called burn-in period). Also, one usually only records every x^{th} sample of a run to address the correlation of successive samples. Assessing whether or not a simulation was run long enough to have generated samples from the target posterior is a non-trivial problem. For an extended discussion see Gel-

function ADDCUSTOMER(w, h)
  sample $k \propto \begin{cases} n^h_{w,k} & 1 \leqslant k \leqslant K^h \\ \alpha P_{\text{lex}}(w) & k = K^h + 1 \end{cases}$
  if $k = K^h + 1$ **then** ▷ open new table
    $res = \frac{\alpha P_{\text{lex}}(w)}{n^h_{\cdot,\cdot} + \alpha}$
    $K^h = K^h + 1$ ▷ increment table count
    $n^h_{w,k} = 1$ ▷ set customer count for table k to 1
  else ▷ sit at old table
    $res = \frac{n^h_{w,k}}{n^h_{\cdot,\cdot} + \alpha}$
    $n^h_{w,k} = n^h_{w,k} + 1$ ▷ increment customer count for table k
  end if
  return res ▷ return probability of seating choice
end function

function REMOVECUSTOMER(w, h)
  sample $k \propto n^h_{w,k}$
  $n^h_{w,k} = n^h_{w,k} - 1$
  if $n^h_{w,k} = 0$ **then** ▷ Table became empty
    $K^h = K^h - 1$ ▷ decrement table count
    return $\frac{\alpha P_{\text{lex}(w)}}{\alpha + n^h_{\cdot,\cdot}}$ ▷ return probability of re-opening table
  else
    return $\frac{n^h_{w,k}}{\alpha + n^h_{\cdot,\cdot}}$ ▷ return probability of re-seating customer
  end if
end function

Figure 2.9: Functions to add customers to and remove them from a given seating arrangement h. h is a collection of variables which contains all the variables which are super-scripted with h.

man et al. (2014, chapter 11.4) (Chapter 11.4). I follow previous work such as Goldwater (2007) and Johnson and Goldwater (2009) and rely on visual inspection of trace plots of several independent runs to check whether the Gibbs sampling has converged.

SIMULATED ANNEALING A short-coming of Gibbs sampling is that successive samples can be highly correlated as variables are updated one at a time, conditional on the values of all other variables. This can also lead to the sampler getting trapped in local modes. Goldwater (2007) proposed to use *simulated annealing* to combat this, and I follow her in this.[21]

21 It is interesting to note that for adaptor grammars (see below), Johnson and Goldwater (2009) found no improvement by annealing over and above using hyper parameter sampling. I suspect this finding to depend on properties of the model and, to some part, on the fact that unlike this Gibbs sampler, the adaptor grammar sampler samples entire utterances. In contrast, for both the Bigram and Unigram model, I find hyper parameter sampling with annealing to result in better inference than hyper parameter inference without annealing.

Briefly, simulated annealing introduces a *temperature* T which allows the sampler to explore the state space more freely. Updates are performed by sampling from $P(B_i \mid h^-)^{\frac{1}{T}}$ rather than $P(B_i \mid h^-)$.[22]

For high temperatures T, raising the probabilities to $\frac{1}{T}$ leads to almost uniform distributions and allows the sampler to explore assignments which, according to the true conditional distributions, may be very unlikely to ever be considered; but which may be required to ultimately reach an overall good segmentation. Thus,by starting the sampler in a high temperature and slowly decreasing T to 1, one facilitates convergence to the target posterior by alleviating the problem of local modes. Once the temperature has reached 1, the sampler will produce samples from the target posterior distribution.

The *cooling schedule* used in annealing is important, and I found Goldwater (2007)'s schedule to yield good results. It splits the total number of N annealing iterations into 10 equal sized bins and, assuming start-temperature T_{start} and stop-temperature T_{stop}, defines the temperature at iteration i as

$$\text{TEMPERATURE}(i) = \begin{cases} T_{\text{stop}} & \text{if } i > N \\ \frac{10}{\lfloor \frac{9i}{N} \rfloor - 1} \frac{T_{\text{start}} - T_{\text{stop}}}{9} + T_{\text{stop}} & \text{else} \end{cases}$$

2.3.7 *The Bigram model*

Goldwater (2007) argued that the independence assumption inherent in the Unigram model is unsatisfactory as in real language, adjacent words are not independent of each other. Indeed the Unigram model tends to *undersegment*, an issue also discussed in chapter 3 and chapter 4. Goldwater's proposal to address this was to model Bigram dependencies, as depicted in the graphical model in Figure 2.10. I review the reference description of the Bigram model in Goldwater et al. (2009) which differs slightly from that in Goldwater (2007).

Conceptually, the major difference to the Unigram model is the use of *an infinite number* of 'lexicon' distributions H_w, one for every possible word in Σ^+. Each H_w determines the distribution of words following a token of w. The different H_w are drawn from a single shared Dirichlet Process whose base distribution is a global lexicon G_0 which correspond to the single lexicon G of the Unigram model. Thus, the entire model is a hierarchical Dirichlet Process model (Teh et al., 2006)

Utterance boundaries are treated as special 'words' of type \$, $\$ \notin \Sigma$ so that they can serve as context for other words. This requires a slightly changed base distribution $P_{\text{lex}'}$ which allows for the generation of these boundary words with probability $p_\$$ (Goldwater et al., 2009). This is achieved by changing the base distributions for the Unigram model by adding two additional states as in Figure 2.11. The first state either

22 I use h^- as shorthand to indicate that the seating arrangement according to which B_i is resampled excludes customers associated with B_i, as discussed above.

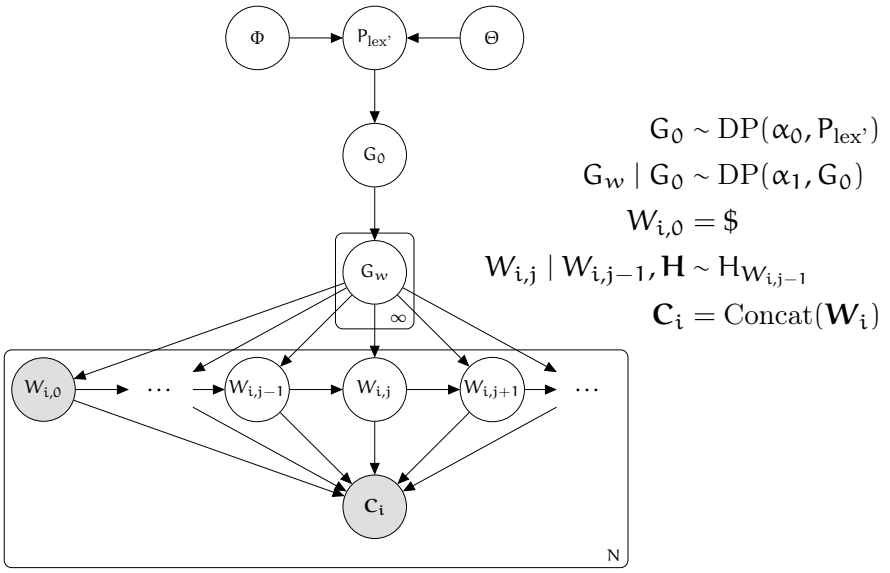


Figure 2.10: Description of the Bigram model for word segmentation. All latent variables except for W_i will be integrated out, resulting in a Chinese Restaurant Franchise representation of the model (see Figure 2.12). $\mathbf{C}_i$ is the observed sequence of segments for utterance i which is the result of concatenating all words $\mathbf{W}_i = W_{i,0}, \ldots, W_{i,m}$. The first 'word' for each utterance is observed as it is a special boundary symbol \$ that is not part of Σ^+. Also, in addition to the global lexicon G_0 there is an infinite number of distributions G_w, one for each $w \in \Sigma+$.

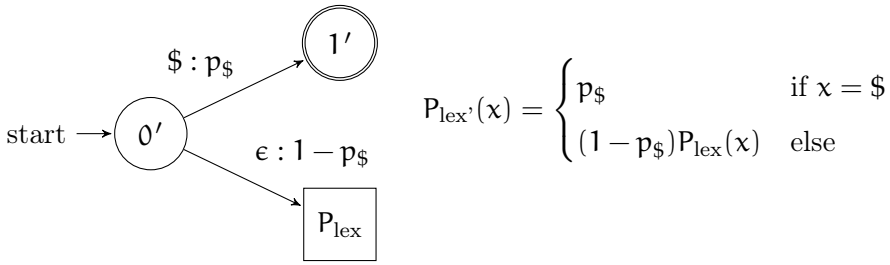


Figure 2.11: Base distribution for the Bigram model. In terms of a probabilistic finite state automaton, we change the automata in Figure 2.4 or 2.5 by changing the start state to the new state $0'$ from which we transition either into the initial state of the old automaton with probability $1 - p_\$$ or generate a boundary word \$ with probability $p_\$$ to transition into the new final state $1'$.

generates an utterance boundary and terminates, or it transitions into the automata of Figure 2.4 or Figure 2.5.

The generative process for a single utterance is as follows. The first word of each utterance $W_{i,0} = \$$. The first 'real word' for each utterance is drawn from $H_\$$, the lexicon for utterance-initial words. An utterance terminates as soon as another instance of \$ is generated.

Integrating out the random variables H_w and G_0 induces a *Chinese Restaurant Franchise* representation (Teh et al., 2006). Rather than

a single seating arrangement as in the Unigram model, for the Bigram model there is a restaurant for every word type w that has been posited in a segmentation, corresponding to H_w, plus a 'global' restaurant corresponding to G_0 in which each customer corresponds to a table in a word-specific restaurant. This 'Chinese Restaurant Franchise' idea and the dependencies between word-specific and the global restaurant is illustrated in Figure 2.12 which is the Bigram model's analog to Figure 2.6.

For notational simplicity, I now take h to be a collection of individual h_w (the seating arrangement corresponding to H_w) and a single h_0 corresponding to the seating arrangement for G_0. Thus, in Figure 2.12 $n_{\$,\cdot}^{h_{ab}} = 1$, as there is one \$ customer in seating arrangement h_{ab}; and $n_{ab,\cdot}^{h_0} = 2$, as there are two ab customers in seating arrangement h_0.

With this, the predictive probability of a word w following a word w', conditional on the current seating arrangement h is

$$P(W_i = w \mid W_{i-1} = w_0, h) = \frac{n_{w,\cdot}^{h_{w_0}} + \alpha_1 P(w \mid h_0)}{\alpha_1 + n_{\cdot,\cdot}^{h_{w_0}}} \tag{2.26}$$

$$P(W_i = w \mid h_0) = \frac{n_{w,\cdot}^{h_0} + \alpha_0 P_{\text{lex}'}(w)}{\alpha_0 + n_{\cdot,\cdot}^{h_0}} \tag{2.27}$$

Due to the coupling between the global and the word-specific restaurants, $n_{x,\cdot}^{h_0}$ is equal to the number of tables labeled x across all h_w restaurants. This can be easily enforced by defining the additional AddCustomer and removeCustomer methods in Figure 2.13 which will be used to add customers to the word-specific restaurants. w_0 is the word preceding the word w which is added or removed.

In essence, these functions are identical to those in Figure 2.9 but enforce the coupling across the individual seating arrangements h_w for each individual word and the global seating arrangement h_0 as depicted in Figure 2.12. If a new table is opened, the original addCustomer method from Figure 2.9 is called to add a w customer to h_0, the seating arrangement corresponding to G_0. Similarly, in removeCustomer a customer is removed from h_0 if a table becomes empty in the process of removing a customer. Again, these functions take care of all the bookkeeping that is required, facilitating implementation considerably.

With this, the Gibbs sampler for the Bigram model can be derived directly from the sampler for the Unigram model, requiring only minor modification to account for the dependencies between adjacent words and the different handing of utterance boundaries.

Denote by w_l the word preceding the span of text affected by changing boundary variable B_i and by w_r the word following this span, and as before by $w_{1.2}$ the entire sequence (corresponding to $B_i = 0$) and by w_1, w_2 the two-word sequence that results if $B_i = 1$. For the example in Figure 2.7 $w_l = \$, w_1 = a, w_2 = b, w_{1.2} = ab$, and $w_r = ab$. With this, a single boundary can be resampled using the algorithm defined in Figure 2.14.

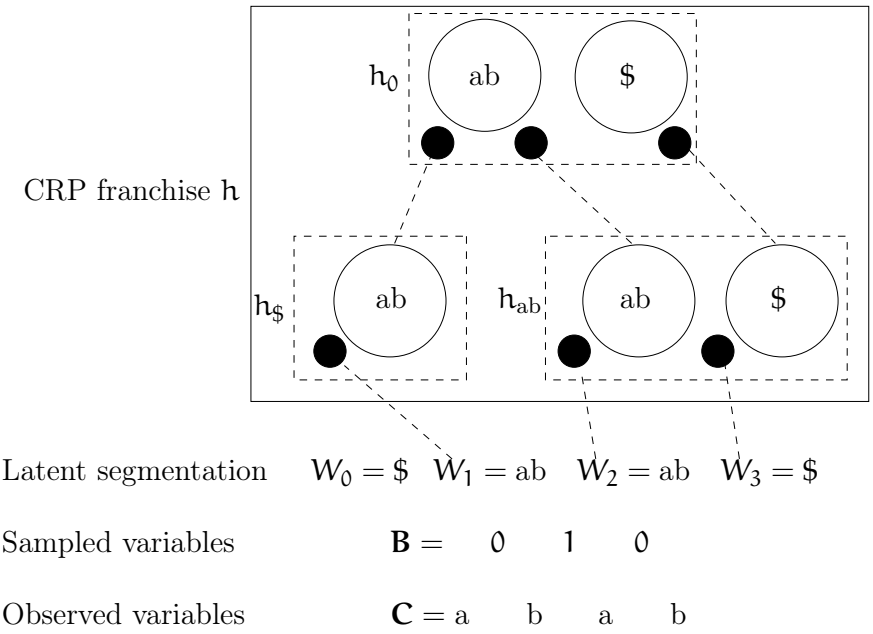


Figure 2.12: Illustration of the relation between random variables for the Bigram model. Rather than a single CRP seating arrangement, the Bigram model yields an underlying CRP *franchise*, a collection of word-specific seating arrangements ($\mathfrak{h}_{\mathbf{ab}}$ and $\mathfrak{h}_\$$) and a global one ($\mathfrak{h}_0$). Note that every W_i variable corresponds to a customer in a word-specific restaurant, and that every table in such a restaurant corresponds to a customer in the global restaurant. Only one of the many possible CRP franchise seating arrangements for the given values of the other random variables is shown.

Again, the first step is to remove the affected customers from the seating arrangement. Note that for the Bigram model, the word following the affected region w_r needs to be removed as well as the identity of its predecessor is unknown and we do not know to which seating arrangement w_r contributes a customer.

The required probabilities can be calculated using equations 2.26 and 2.27. As for the Unigram model, calculating the exact conditional probabilities requires intermediate changes to the seating arrangement, indeed more updates than before because of the Bigram dependencies.[23]

2.3.8 *Hyper parameter inference*

A topic that has received relatively little attention in recent applications of the Unigram and Bigram model is the role played by the hyper parameters. Goldwater (2007) and Goldwater et al. (2009) investigated

23 This can induce noticeable overhead and I found, for any but the tiniest corpora that consist of one or two short utterances, no noticeable difference between samplers that implemented the intermediate update and those that didn't. Thus, if speed is a bottleneck ignoring the calls to addCustomer and removeCustomer during calculating the probabilities of the hypothesis is a possible optimization.

function ADDCUSTOMER(w_0, w, h)
  sample $k \propto \begin{cases} n^{h_{w_0}}_{w,k} & \text{if } 1 \leqslant k \leqslant K^{h_{w_0}} \\ \alpha_1 P(w \mid h_0) & \text{if } k = K^{h_{w_0}} + 1 \end{cases}$
  if $k = K^h + 1$ **then** ▷ open new table
    $res = \frac{\alpha_1 \text{ADDCUSTOMER}(w, h_0)}{n^{h_{w_0}}_{\cdot,\cdot} + \alpha_1}$
    $K^{h_{w_0}} = K^{h_{w_0}} - 1$
    $n^{h_{w_0}}_{w,k} = 1$
  else ▷ added to old table
    $res = \frac{k}{n^{h_{w_0}}_{\cdot,\cdot} + \alpha_1}$
    $n^{h_{w_0}}_{w,k} = n^{h_{w_0}}_{w,k} + 1$
  end if
  return res
end function
function REMOVECUSTOMER(w_0, w, h)
  sample $k \propto n^{h_{w_0}}_{w,k}$
  $n^{h^{w_0}}_{w,k} = n^{h^{w_0}}_{w,k} - 1$
  if $n^{h_{w_0}}_{w,k} = 0$ **then** ▷ Table became empty
    $K^{h_{w_0}} = K^{h_{w_0}} - 1$
    return $\frac{\alpha_1 \text{REMOVECUSTOMER}(w, h_0)}{n^{h^{w_0}}_{\cdot,\cdot} + \alpha_1}$
  else
    return $\frac{n^{h_{w_0}}_{w,k}}{n^{h_{w_0}}_{\cdot,\cdot} + \alpha_1}$
  end if
end function

Figure 2.13: ADDCUSTOMER and REMOVECUSTOMER functions for the Chinese Restaurant Franchise of the Bigram model.

how the precise values of the concentration parameters affect segmentation performance through evaluating a range of manually chosen hyper parameter values. While the Unigram model was found to be rather robust across a wide range of hyper parameter values, the Bigram model turned out to depend considerably on the precise value of the α_1 parameter (Goldwater et al., 2009, Fig. 7).

Goldwater et al. (2009) manually picked the set of hyper parameter values that yields the best performance. While this idea of 'parameter tuning' is common in Computational Linguistics and, for practical applications, can be viewed as part of applying a statistical model to a particular data set, from a scientific (as opposed to an engineering) point of view this raises the question whether the hyper parameters have to be 'built into' the model or whether they can be inferred in an unsupervised fashion as well. In particular, manually determining an optimal set of hyper parameter values for different inputs is not practical for evaluating a model on a wide variety of corpora and, more impor-

procedure SAMPLEBOUNDARY(i, h)
  determine affected span $w_l, w_{1.2}, w_1, w_2$, and w_r
  if $B_i = 1$ **then**
    REMOVECUSTOMER(w_l, w_1, h)
    REMOVECUSTOMER(w_1, w_2, h)
  else
    REMOVECUSTOMER($w_l, w_{1.2}, h$)
  end if
  $p_b = P(w_1 \mid w_l, h)$
  ADDCUSTOMER(w_l, w_1, h) ▷ temporary update
  $p_b = p_b \times P(w_2 \mid w_1, h)$
  ADDCUSTOMER(w_1, w_2, h) ▷ temporary update
  $p_b = p_b \times P(w_2, w_r \mid h)$
  REMOVECUSTOMER(w_l, w_1, h) ▷ Undo temporary updates

  REMOVECUSTOMER(w_1, w_2, h)
  $p_{\neg b} = P(w_{1.2} \mid w_l, h)$
  ADDCUSTOMER($w_l, w_{1.2}, h$) ▷ temporary update
  $p_{\neg b} = p_{\neg b} \times P(w_r \mid w_{1.2}, h)$
  REMOVECUSTOMER($w_l, w_{1.2}, h$) ▷ Undo temporary update

  if NEXTDOUBLE $\times (p_b + p_{\neg b}) \leqslant p_b$ **then** ▷ put boundary
    $B_i = 1$
    ADDCUSTOMER(w_l, w_1, h)
    ADDCUSTOMER(w_1, w_2, h)
    ADDCUSTOMER(w_2, w_r, h)
  else ▷ don't put boundary
    $B_i = 0$
    ADDCUSTOMER($w_l, w_{1.2}, h$)
    ADDCUSTOMER($w_{1.2}, w_r, h$)
  end if

end procedure

Figure 2.14: Algorithm to resample word-boundary B_i in the Bigram model given a current CRP Franchise h.

tantly, unattractive for an unsupervised model of language acquisition. Following Johnson and Goldwater (2009) discussion of hyper parameter sampling for their adaptor grammar framework, we implement hyper parameter inference for the Unigram and Bigram model.

We treat the hyper parameters as random variables and, following Teh et al. (2006) and Johnson and Goldwater (2009), put 'vague' Gamma priors on them, adding

$$\alpha_0 \sim \mathsf{Gamma}(0.001, 0.001)$$
$$\alpha_1 \sim \mathsf{Gamma}(0.001, 0.001)$$

to the definition of the word segmentation models. These Gamma distributions are parametrized in terms of a shape α and an inverse scale (also called rate) β, the most common parametrization within Bayesian statistics (Gelman et al., 2014, p. 576, also see Table 2.1). In this context, 'vague' means that the prior distribution has very high variance – in particular, choosing $\alpha = \beta = 0.001$ results in a variance of $\frac{\alpha}{\beta^2} = 1000$. This shows that this prior corresponds to a very weakly held belief and will be easily overwhelmed by even a single observation, making it a useful uninformative prior. For additional discussion of Gamma priors, see MacKay (2003) and Gelman et al. (2014).

The likelihood for the concentration parameter α is equation 2.14 taken as a function of α where can omit the part depending on $\mathbf{L}$ as the probability of the table labels does not depend on α:

$$P(h_0 \mid \alpha_0) \propto \frac{\Gamma(\alpha_0)}{\Gamma(n_{\cdot,\cdot}^{h_0})} \prod_{k=1}^{K^{h_0}} \alpha\Gamma(n_{\cdot,k}^{h_0}) \tag{2.28}$$

$$P(h_w, \ldots, \mid \alpha_1) \propto \sum_{w'} \frac{\Gamma(\alpha_{w'})}{\Gamma(n_{\cdot,\cdot}^{h_{w'}})} \prod_{k=1}^{K^{h_{w'}}} (\alpha\Gamma(n_{\cdot,k}^{h_{w'}}) \tag{2.29}$$

For α_1, the likelihood sums over all word-specific seating arrangements h_w as all corresponding G_w are drawn from the same $DP(\alpha_1, G_0)$.[24] This allows calculation of the posterior distribution

$$P(\alpha \mid h) \propto P(h \mid \alpha)P(\alpha) \tag{2.30}$$

which can be used in several ways to choose hyper parameters. In general, equation 2.30 will be easy to optimize which suggests working with the MAP estimate, an idea that is known as *MAP-II* as we only use the MAP estimate for the hyper parameter, performing full posterior inference for the remaining random variables (Murphy, 2012, p.)[25] Alternatively, one can resample values for the hyper parameters from the respective posteriors at each iteration, as suggested by Johnson and Goldwater (2009). For this, a wide variety of samplers can be used, and I follow their suggestion of using a Slice sampler (Neal, 2003).

It is worth pointing out, however, that the posterior distribution of the concentration parameter tends to be sufficiently peaked so that sampling and directly using the MAP estimate obtained by optimization made no discernible difference in any of the experiments we performed. As MAP-II only involves solving a one-dimensional optimization procedure rather than performing several iterations of an MCMC method such as Slice sampling, it may be preferred in practice.

24 One can also imagine using a separate hyper parameter α_w for each G_w although in preliminary experiments, I found this to work worse, presumably due to the scarcity of observations for most of the word types.

25 If we assume uniform priors on the hyper parameters (meaning that we directly maximize the likelihood, rather than the posterior), this would be the *ML-II* or *Empirical Bayes* estimate of the hyper parameters.

2.4 APPLYING THE MODELS TO DATA

I briefly illustrate how the models just described can be applied to concrete corpora. In addition to providing a concrete illustration of the idea of Monte Carlo approximations, I also point out a hitherto unreported issue of the Bigram model when its hyper parameters are inferred, rather than manually set.

As data, I use the 'Alice' subsection of the commonly used Brent-Bernstein-Ratner corpus Brent (1999); Bernstein-Ratner (1987). For both the Unigram and the Bigram model, I consider the special cases in which the hyper parameters of the Dirichlet Processes are set to the best-performing values reported in Goldwater et al. (2009) (Fix Hyper parameters) and a model in which the hyper parameters are treated as random variables and their values are also inferred (Inferred Hyper parameters). For the base distribution, I consider both the distribution of Figure 2.4 (No Constraint) and Figure 2.5 (Syllabic Constraint). Combining all possibilities, this yields four variants of the Unigram and the Bigram model, respectively. I refer to the Unigram model with fixed parameters and a constrained base distribution as UNI-FH-SC and to the Bigram model with inferred hyper parameters and an unconstrained base distribution as BI-IH-NC.

To generate samples from the posterior over segmentations, I run four chains of the Gibbs sampler for each of the models. Running multiple chains makes possible cross-chain comparisons that can highlight convergence problems (see the discussion in Gelman et al., 2014, chapter 11.4). Each chain is run for 20000 iterations, and simulated annealing from temperature 10 to 1 is used for the first 10000 iterations to facilitate convergence. During the last 10000 iterations, every 10^{th} samples is collected, generating a total of 1000 samples per chain or 4000 per model.

2.4.1 *Using posterior samples*

Each of the individual samples is a segmentation of the entire observed data. Thus, the posterior approximation is built on the basis of 4000 individual segmentations, and, using equation 2.23, we can calculate the posterior probability of a particular segmentation by calculating the relative frequency of this segmentation among the set of samples we collected. The number of possible segmentations of any given text is exponential in the length of the text (as between any two adjacent phonemes, there either could or could not be a boundary), and for all but tiny corpora there is little hope generating the exact same segmentation – matching on all of the $\mathcal{O}(2^n)$ possible boundaries – more than a few times. To address this, one can use the samples to approximate the marginal posterior over segmentations for each individual utterance.

$\hat{P}$(seg)	segmentation	token f-score
0.45	juwɑnt tu si ðəbʊk	0.40
0.41	ju wɑnttu si ðəbʊk	0.40
0.09	ju wɑnttu si ðə bʊk	0.73
0.04	juwɑnt tu si ðə bʊk	0.73
< 0.01	ju wɑnt tu si ðəbʊk	0.73
< 0.005	juwɑnt tusi ðəbʊk	0.00

Table 2.3: Approximate marginal posterior over segmentations for the utterance "you want to see the book" according to the UNI-FH-NC model, calculated from 4000 samples. We also indicate the token f-score of each individual segmentation, showing that high posterior probability and high token f-score need not coincide. The posterior reflects that the model actually prefers analyzing the initial part of the utterance as ju wɑnttu (with 50% probability in total), even though the single most probable segmentation segments this part as juwɑnt tu.

To illustrate this idea, Table 2.3 gives the approximation to the marginal posterior for the first utterance of the data, "you want to see the book" or, in phonemic notation, "ju wɑnt tu si ðə bʊk", according to the UNI-FH-NC model. There are two segmentations that together account for roughly 86% of the posterior belief but there is no single segmentation that has more than or at least 50% posterior probability. We also see that the model is uncertain about how the initial part of the utterance ought to be segmented: either as "youwant to" or as "you wantto" which are, incidentally, both not quite right but, from a linguistic perspective, make sense as undersegmentations. However, all segmentations agree that the word "see" ought to be segmented out, showing that a model can have different certainty with respect to different parts within an utterance. To my knowledge, this kind of qualitative examination of marginal utterance posteriors is not common in current work, arguably because it is often impractical to generate several thousands of samples for large corpora and, in addition, it is infeasible to qualitatively evaluate tenths of thousands of utterances in close detail. Yet, I think it is worth pointing out that the framework of Bayesian modeling offers the possibility of performing this kind of qualitative evaluation, and I will suggest some more ways of examining posterior distributions to quantify uncertainty in the final chapter of this thesis as suggestions for further work.

An idea introduced to word segmentation by Johnson and Goldwater (2009) is using the marginal posterior distributions to create a 'maximum marginal segmentation' that will be evaluated. The idea is to combine the information present in the full posterior into a single segmentation of the entire corpus that can be evaluated as follows: for each individual utterance, determine the marginal MAP segmentation,

i.e. that segmentation of the individual utterance which, across all sampled segmentations for the entire corpus, occurs the most frequently. This may (and, very likely, will) result in a segmentation for the entire corpus that was never generated but, in a clear sense, synthesizes the information present in the full posterior by 'averaging' across different segmentations.

2.4.2 *Evaluation metrics*

The quality of a segmentation according to a given gold standard is commonly quantified through precision, recall and the harmonic mean of the two (f-score) for tokens, boundaries and word types (Brent, 1999). To illustrate, consider the most probable segmentation in Table 2.3, "youwant to see thebook". This segmentation posits 3 boundaries which all co-incide with boundaries of the gold segmentation "you want to see the book". Hence, boundary precision is $\text{bp} = \frac{\#\text{ predicted}}{\#\text{ correct}} = \frac{3}{3} = 100\%$. On the other hand, recall is only $\text{br} = \frac{\#\text{ predicted}}{\#\text{ gold}} = \frac{3}{5} = 60\%$, and boundary f-score is $\frac{2*\text{bp}*\text{br}}{\text{bp}+\text{br}} = 75\%$. On a word token level, the segmentation fares considerably worse. It posited 4 words, only 2 of which are correct ("to" and "see"), and it only identified 2 of the 6 words in the gold segmentation, yielding $\text{tp} = 50\%, \text{tr} = \frac{1}{3}$ and $\text{tf} = 40\%$. In this example, the lexicon scores co-incide as each token occurs exactly once.

The major benefit of these metrics is that they provide an easy and convenient way to evaluate the quality of a segmentation although, obviously, a lot of the information contained in the posterior is not reflected in these numbers, even if they are calculated on the maximum marginal segmentation. They are, however, the best way currently known to compare a large number of models, and we give the scores for the maximum marginal segmentations of the corpus for the different models in Tables 2.4 and 2.5. We also report the mean score calculated over all individual samples (that is, the posterior expectation of each evaluation score), corroborating Goldwater et al. (2009)'s observation that using the maximum marginal segmentation leads to more accurate segmentation. Also, we find – in line with their exploration of the role of the base distribution – that the Unigram model's performance is quite robust to whether or not a constrained base distribution is used. The scores for all settings are virtually identical (with maximum marginal expectation scores tending to be slightly better than the expected scores of the individual samples), even though the manually chosen hyper paramter $\alpha_0 = 20.0$ is considerably different from the inferred value of $\alpha_0 \approx 150$.[26]

26 Not surprisingly, this value yields an expected number of tables for the Chinese Restaurant Process of ≈ 480 which is close to the true number of types in the data (442).

model	method	bp	br	bf	tp	tr	tf	lp	lr	lf
UNI-FH-NC	mms	.92	.74	**.82**	.73	.63	**.67**	.67	.66	.66
	exp	.91	.73	.81	.71	.61	.66	.65	.63	.64
UNI-FH-SC	mms	.94	.69	.79	.72	.59	.65	.65	.68	.66
	exp	.94	.69	.80	.72	.59	.65	.64	.67	.66
UNI-IH-NC	mms	.92	.70	.80	.70	.58	.64	.65	.67	.66
	exp	.91	.70	.79	.69	.58	.63	.61	.63	.62
UNI-IH-SC	mms	.95	.69	.80	.72	.58	.64	.65	.71	**.68**
	exp	.94	.68	.79	.71	.57	.64	.62	.68	.65

Table 2.4: Scores for the different Unigram models on the Alice corpus, with overall best f-scores in bold face. 'mms' is the score calculated on the maximum marginal segmentation and 'exp' the expected score calculated by averaging the scores over the individual samples. By and large, the scores for all the models are virtually identical, indicating the typical under-segmentation behavior of the Unigram model. This is despite the fact that the inferred α_0 is considerably larger than the manually chosen value of 20, indicating that the Unigram model is robust to both choice of base distribution and hyper parameter.

model	method	bp	br	bf	tp	tr	tf	lp	lr	lf
BI-FH-NC	mms	.92	.83	.87	.78	.73	.76	.69	.67	.68
	exp	.88	.82	.85	.74	.71	.72	.60	.60	.60
BI-FH-SC	mms	.94	.81	.87	.81	.73	.77	.70	.72	**.71**
	exp	.93	.79	.85	.78	.70	.74	.65	.70	.67
BI-IH-NC	mms	.68	.93	.79	.53	.66	.58	.47	.45	.46
	exp	.66	.93	.77	.49	.63	.56	.47	.39	.42
BI-IH-SC	mms	.91	.89	**.90**	.83	.81	**.82**	.70	.72	**.71**
	exp	.89	.87	.88	.80	.79	.79	.68	.68	.68

Table 2.5: Scores for the different Bigram models on the Alice corpus, with overall best f-scores in bold face. 'mms' is the score calculated on the maximum marginal segmentation and 'exp' the expected score calculated by averaging the scores over the individual samples. Unlike for the Unigram model, we see more noticeable impacts of hyper parameters and base distribution – in particular, note the dramatic performance drop (and reversal to over segmentation behavior) when hyper parameters are inferred with an unconstrained base distribution. Also, the expected scores differ more noticeably from the mms scores than for the Unigram model, indicating that overall, the posterior over segmentations is less concentrated

For the Bigram model, there is a more interesting picture. BI-IH-NC performs considerably worse than BI-FH-NC and even worse than UNI-FH-NC. In fact, this model is *over segmenting*, breaking common short words such as "you" into phoneme-bigrmas as in the MAP segmentation for the first utterance y u want tu s i ðə bʊk. This contrasts markedly from the performance of BI-FH-NC which outperforms the Unigram models by a fair margin, in line with the generally held view that a Bigram model is preferable to a Unigram model (Goldwater et al., 2009). While it was known that the Bigram model depends heavily on the values of the concentration parameters (Goldwater, 2007; Goldwater et al., 2009), the fact that hyper parameter inference leads to worse performance indicates that the parameter values supported by the data lead to bad performance. Indeed, the inferred posterior means $\alpha_0 \approx 300$ and $\alpha_1 \approx 7$ are rather different from the manually chosen values $\alpha_0 = 3000, \alpha_1 = 100$.

Interestingly, however, adding the possible word constraint to the Bigram model boosts performance the performance of BI-IH-SC beyond that of the Bigram model with manually specified parameters and leads to the best performing model overall. This suggests, then, that the Bigram model is indeed preferable to the Unigram model but either needs to use manually specified hyper parameters or needs to employ a more substantive base distribution. From a modeling point of view, I prefer the latter choice as it does not require manual 'parameter tuning'. It might also be preferable from a psychological point of view as it attributes less 'a priori knowledge' to the learner. Also, the use of hyper parameter sampling can lead to a better understanding of the models that are studied; concretely, here I found that unless sufficiently constrained, the Bigram model prefers to under- rather than oversegment and can yield rather bad segmentations.[27]

This concludes the introductory review of the Unigram and Bigram model as originally introduced by Goldwater (2007) and Goldwater et al. (2009). Chapter 3 presents an alternative inference algorithm for these models and they form, in a sense, the basis of the explorations performed in this thesis. I close this chapter with a brief review of the related but slightly different Adaptor Grammar framework that I will use for the experiments in Chapter 4 and 5.

2.5 ADAPTOR GRAMMARS

Adaptor Grammars (Johnson et al., 2007b) make it possible to define certain non-parametric Bayesian models through specifying a context free grammar. It is worth pointing out that the models explored in this thesis could all be expressed in terms of finite states methods such as infinite Hidden Markov Models (Beal et al., 2002). That is, the Adaptor Grammars we write are – in terms of generative capacity – regular

27 A similar finding has been made by Frank (2014) for Goldwater et al. (2011)'s Bayesian model of morphological segmentation.

languages. However, formulating these models as context free grammars is both helpful for reasoning about them and, more importantly, allows us to use Johnson et al. (2007b)'s general purpose inference algorithm for a wide variety of different models.

I define Adaptor Grammars formally, following closely the original description of Johnson et al. (2007b). After this, we will show how an Adaptor Grammar model can be expressed without the use of context free grammar rules.

2.5.1 *Formal definition of Adaptor Grammars*

An adaptor grammar (AG) is an extension of a probabilistic context-free grammar (PCFG). A PCFG is an extension of a context-free grammar (CFG) which defines a probability distribution over the trees generated by a CFG.

A CFG is a 4-tuple $\langle \mathcal{N}, \mathcal{W}, \mathcal{R}, S \rangle$, where $\mathcal{N}$ is a finite set of non-terminal symbols, $\mathcal{W}$ is a finite set of terminal symbols that is disjoint from $\mathcal{N}$, $\mathcal{R}$ is a finite set of context-free rules of the form $X \to \alpha, \alpha \in (\mathcal{N} \cup \mathcal{W})^+$ and $S \in \mathcal{N}$ is a start symbol.[28] I use $\mathcal{R}_A$ to refer to all rules in $\mathcal{R}$ that have the non-terminal symbol A on their left-hand side.

The set of trees $\mathcal{T}_S$ that are generated by a CFG can be defined recursively, using $\mathcal{T}_x$ to stand for the set of all trees rooted in x. For every terminal symbol a, $\mathcal{T}_a$ is the set containing only a single tree with a single node labeled a. For every non-terminal X,

$$\mathcal{T}_X = \bigcup_{X \to A_1, \ldots, A_n \in \mathcal{R}} \textsc{Tree}_X(\mathcal{T}_{A_1}, \ldots, \mathcal{T}_{A_n})$$

Here, $\textsc{Tree}_X(\mathcal{T}_{A_1}, \ldots, \mathcal{T}_{A_n})$ refers to the set of all trees rooted in X that have n immediate children such that the i^{th} child tree is an element of $\mathcal{T}_{A_i}$. The definition of $\mathcal{T}_a$ for all terminal symbols a provides the base case for this recursive definition.

A CFG can be used to derive strings in $\mathcal{W}^+$ as follows. Starting with the start symbol S, perform a sequence of rewrite steps until no non-terminal symbols are left. In each rewrite step, a single non-terminal X is replaced by the right-hand side of any rule in $\mathcal{R}_X$. For example, if the rule $S \to NP\ VP$ is in $\mathcal{R}_S$, we can rewrite S as $NP\ VP$. We then choose rules to rewrite NP and VP until only terminal symbols are left. A derivation can also be expressed as a labeled tree which is rooted in S. Every node in the tree corresponds to one of the non-terminals in the derivation and its children are the elements of the right-hand side of the rule that was used to rewrite it in the derivation.

A PCFG adds to a CFG a finite vector θ of rule probabilities. I index this vector by rules such that $\theta_{X \to \gamma}$ refers to the probability of the rule

28 $(\mathcal{N} \cup \mathcal{W})^+$ refers to the set of all strings that consist of an arbitrary combination of elements of $\mathcal{N}$ and $\mathcal{W}$, excluding the empty string. Following common practice, I do not allow ϵ rules in a CFG.

$X \to \gamma$. A PCFG requires that $\theta_{X\to\gamma} \geqslant 0$ and that for all non-terminal symbols X, $\sum_{r\in\mathscr{R}_X} \theta_r = 1$.

A PCFG defines a distribution G_X over trees for every set $\mathscr{T}_X$ which can be defined as follows.[29]

For terminal symbols $w \in \mathscr{W}$, G_w puts all its mass on the unit tree consisting only of a single node labeled w. For non-terminal symbols $X \in \mathscr{N}$,

$$G_X = \sum_{X\to B_1,\ldots,B_n\in\mathscr{R}_X} \theta_{X\to B_1,\ldots,B_n} \textsc{TreeDist}_A(G_{B_1},\ldots,G_{B_2})$$

with

$$\textsc{TreeDist}_A(G_1,\ldots,G_n)\Big(({}_A\, t_1 \ldots t_n)\Big) = \prod_{i=1}^{n} G_i(t_i)$$

I use bracket-notation $({}_A\, t_1 \ldots t_n)$ for a tree rooted in A with children t_1 to t_n which, themselves, are (possibly unit) trees. In other words, TreeDist "is a distribution over trees where the root is labeled A and each subtree t_i is generated independently from G_i" (Johnson et al., 2007b). Intuitively, the probability of a tree according to a PCFG is simply the product of the rule probabilities used in a derivation of the tree.

When performing inference for rule probabilities, they are commonly modeled as drawn from Dirichlet distributions. This is because, as shown above, the Dirichlet prior is conjugate to the categorical likelihood and PCFGs also define such a likelihood. Hence, we can analytically integrate over the rule probabilities which leads to efficient inference algorithms as discussed in Johnson et al. (2007a).

An AG is derived from a PCFG by selecting a subset $\mathscr{A} \subseteq \mathscr{N}$ of adapted non-terminals $\underline{X}$[30] and defining a new distribution over trees H_X for each symbol X as follows. If $X \in \mathscr{W}$, $H_X = G_X$ is the distribution which puts all its mass on the tree with the single node labeled X. If $X \in \mathscr{N}$, $X \notin \mathscr{A}$, i.e. if X is a non-adapted non-terminal,

$$G_X = \sum_{X\to B_1,\ldots,B_n} \theta_{X\to B_1,\ldots,B_n} \textsc{TreeDist}(H_{B_1},\ldots,H_{B_n})$$

$$H_{C\{X\}} = G_X$$

Thus, for terminal symbols and all non-adapted non-terminals $H_X = G_X$ and is defined as for the PCFG. For adapted non-terminals $\underline{X} \in \mathscr{X}$, $G_{\underline{X}}$ is defined as for non-adapted non-terminals but

$$H_{\underline{X}} = \mathsf{DP}(\alpha, G_{\underline{X}})$$

29 This definition, taken from Johnson et al. (2007b), is different from the more standard way of defining the distribution defined by a PCFG simply as the product of the probabilities of the rules used in a derivation (Johnson, 2007, see, e.g.), but it makes it possible to see Adaptor Grammars as direct extensions for PCFGs.

30 Following convention, I distinguish adapted non-terminals from non-adapted non-terminals by underlining the former.

Intuitively, the distribution over trees rooted in an adapted non-terminal is a draw from a Dirichlet Process whose base distribution is a prior distribution which defines a (possibly infinite) set of trees using the PCFG recursion. Recall that a draw from a DP can be seen as an infinite categorical distribution – here, a distribution that has a parameter θ_T for every $T \in \mathscr{T}_X$. Unlike a PCFG, then, an AG cannot be characterized in terms of a finite vector θ because the number of trees rooted in an adapted non-terminal may be infinite. This makes an AG a non-parametric model with an infinite number of parameters.

For practical implementations, the infinite distributions for the adapted non-terminals are integrated out which, for hierarchical models in which adapted non-terminals are dominated by other adapted non-terminals, gives rise to a Chinese Restaurant Franchise representation. MCMC inference for AGs using this representation is described in Johnson et al. (2007b) and not reviewed here although the algorithm is somewhat similar to the blocked sampling algorithm discussed in Chapter 3. Again it is worth pointing out that even though intuitive explanations of adaptor grammars make use of the Chinese Restaurant Process and the idea of there being a cache for previously generated trees, the actual model is independent of this idea. In fact, Cohen et al. (2010) describes a variational inference algorithm for adaptor grammars that does not make use of the Chinese Restaurant Process.

2.5.2 *AG as model definitions*

I will briefly show how a model described as an AG can be related to the kind of model definition used above, using as example Johnson et al. (2007b)'s AG formulation of the Unigram model as defined in Figure 2.3.

$$\begin{aligned}
\text{Seg} &\rightarrow x && (2.31)\\
\text{Segs} &\rightarrow \text{Seg} && (2.32)\\
\text{Segs} &\rightarrow \text{Seg Segs} && (2.33)\\
\underline{\text{Word}} &\rightarrow \text{Segs} && (2.34)\\
\text{Words} &\rightarrow \underline{\text{Word}} && (2.35)\\
\text{Words} &\rightarrow \underline{\text{Word}}\ \text{Words} && (2.36)\\
& && (2.37)
\end{aligned}$$

This grammar can be related to the model defined in Figure 2.3 as follows. Rules 2.31-2.33 define the same distribution over words that P_{lex} in Figure 2.4, i.e. the Unigram phoneme base distribution. In particular, θ_{Seg}, the vector of rule probabilities for the rules in $\mathscr{R}_{\text{seg}}$ corresponds to Θ; and $\theta_{\text{Segs}} \rightarrow \text{Segs Segs}$ corresponds to Φ, the stopping probability. This illustrates how, in principle, a distribution modeled by context free rules in an adaptor grammar can be re-expressed purely with finite-state means as in Figure 2.4.

Rule 2.34 defines the distribution for trees rooted in the adapted non-terminal $\underline{\text{Word}}$, and $H_{\underline{\text{Word}}}$ corresponds directly to G in Figure 2.3. Finally, the simple unigram Markov Process defined by rules 2.35 and 2.36 captures the generative process of generating $W_i \sim G$ and terminating an utterance when $F_i \sim \text{Bern}$ indicates to stop.

The real power of Adaptor Grammars lies in their ability to define rich hierarchies, as made use of in chapter 4 and chapter 5 by simply 'nesting' adapted non-terminals. For example, by grouping words into 'collocations' (phrases made up of multiple words) and adapting the non-terminals dominating these phrases, one can define a collocation model Johnson and Goldwater (2009) simply by adding the following to the AG for the Unigram model:

$$\text{Collocs} \rightarrow \underline{\text{Colloc}} \tag{2.38}$$

$$\text{Collocs} \rightarrow \underline{\text{Colloc}}\ \text{Collocs} \tag{2.39}$$

$$\underline{\text{Colloc}} \rightarrow \text{Words} \tag{2.40}$$

According to rule 2.40, $H_{\underline{\text{Colloc}}} \sim \text{DP}(\alpha, H_{\text{Words}})$. As Words dominates the adapted non-terminal $\underline{\text{Word}}$ for which $H_{\underline{\text{Word}}} \sim \text{DP}(\alpha, H_{\text{Segs}})$, this yields a hierarchical Dirichlet Process model.

When integrating out the relevant distributions, this results in a Chinese Restaurant Franchise in which table labels can be structured objects themselves and customers in the higher-level restaurant correspond to the relevant parts of a structured object. This is illustrated in Figure 2.15.

2.5.2.1 *Hierarchies in adadptor grammars*

Interestingly, the 'probabilistic hierarchy' of the Chinese Restaurant Franchise is the mirror-image of the phrase-structure hierarchy defined by the CFG. Thus, whereas the $\underline{\text{Word}}$ non-terminal is dominated by the $\underline{\text{Colloc}}$ non-terminal in a tree, the $\underline{\text{Word}}$ restaurant is higher up than the $\underline{\text{Colloc}}$ restaurant in the franchise. This has the effect that the distributions of non-terminals that sit 'very low' in the trees generated by an AG will be estimated from a 'dampened' frequency distribution (Goldwater et al., 2011, see also).

Concretely, imagine another instance of the tree $(\underline{\text{C}}\,(\underline{\text{W}}\,\text{the})(\underline{\text{W}}\,\text{dog}))$ is generated from the seating arrangement in Figure 2.15. The entire tree can be generated directly by seating another customer at the first table, and this will leave all counts in h_w unaffected. Thinking about this in terms of inference rather than generation, this means that even though the structure $(\underline{\text{w}}\,\text{the})$ is observed in the input, this may not influence the estimate of $(\underline{\text{w}}\,\text{the})$ but only the estimate of the larger structure in which it was contained. This tendency of estimating statistics of linguistic structures from the types in which they occur rather than from their token frequency has also been observed for human learners (Thiessen and Saffran, 2007) and arises naturally in the context of hierarchical Bayesian modeling.

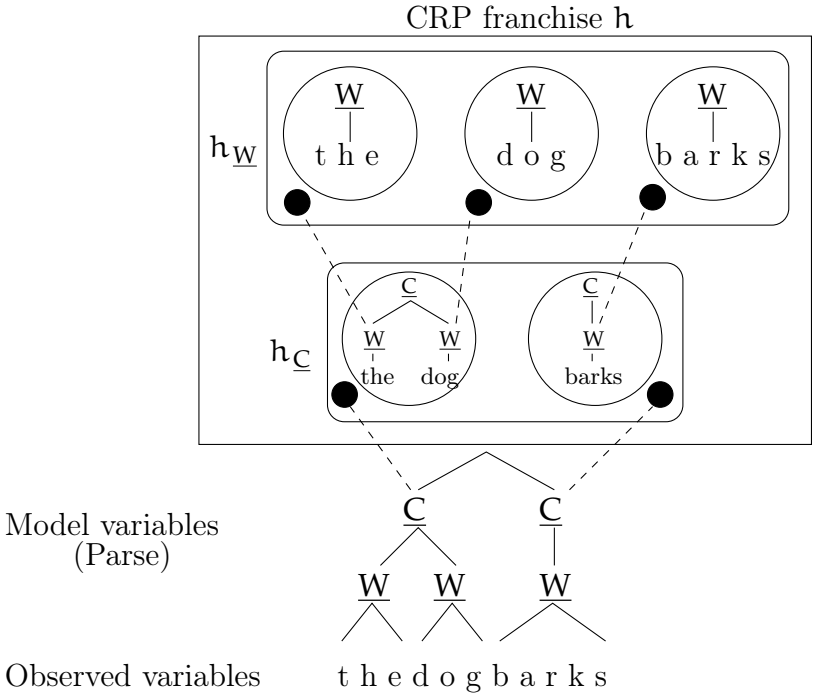


Figure 2.15: Illustration of the Chinese Restaurant Franchise induced by the collocation model, and a seating arrangement for a particular analysis of the input thedogbarks. It is not simply tables that correspond to customers in the higher-level restaurant but actually sub-parts of the structured objects that comprise table labels. This makes it possible to define deep hierarchies in which multiple levels of structure are learned in a non-parametric way jointly.

One can easily add more levels to this hierarchical model by extending the grammar further, and I discuss examples of this in chapters 4 and 5.[31]

It is worth stressing, however, that many models which are defined using adaptor grammars can be re-expressed in a purely finite state framework, as exemplified by the discussion above – the collocation model does not rely on the generative power of context-free grammars. Yet, adaptor grammars make it easy to define a variety of models without having to derive a specific inference algorithm as in, for example, Zhao et al. (unpublished).

Finally, it is worth pointing out a subtle but important difference to the Bigram model. An adaptor grammar model will always consist of a *finite* number of distributions, one per non-terminal, even though any individual distribution may be non-parametric. However, to encode a

31 There can also be multiple 'independent' hierarchies – if one defines a set of connected non-terminals to be a set that contains all and only those non-terminals that, in any tree generated by the grammar, stand in a dominance relationship, there will be a distinct Chinese Restaurant Franchise for any such set of connected non-terminals.

Bigram language model as a PCFG, one needs a dedicated non-terminal symbol for every word in the vocabulary. Paired with the assumption of an infinite vocabulary, as in the Bigram model, this requires an infinite number of non-terminal symbols.[32]

The idea of generalizing PCFGs to infinite numbers of non-terminal symbols has indeed been explored, see Liang et al. (2007), although to my knowledge, no one has combined adaptor grammar's ability to assign non-parametric distributions to non-terminals and the ability to have an infinite number of non-terminals into a single general framework. Thus, as of now, the Bigram model cannot be expressed as an adaptor grammar.

To conclude, adaptor grammars provide an easy-to-use framework that allows one to study a large class of models without having to implement a model-specific inference algorithm. In a sense, adaptor grammars can be seen as the natural language processing analogues of 'general purpose' Bayesian inference tools such as Stan (Stan Development Team, 2014) or BUGS (Lunn et al., 2012). These tools are not suited to handle inference under models that involve highly structured discrete latent structures whereas adaptor grammars make it possible to express a large class of these kinds of models in a generic way.

32 At least in theory – in practice, one can always enumerate the very large but finite number of substrings in any finite amount of input over which we perform inference. Yet, performing inference using such unwieldy grammars with very large numbers of non-terminals is likely to be inefficient and violates the intuition that there could always be a word that has not been observed in the finite input which such a model would have to treat differently from all other words.

3 PARTICLE FILTERS FOR WORD SEGMENTATION

This chapter presents a particle filter algorithm for the word segmentation models of Goldwater et al. (2009). Particle filters (Murphy, 2012) are mathematically well-motivated incremental algorithms that produce a finite approximation to the true posterior of a model, the quality of which increases with larger numbers of particles and converging on the true posterior as the number of particles goes to infinity. This guarantee of asymptotic correctness makes them similar to Markov Chain Monte Carlo methods that are usually used to study Bayesian models and sets them qualitatively apart from most previously proposed online learners for word segmentation that are based on heuristics.

The chapter is structured as follows. First, I discuss the idea of incremental inference, also briefly discussing previous work on online learners for word segmentation. I then derive a strictly incremental particle filter algorithm for the Unigram and Bigram model of word segmentation introduced in Chapter 2. After discussing its performance, I describe an extension to the algorithm that considerably boosts its performance using the idea of rejuvenation at the expense of strict incrementality.

The chapter concludes with a critical discussion of my finding that, under specific circumstances, incremental algorithms yield better segmentations than batch algorithms although they can be shown to perform worse inference (as discussed below). I caution against interpreting these results along the lines of "Less is more" (Newport, 1990; Phillips and Pearl, in press) and that they ought to be viewed as pointing out shortcomings of models rather than suggesting cognitively plausible inference algorithms. In particular, I perform a detailed evaluation and show how both bad and good performance of different incremental algorithms as compared to their batch counterparts can be explained by considering how the segmentations implied by the assumed model change over time, a point systematically addressed in Chapter 4.

3.1 MOTIVATION FOR ONLINE ALGORITHMS

Inference in probabilistic models can usually not be performed analytically and relies on approximate algorithms. A popular choice for algorithms for approximate posterior inference are Markov Chain Monte Carlo algorithms, a general class of algorithms that produce samples from the posterior distribution of interest by making many passes over the entire corpus. Thus, they are batch algorithms that treat the entire input in a single large and iterate over it many times.

Both from a practical and a theoretical stand-point, alternative algorithms that perform inference in an incremental fashion are desirable. For one thing, making many iterations over large amounts of data can be computationally prohibitive and for very large datasets, it may not even be possible to hold the entire input in memory all at once. From a theoretical perspective, incremental algorithms can be seen as addressing the popular (if somewhat misguided) criticism leveled against Bayesian modeling that Markov Chain Monte Carlo inference algorithms are not cognitively plausible due to their batch nature. This can be seen as the main motivation for studying incremental algorithms within cognitive modeling, and I will return to a discussion of the results in the light of this at the end of this chapter.

3.1.1 *Constraints on Online Algorithms*

A standard definition of online learning is that it involves (a) seeing each example only once[1] and (b) making learning decisions on the basis of one example at a time immediately after having seen it, using a finite amount of computation (Bishop, 2006, p. 73). The particle filter presented in this chapter is an example of an algorithm that satisfies these two requirements and consequently constitutes an online learning algorithm according to this definition.

There are more lenient views of online learning which relax these assumptions to some extent. For example, the Online EM algorithms in Liang et al. (2009) perform local updates as required by (b) but iterate over the whole data multiple times, violating (a). Pearl et al. (2010)'s DMCMC algorithm, discussed in the next section, is able to revisit earlier examples in the light of new observations, violating thus both (a) *and* (b) but still performs sequential inference as I will discuss later on. In fact, it can be viewed as a special case of the particle filter with rejuvenation, and we will see that it is only under a thus relaxed notion of online learning that inference for the word segmentation models turns out to work reasonably well.

3.2 PREVIOUS WORK

Online learning algorithms for Bayesian models are discussed within both Statistics and Computational Linguistics but have, to my knowledge, not yet been widely applied to the specific problem of word segmentation. I briefly review previous work on incremental algorithms for word segmentation.

1 This applies to example tokens. There may well be multiple tokens of the same example type.

3.2.1 *Dynamic Programming Maximization*

Brent (1999) and Venkataraman (2001) propose a heuristic online learning algorithm that is a local MAP learner in the sense of Sanborn et al. (2006): it tries to determine the posterior over segmentations for a sequence of unsegmented utterances $\mathbf{u}_{1:n}$ by determining the local maximum a posteriori segmentation of each utterance in the sequence, given only the observations it has observed so far.

Algorithmically, it determines the maximum a posterior segmentation σ_1 for the first unsegmented utterance u_1 according to only the prior constraints of the segmentation model. Then, it updates the segmentation model with the words posited in this segmentation and proceeds to the next utterance. In the terminology of Chapter 2, we can concisely define the algorithm as follows:

$$\begin{aligned} h_0 &= \emptyset \\ \sigma_i \mid h_{i-1} &= \arg\max_{\sigma} P(\sigma \mid u_i, h_{i-1}) \\ h_i &= \textsc{update}(h_{i-1}, \sigma_i) \end{aligned}$$

Here, h_{i-1} refers to the seating arrangement (see Chapter 2) which describes the model state after having processed the first $i-1$ utterances $\mathbf{u}_{1:i-1}$. I write $h_0 = \emptyset$ to indicate that, initially, the seating arrangement contains no customers and the predictive distributions according to which the probability of a segmentation can be calculated only reflect the priors built into the model.

Starting from such an empty seating arrangement, the segmentation for utterance u_i is determined by determining the maximum a posteriori segmentation for u_i given h_{i-1} and using this segmentation to update the segmentation arrangement. I spell out the UPDATE method in more detail below as it also forms part of my algorithm, see Figure 3.5.

For details about how the maximum a posterior segmentation can be determined, see Brent (1999) and Venkataraman (2001). Basically, a variant of the Viterbi algorithm that employs Dynamic Programming treating the spans of the utterance as overlapping sub-problems can be applied in a rather straight-forward fashion, hence the name Dynamic Programming Maximization or DPM for short. However, we will see later that, in fact, being able to use an efficient Dynamic Program comes at the expense of maximizing not $P(\sigma \mid u_i, h_{i-1})$ as defined by the segmentation model but a slightly different yet closely related distribution $Q(\sigma \mid u_i, h_{i-1})$. Thus, this specific local MAP learning algorithm can be viewed as embodying two heuristics – relying on local maximization *and* making use of approximate maximization.

Goldwater (2007) found that this algorithm does perform rather differently to her Gibbs sampler for the same models and, by comparing its results to those of a non-heuristic Gibbs sampler, found that it over-estimates performance of the Unigram model at the expense of under-estimating the relative gain of modeling Bigram dependencies. In

fact, she could show that this heuristic algorithm led Brent (1999) and Venkataraman (2001) to over-estimate the segmentation performance of the Unigram model, and led Venkataraman (2001) to under-estimate the benefit that can be gained from modeling Bigram dependencies.

Similar differences between local MAP learners and batch algorithms have been found for other models by Sanborn et al. (2006). I return to this point and provide an explanation of this divergence in the discussion of the experimental results and will also relate my findings to the idea of 'rational process models' introduced in Sanborn et al. (2006) in the discussion at the end of this chapter.

3.2.2 *Dynamic Programming Sampling*

A slight variant of Local MAP learning is to randomly sample a segmentation for each utterance from the local posterior over segmentations, rather than deterministically choosing the local MAP. Pearl et al. (2010) call this algorithm Dynamic Programming Sampling and it embodies, at a high level, the idea of a particle filter.

The goal of a particle filter is to approximate a sequence of posterior distributions over time. At every time-step, it maintains a set of weighted particles – each particle corresponds to one of the many possible hypotheses over which the posterior is defined, and each weight corresponds to the posterior probability. I discuss particle filters and the idea of sequential inference in much more detail in sections 3.4 and 3.5 but it is worth pointing out that Dynamic Programming Sampling can be viewed as a 1-particle particle filter.

As Dynamic Programming Sampling randomly samples a segmentation rather than always choosing the most probable segmentation, its performance can differ wildly across runs. Not surprisingly, Pearl et al. (2010) as well as Sanborn et al. (2006) found it to, on average, perform worse than local maximization and exhibit very high variance.

3.2.3 *Decayed markov Chain Monte Carlo*

Pearl et al. (2010) present a Decayed Markov Chain Monte Carlo algorithm (Marthi et al., 2002) that can be viewed as a sequential version of the Gibbs Samplers presented in the previous chapter. For each observed utterance, the algorithm is allowed to reconsider any possible boundary position it has encountered so far in light of its current knowledge, but the probability of reconsidering any specific boundary position decreases with its distance from the current utterance. In effect, boundaries in recent utterances are more likely to be reconsidered than boundaries in earlier ones. This property is interpreted by Pearl et al. (2010) as a kind of memory decay.

They examined both how the amount of computation spent on past observations and the choice of decay function (which puts a soft con-

straint on how far back a learner effectively looks) affects inference. Not surprisingly, they found that using more computation leads to better performance and that the Bigram model seems to require a 'larger memory' (in the sense of considering observations further in its past) than the Unigram model.

Their algorithm can be seen as an instance of a particle filter with a single particle that uses *rejuvenation* as discussed below, and we will see that my findings are similar to theirs.[2] Note that this kind of algorithm is not strictly online as, in the words of Pearl et al. (2010), it has "knowledge of 'future' utterances when it samples boundaries further back in the corpus than the current utterance". I return to this point in the discussion.

3.2.4 *Batch inference for word segmentation*

It is worth briefly mentioning the state-of-the-art inference algorithms in Bayesian word segmentation which are *not* incremental but batch algorithms. One can distinguish two closely related classes of algorithms.

On the one hand the original Gibbs samplers of Goldwater (2007) which were reviewed in Chapter 2 sample individual boundary positions to produce samples of entire segmentations. On the other hand, there are blocked samplers that resample the segmentations for entire utterances rather than just individual word boundaries, using a Metropolis-inside-Gibbs step to perform the utterance-wise updates (Johnson et al., 2007b; Mochihashi et al., 2009). The blocked sampling algorithm plays an important role for the particle filter and is discussed in more detail below.

Utterance based samplers have also been applied to more complex models in the adaptor grammar framework, introduced in Johnson et al. (2007b). To date, adaptor grammar models have reported the best scores with roughly 85% word token f-score on the a variety of corpora that are similar to the ones considered here. It is worth emphasizing, however, that this score is attained by a considerably more complex model that learns both syllable structure and hierarchical inter-word dependencies rather than the Unigram and Bigram model I consider here.[3] Thus, superior performance of adaptor grammar models is not a direct result of the specific inference algorithm used but of the choice of a better segmentation model.

2 A subtle difference between their implementation and the particle filter with rejuvenation is that they sample individual boundaries using the Gibbs sampling steps discussed in Chapter 2 whereas my particle filters resample entire utterrances using blocked sampling described below.

3 In theory, the particle filter framework presented in this chapter could also be applied directly to adaptor grammar models although I leave this for future work.

3.3 THE GOLDWATER MODEL FOR WORD SEGMENTATION

The models I study are the Unigram and Bigram models described in Goldwater et al. (2009). As a brief reminder for the reader, I give a brief high-level description of the Unigram model and refer to the review in Chapter 2 or the original descriptions in Goldwater (2007) and Goldwater et al. (2009) for more details.

The model defines a generative process for sequences of words $\boldsymbol{w}_{1:n}$. Each sequence can be interpreted as a particular segmentation of an unsegmented utterance $\mathbf{u}$ that consists of all the segments that make up the words in $\boldsymbol{w}_{1:n}$ concatenated in the right order and with no white spaces. For example, if $\boldsymbol{w} = \langle \text{the}, \text{dog} \rangle$ then $\mathbf{u} = \langle \text{t,h,e,d,o,g} \rangle$.

The first word in this sequence is generated by a distribution over possible words, the so-called base distribution P_{lex} that, in principle, can generate words of an unbounded length (see Figures 2.5 and 2.4). Each following word is either generated by 'reusing' one of the previously generated words, or by making a new draw from the base distribution. This generative process, also known as the (labeled) Chinese Restaurant Process (CRP) and discussed in more detail in Chapter 2, can be described through

$$P(W_1 = w) = P_{\text{lex}}(w)$$

$$P(W_{i+1} = w \mid \boldsymbol{w}_{1:i}) = \frac{C(w, \boldsymbol{w}_{1:i}) + \alpha P_{\text{lex}}(w)}{i + \alpha}$$

Here, $C(w, \boldsymbol{w}_{1:i})$ is the number of times that word w occurs in the sequence of previously generated words $\boldsymbol{w}_{1:i}$ and α is the *concentration parameter* of the CRP. It controls the probability of generating previously unseen words by making a new draw from P_{lex}, with larger values encouraging introduction of novel words and small values resulting in fewer types that repeat more often.

An intuitive understanding of the CRP makes use of a restaurant metaphor: each generated word corresponds to the dish eaten by a customer in a restaurant with an infinite number of tables. Each table serves exactly one dish which all customers sitting at it share, and customers enter the restaurant sequentially and either sit at an already occupied table with probability proportional to the number of people already sitting there or sit at a currently unoccupied table with probability proportional to α. In this case, they 'order' a dish for the table by sampling from the base distribution P_{lex} (see Chapter 2 for more detailed discussion).

Finally, note that the word sequences generated by this process are exchangeable, i.e. every permutation of words is assigned the same probability. Therefore, according to de Finetti's theorem (de Finetti, 1990) there exists some distribution G conditional on which all words in the sequence are distributed independently and identically. Indeed, the word segmentation models are formally defined not in terms of the CRP but in terms of non-parametric distributions over words which are drawn

from Dirichlet Process priors. While the CRP allows us to efficiently perform inference under models which involve infinite distributions which we cannot explicitly represent, for conceptual clarity it is important to keep in mind that the CRP only arises as an effect of collapsing a model which explicitly mentions these distributions. See Chapter 2 for a discussion of the relation between the Chinese Restaurant representation under which inference is performed and the segmentation model which is defined in terms of the Dirichlet Process. I will come back to this point when discussing the hypothesis space of the segmentation model in the next section.

3.4 INCREMENTAL INFERENCE

While this description has focused on the generative aspect of the model, probabilistic models like this are usually not used to generate random sequences of words but to perform inference over the *latent* variables of the model, in this case, the actual words that make up the sequence of segments. Thus, we are interested in the posterior distribution over segmentations $\mathbf{S}_{1:n}$ for a sequence of unsegmented utterances $\mathbf{u}_{1:n}$, $P(\mathbf{S}_{1:n} \mid \mathbf{u}_{1:n})$. Before presenting the particle filter inference algorithm, I elaborate on the idea of incremental inference using a concrete word segmentation example and also clarify the nature of the hypothesis space, a point that was put in slightly misleading ways in Börschinger et al. (2011) and Börschinger and Johnson (2012).

In *incremental inference*, we want to sequentially calculate $P(\mathbf{S}_{1:t} \mid \mathbf{u}_{1:t})$ for all t. $\mathbf{S}_{1:i}$ is a sequence of random variables, and each S_t ranges over the possible segmentations of the first t utterances of the corpus $\mathbf{u}_{1:t}$. I refer to the posterior distribution $P(\mathbf{S}_{1:t} \mid \mathbf{u}_{1:t})$ as the posterior distribution at time t, indicating that it corresponds to the posterior distribution after having observed the first t utterances of the input.

A *batch algorithm* can infer $P(\mathbf{S}_{1:i} \mid \mathbf{u}_{1:i})$ for any i by performing multiple iterations over $\mathbf{u}_{1:i}$, treating all observations as 'known' from the beginning. Thus, in Gibbs sampling each of the many iterations the sampler will perform can condition on all observations except for the small span of unsegmented text that is affected by boundary which is resampled (see Chapter 2).

An incremental inference algorithm, in contrast, calculates a sequence of posteriors, basing inference of the posterior at time t exclusively on the posterior inferred for time $t-1$ and the observation made at time t – there is no need to, so to speak, re-examine previous observations. Incidentally, this idea falls out naturally in a Bayesian framework.

3.4.1 *Incremental Bayesian Inference*

As discussed at the end of Chapter 2, Bayesians view posterior inference as setting a normative standard for how beliefs ought to be updated on

the basis of evidence. In particular, updating a *prior belief* $P(H)$ on the basis of some evidence E to a *posterior belief* $P(H \mid E)$ is called *Bayesian updating* as it relies on Bayes' Theorem:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(H)} \propto P(E \mid H)P(H)$$

Assuming that, conditional on the hypothesis, the observations are identically and independently distributed, one can show that there is no difference between applying Bayesian updating using several observations at once or using the same observations one at a time, as can easily be seen by the following algebraic manipulation:

$$\begin{aligned} P(H \mid E_1, E_2) &\propto P(E_1, E_2 \mid H)P(H) \\ &\propto P(E_1 \mid H)P(E_2 \mid E_1, H)P(H) \\ &\propto P(H \mid E_1)P(E_2 \mid H) \end{aligned}$$

The second line uses the chain rule to $P(E_1, E_2 \mid H)$. In the third line, $P(E_1 \mid H)$ and $P(H)$ are combined into $P(H \mid E_1)$, the posterior distribution over H given only the first observation; and, exploiting the assumption that E_1 and E_2 are conditionally independent given H, $P(E_2 \mid E_1, H)$ simplifies to $P(E_2 \mid H)$.

Thus, we can write $P(H \mid E_1, E_2)$ as the product of the likelihood of H given E_2 and the posterior distribution of H given E_1 – "yesterday's posterior is today's prior".[4] Thus, as Bishop (2006) argues, a "*sequential* approach to learning arises naturally when we adopt a Bayesian viewpoint" (p. 73) and particle filters can be viewed as exploiting the ability to apply Bayes' Theorem recursively to a sequence of observations.

Crucially, this ability assumes that observation are *conditionally independent given the hypothesis*. This assumption, however, is unproblematic even though *unconditional* independence of observations would, of course, render Bayesian updating impossible, as pointed out by De Finetti:

> If I admit the possibility of modifying my probability judgment [=beliefs] in response to observations of frequencies; it means that – by definition – my judgement of the probability of one trial is not independent of the outcomes of the others[.] (De Finetti (1990), quoted according to (Gillies, 2000, p. 75))

As long as the observations are judged to be *exchangeable* – meaning their ordering does not affect their joint probability – De Finetti's theorem guarantees that there is some random variable H (which ranges over distributions over the observations) such that all observations are *conditionally independent given* H. Thus, the conditional independence

4 I haven't been able to determine who ought to be credited with this slogan.

assumption made in the derivation above holds whenever the joint distribution of the observations is exchangeable which is the case for virtually all models ever considered.

With this, I turn to incremental inference under the word segmentation models. Recall from chapter 2 that the CRP induces an exchangeable distribution over sequences of words. Thus we know that there is some underlying distribution according to which the words in the sequence are conditionally independent, and in chapter 2 we saw that this distribution is a draw from a Dirichlet Process:[5]

$$\begin{aligned} G &\sim \mathrm{DP}(\alpha, P_{\mathrm{lex}}) \\ W_i \mid G &\sim G \end{aligned}$$

As discussed in section 2.3.6.1, we can analytically derive the posterior over G if we know the sequence of segmented words. Also, because the words that make up these utterances are conditionally independent given G, so are the individual sequences of words corresponding to each individual utterance. Hence, we can also perform sequential inference through recursive application of Bayes' Theorem:

$$\begin{aligned} G \mid \mathbf{u}_1 &\sim \mathrm{DP}\left(\alpha + |\mathbf{u}_1|, \frac{\sum_{i=1}^{|\mathbf{u}_1|} \delta_{u_{1,i}} + \alpha P_{\mathrm{lex}}}{\alpha + |\mathbf{u}_1|}\right) = \mathrm{DP}(\alpha^1, P^1_{\mathrm{lex}}) \\ G \mid \mathbf{u}_1, \mathbf{u}_2 &\sim P(G \mid \mathbf{u}_1) P(\mathbf{u}_2 \mid G) \\ &= \mathrm{DP}\left(\alpha^1 + |\mathbf{u}_2|, \frac{(\sum_{i=1}^{|\mathbf{u}_2|} \delta_{u_{2,i}}) + \alpha^1 P^1_{\mathrm{lex}}}{\alpha^1 + |\mathbf{u}_1|}\right) \\ &\dots \\ G \mid \mathbf{u}_{1:n} &\sim \mathrm{DP}\left(\alpha^{n-1}, \frac{(\sum_{i=1}^{|\mathbf{u}_n|} \delta_{u_{n,i}}) + \alpha^{n-1} P^{n-1}_{\mathrm{lex}}}{\alpha^{n-1} + |\mathbf{u}_n|}\right) \end{aligned}$$

As we do not observe segmented utterances, we have to sum over all possible latent segmentations which turns the posterior distributions into *mixtures of Dirichlet Processes* with one component per latent segmentation. This sum is usually infeasible to perform as the number of possible segmentations grows exponentially with the size of the corpus, and working directly with infinite objects such as Dirichlet Processes also poses practical problems.

For these reasons, the inference algorithms directly approximates *the marginal posterior over segmentations* with G integrated out and operates in the Chinese Restaurant representation, as discussed at length in Chapter 2. Thus, each state considered by the algorithm is a seating arrangement which provides the sufficient statistics to calculate the posterior expectation of G conditional on a particular segmentation of the

5 For ease of presentation, I limit discussion to the Unigram model which only involves a single G whereas the Bigram model includes an infinite number of H_ws, see Chapter 2.

corpus. As conditioning on a 'seating arrangement' is equivalent to conditioning on a particular G – its posterior expectation – the words and, consequently, the utterances are also conditionally independent given a particular seating arrangement h.

This makes successive application of Bayes' Theorem applicable and forms the basis for the particle filter. After this somewhat formal discussion, I give a concrete illustration of sequential inference for the marginal posterior over segmentations.

3.4.2 *An example for incremental inference*

Figure 3.1 illustrates the sequence of marginal posteriors over segmentations for a toy example comprising the three 'utterances' abcd, defg and cdde in this order. Here, one can exhaustively enumerate all possible $8, 64,$ and 512 segmentations for the first, the first two and all three utterances, respectively, and analytically calculate their posterior probabilities.[6].

Each column corresponds to a different posterior over segmentations, with the first column corresponding to the posterior at $t = 1$ and the third column to the posterior at $t = 3$. I represent segmentations as circles, with the radius of a circle roughly corresponding to the posterior probability of this segmentation. Except for $t = 1$, one cannot depict all possible segmentations in which case I only illustrate some segmentations and summarize the remaining ones as a single big circle which provides the overall number of non-depicted segmentations and their total posterior probability mass.

3.4.2.1 *State-space and hypothesis space*

Figure 3.1 suggests, somewhat misleadingly, that the hypothesis space changes over time. This impression arises because we are working with a collapsed model whose state space comprises the possible analyses of the input rather than the infinite number of distributions over the space of all possible words; particle filters working with a collapsed representation are also called Rao-Blackwellised particle filters (Murphy and Russell, 2001).

Recall from the previous discussion that each segmentation 'hypothesis' defines the sufficient statistics required to recover a posterior over the infinite distribution G. In this sense, even though the number of possible latent segmentations that need to be considered grows over time simply because the size of the corpus grows, the *hypothesis space* according to the segmentation model does not change – it always comprises *all distributions over the space of all possible words.*

6 The probabilities are calculated using a Unigram model with concentration parameter $\alpha = 1.0$ and a Unigram phoneme base distribution with fixed uniform phoneme probabilities and a stopping probability of 0.5 (see Figure 2.4)

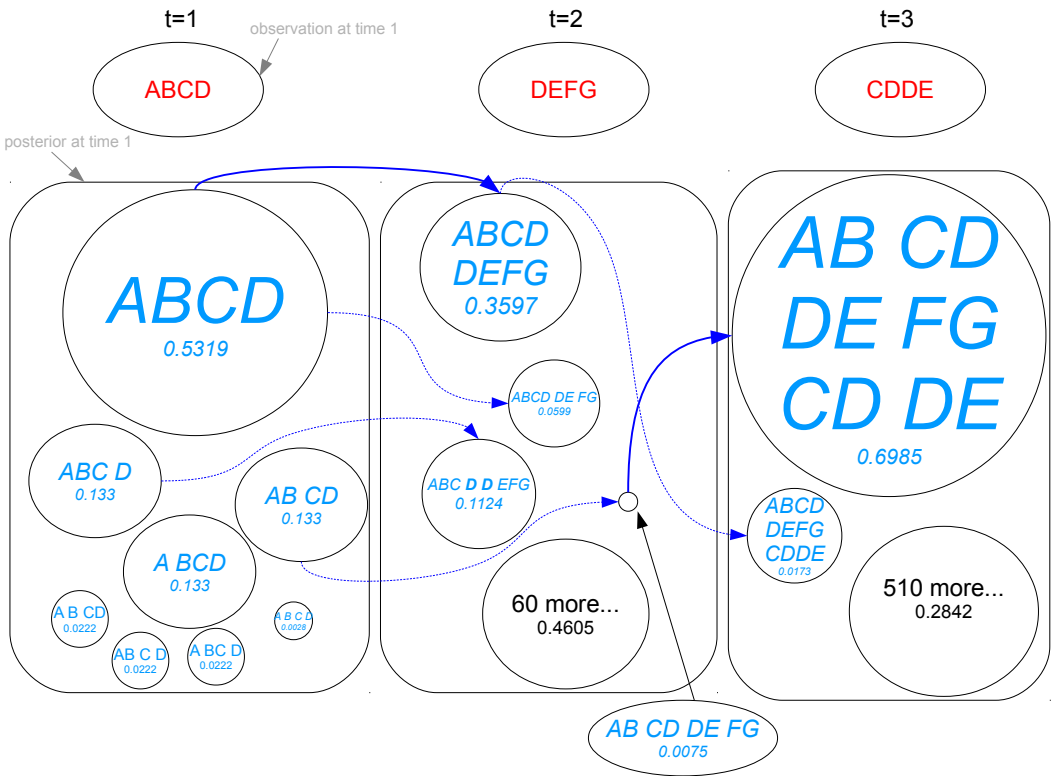


Figure 3.1: Posteriors at time $t = 1, t = 2, t = 3$ for the sequence of utterances ABCD,DEFG and CDDE. Each circle represents a specific hypothesis, i.e. a segmentation of all utterances up to time t. The arrows indicate relations between hypotheses over time with the solid arrow indicating pointing towards the most probable hypothesis according to the next posterior. Not all hypotheses are shown, nor are all possible extensions of a hypothesis indicated through arrows.

What *does change*, however, is the *state space*. At time t, it comprises all possible segmentations of the input of the unsegmented utterances $\mathbf{u}_{1:t}$. This is different from more common applications of sequential inference where, irrespective of the number of measurements on which one conditions, the state space is identical. For example, in object tracking the state space at every time step comprises the co-ordinates and the velocity of the object. In contrast, for us the state space at time step t comprises all possible segmentations of the first t utterances and is different from and bigger than the state space at time $t' < t$ and smaller than the state space at time $t'' > t$.

Second, one can group the segmentations at different time-steps into disjoint 'trajectories': consider a specific latent segmentation h_t at time t that consists of the t segmentations $\mathbf{s}_{1:t}$ of the first t unsegmented utterances $\mathbf{u}_{1:t}$. For example, the most probable such latent segmentation at $t = 2$ in Figure 3.1 comprises the segmentations $s_1 =$ abcd and $s_2 =$ defg. This is only 'compatible' with exactly one of the latent seg-

mentations at time $t = 1$, namely abcd, indicated by the arrow going from the latter to the former.

Thus, while a latent segmentation may be extended in multiple ways (as is illustrated for abcd in Figure 3.1), no two latent segmentations at time t can be extended to yield the same latent segmentation at time $t+1$. Because of this, it can happen that every possible extension of a high-probability segmentation at time t results in a very low segmentation at time $t+1$; and that the only high probability segmentations at time $t+1$ are extensions of low probability segmentations at time t.

Concretely, consider segmentation abcd at time $t = 1$. This is by far the most probable segmentation given only a single utterance which is plausible as there is, so far, no discernible pattern for repetition of smaller elements. There are multiple ways in which this particular segmentation can be 'extended' upon observing the next utterance defg, one for every possible way of segmenting it. Two possible extensions are depicted, and we see that one of the extensions is the most probable segmentation at $t = 2$ whereas another possible extension attains considerably lower posterior probability.[7] However, upon observing the third utterance cdde the most probable extension of what was the most probable segmentation at $t = 1$ only attains 1.7% posterior probability, illustrating a case where the single most probable segmentation at time t' can only be extended to low probability hypotheses at time $t'+1$.

The reverse holds for ⟨ab cd, de fg⟩ which, at time $t = 2$, only has a posterior probability of 0.75%, i.e. less than 1%. Yet, it is the only segmentation that can be extended to ⟨ab cd, de fg, cd de⟩ which, with almost 70% of the posterior probability mass, is by far the single most probable segmentation at time $t = 3$.

As a final point, note that even though the order of observations plays no role when one conditions on all of them because the model is exchangeable, the ordering of the observations does lead to *different sequences* of posterior distributions over segmentations. To illustrate, compare Figure 3.2 to Figure 3.1. Even though at $t = 3$, the probabilities are exactly identical, the distributions at $t = 2$ differ rather dramatically.

I now turn to discussing a particle filter algorithm which performs incremental inference for the marginal posterior over segmentations as just described for Bayesian word segmentation models.

3.5 PARTICLE FILTERING FOR WORD SEGMENTATION

The algorithm is an instance of a particle filter, more precisely, of the Sequential Importance Sampling Resampling (SISR) algorithm (Murphy, 2012, p. 824f). I briefly give the general idea of particle filters before spelling out in detail the algorithm that can be used for word segmentation models.

7 Of course, there are 6 more extensions not shown.

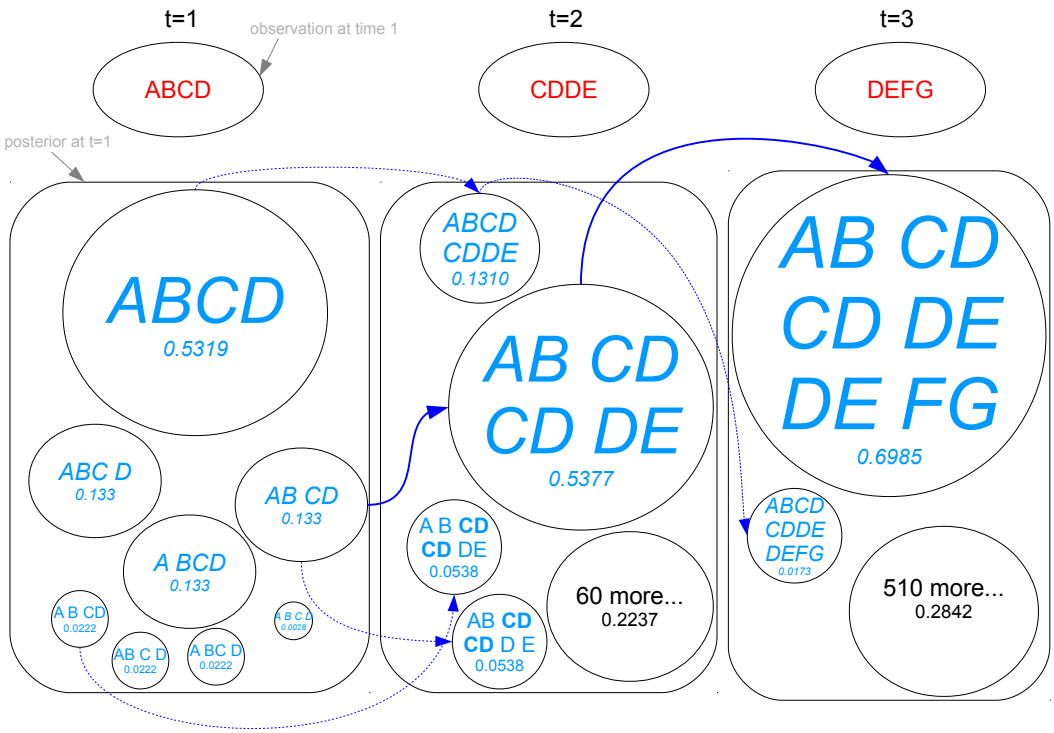


Figure 3.2: Posteriors at time $t = 1, t = 2, t = 3$ for a different permutation of the utterances abcd,defg and cdde. Note that at $t = 3$, the posterior probabilities over segmentations of the entire corpus are identical to those in Figure 3.1, yet at $t = 2$ the posterior distribution over segmentations changes dramatically because the observations on which the posterior at $t = 2$ conditions is different.

The general idea of particle filters is rather simple. One sequentially approximates a target posterior distribution P by N weighted point samples or particles. As pointed out in the previous discussion, here we attempt to approximate the marginal posterior over segmentations of a corpus – thus, each particle corresponds to a *segmentation hypothesis*. In terms of the original segmentation model, each latent segmentation actually provides the sufficient statistics for an entire distribution over infinite distributions over possible words (see also section 2.3.6.1).

Thus, each particle corresponds to a *set of hypotheses* (this idea is similar to that of Steinhardt and Liang, 2014) in the sense discussed for the circles in Figure 3.1. It is assigned a weight that reflects how well this set of hypotheses or, equivalently, the corresponding latent segmentation, is supported by the observations processed so far. At every time-step, the set of particles only represents a sample from the full hypothesis space and the weights represent the marginal posterior

probability of each latent segmentation.[8] Using these weights, we can calculate a Monte Carlo approximation to the marginal posterior probability of every segmentation at each time.

Algorithmically, a particle filter starts from N initial particles which reflect only the prior knowledge encoded in the model. Each individual particle is updated sequentially by randomly sampling among possible future 'extensions' of its corresponding latent segmentation in the sense illustrated in Figure 3.1. After each update, the weight of the particle is adjusted to indicate how well the sampled extension fits the next observation, increasing the weight of particles that provide assign high probability to it and decreasing the weight of those that assign low probability. To specify this in more detail the notion of a state space model is useful.

3.5.1 *State space model*

A state space model defines how a *latent state* Z evolves over time and generates observed measurements U at every time-step. Sequential inference aims to identify the posterior distribution over latent states at each time-step given only the observations up to this time-step, i.e. $P(Z_t \mid Y_{1:t})$.

We can recast word segmentation as such a problem by considering segmentations $\mathbf{s}_{1:t}$ to be the latent states of which we only ever observe the associated unsegmented utterances $\mathbf{u}_{1:t}$. Incremental inference for the posterior over segmentations amounts to inferring the latent sequence of segmentations form the observed unsegmented utterances.

A *state space model* is defined in terms of two conditional distributions. A state-transition probability distribution $P(h_{t+1} \mid h_t)$ governs the transition between the latent state at time t and the latent state $t+1$; in the segmentation model, the latent states at time t are seating arrangements that correspond to the specific segmentation choices $\mathbf{s}_{1:t}$ made for all observed utterances $\mathbf{u}_{1:t}$. And a distribution $P(u_t \mid h_t)$ that generates the observation at time t given the latent state at time t, in this case an unsegmented utterance. The relationship between segmentations and seating arrangements is explained in more detail in Chapter 2 – for ease of presentation, I will usually depict latent states only as segmentations rather than corresponding seating arrangements, as in Figure 3.1.

I define the transition function for latent states in a two step process as follows. Assume we already have generated latent t *segmented utterances* and a corresponding seating arrangement h_t, i.e. a particular latent state at time t. At $t = 0$, this will just be an empty seating arrangement h_0 as no observations have been made.

8 In this sense, the particle filter approximates the exact posterior distribution which, as discussed in section 3.4.1, is a mixture of DPs with one component per latent segmentation as a mixture that only has n components, one for each particle.

To generate a latent state for time $t = 1$, we first *sample a random sequence of words* by running the generative process that underlies the word segmentation model. This amounts to sampling from the posterior predictive distribution $P(W \mid h_0)$ until an utterance boundary is generated (see Chapter 2, equations 2.16 and 2.17).[9] I write this as

$$\boldsymbol{w}_1 \mid h_0 \sim P(\cdot \mid h_0)$$

$\boldsymbol{w}_1$ is the latent segmentation for the first observation. Given the words that make up $\boldsymbol{w}_1$, we can update h_0 accordingly which I write as

$$h_1 \mid \boldsymbol{w}_1 = \textsc{update}(h_0, \boldsymbol{w}_1)$$

For this, we use the function UPDATE which sequentially adds customers to an existing seating arrangement. Thus, we have generated *one particular* latent state at $t = 1$ by first randomly generating a sequence of words and then using this sequence of words to update the seating arrangement h_0. We now repeat the process with the second observation, first generating a random sequence of words from $P(\cdot \mid h_1)$ and updating h_1 to h_2 using this sequence of words. For the general case, an update step can be written as

$$\boldsymbol{w}_{t+1} \mid h_t \sim P(\cdot, h_t) \tag{3.1}$$

$$h_{t+1} \mid \boldsymbol{w}_{t+1} = \textsc{update}(h_t, \boldsymbol{w}_{t+1}) \tag{3.2}$$

Specifying the probability distribution that generates observations from the latent state is trivial as the observation at time $t+1$ is just the concatenation of the words $\boldsymbol{w}_{t+1}$ that were generated in sampling the current latent state h_{t+1}.

$$u_{t+1} \mid \boldsymbol{w}_{t+1} = \textsc{Concat}(\boldsymbol{w}_{t+1})$$

3.5.2 *Naive Particle Filter*

Particle filters perform inference in a state-space model by forward-simulating a finite number of n particles. At every time t, we assume to have access to a set of particles $\mathbf{h}_t^{(1:n)}$ with weights $\boldsymbol{w}_t^{(1:n)}$ that provide a Monte Carlo approximation to the posterior distribution of interest at time t:[10]

$$\hat{P}(H_t = x \mid \mathbf{u}_{1:t}) = \sum_{i=1}^{n} w_t^{(i)} \mathbb{1}[x = h_t^{(i)}]$$

9 Note that I am merely defining the state space model under which inference will be performed rather than the actual inference algorithm. Hence, the 'segmentations' are sampled unconditionally on any observation as they define what the observations are. See the discussion of generative models and inference in section 2.2.

10 I use super-scripts to refer to individual particles. Thus, $h_t^{(i)}$ refers to the i^{th} particle at time t; $\mathbf{h}_t^{(1:n)}$ refers to the set of n particles at time t.

For time $t = 0$, a set of initial particles is usually generated by sampling from the prior distribution and assigning equal weight to every particle. As we perform inference under a collapsed representation, each initial particle corresponds to an empty seating arrangement.

Given a set of weighted particles at time t, we generate a novel set of particles at time $t+1$ through two steps. First, we update each individual particle through performing a 'forward-simulation' step. That is, we sample

$$h_{t+1}^{(i)} \sim P(Z_{t+1} = h_{t+1}^{(i)} \mid Z_{t+1} = h_t^{(i)})$$

as defined by the state space model. Secondly, we update the weights to take into account the novel observation u_{t+1}, essentially multiplying the previous weight $w_t^{(i)}$ and the joint probability of the novel latent state and the observation $P(u_{t+1}, h_{t+1}^{(i)} \mid h_t^{(i)})$.

I won't go into the details of this general algorithm, known as the Bootstrap filter (Gordon et al., 1993), because it exhibits a severe shortcoming for applications such as ours: in particular, $P(u_{t+1} \mid \hat{h}_{t+1})$ is non-zero if and only if the words which were randomly sampled in the update step can be concatenated such that they make up u_{t+1}. This is a consequence of choosing a deterministic mapping using the CONCAT function to generate observations from latent states.

If sampling the new latent state is not constrained by the next observation, in such a case most proposed particles will assign 0 probability to the next observation (as their corresponding segmentation is incompatible with the observed sequence of segments) and, consequently, end up with a new weight of 0. This is illustrated in Figure 3.3 where each dashed line indicates a particle that is incompatible with the next observation. In a high-dimensional space such as ours, one can expect virtually every proposed extension to be assigned a weight of 0, making application of this kind of inference practically impossible.

Luckily, this is not the only way to perform particle filtering. Murphy (2012) argues that it is "much better to actually look at the data u_t when generating a proposal." (p. 827) It can be shown that the optimal distribution according to which one should evolve particles is $P(\hat{h}_i^{(t+1)} \mid \hat{h}_i^{(t)}, u_{t+1})$ (ibid.), rather than using simple forward-simulation from $P(\cdot \mid h_t^{(i)})$ which ignores u_{t+1}.

Following this suggestion, we will sample the segmentation used to update a particle a particle from the conditional distribution over segmentations given the current observation $P(\cdot \mid h, u)$ rather than $P(\cdot \mid h)$. This ensures that all updated particles will be assigned non-zero weights but raises the practical problem of how one can sample from $P(\cdot \mid h, u)$, i.e. the posterior distribution over segmentations for a given unsegmented utterance u and a seating arrangement h.

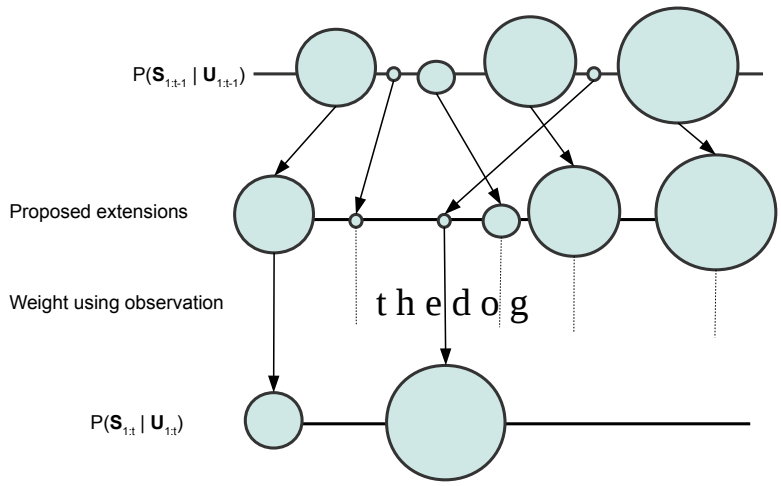


Figure 3.3: Illustration of the Bootstrap particle filter and its problem of generating extensions that are 'incompatible' with the next observation, resulting in particles with weight 0 (indicated as dashed lines that do not result in a novel particle). Refer to text for discussion.

3.5.3 *Data-driven simulation using Sequential Importance Sampling*

Sequential Importance sampling (Murphy, 2012, p.824ff) sidesteps this problem by allowing to update particles using any proposal distribution Q provided its support includes that of the conditional distribution $P(\cdot \mid h, u)$ from which we really want to sample. All that is required is that we be able to calculate the probability of every sampled segmentation according to $P(\cdot \mid h, u)$ which, as we will see, can be done rather easily in this case, even if sampling from P directly is infeasible.

Thus, one can choose any distribution Q from which it is easy to sample to propose the updates and simply correct for the fact that we extended the particles to a different distribution in the particle reweighting step, ensuring that the particle filter still provides an approximation to the intended posterior. The use of a proposal distribution is similar to Metropolis-Hastings sampling as used in, e.g., Johnson et al. (2007b). The main difference is that, rather than rejecting generated samples with some probability, one corrects for the difference between proposal and true target distribution through a weighting scheme.

I briefly review the proposal distribution the algorithm makes use of which was originally introduced in Mochihashi et al. (2009) and is similar to the backward-sampling idea presented in Johnson et al. (2007b).

3.5.3.1 *Blocked sampling from a proposal distribution*

To propose extensions of a particle, we need a distribution that is defined over segmentations of a given utterance u. Ideally, we want this distribution to be as close as possible to the true posterior distribution $P(\cdot \mid h, u)$ defined by the word segmentation model in terms of the posterior predictive distribution induced by the seating arrangement h. For the Unigram model, this is[11]

$$P(\boldsymbol{w}_{1:n} \mid h, u) = P(w_1 \mid h) \left(\prod_{i=2}^{n} (P(w_i \mid h \bigcup \boldsymbol{w}_{1:i-1})(1 - p_s) \right) \times p_s \mathbb{1}[\textsc{Concat}(\boldsymbol{w}_{1:n}) = u]$$

The reason this distribution is hard to sample from is that to calculate this probability exactly, the seating arrangement h needs to sequentially updated after every individual word in the sequence, indicated by writing $h \bigcup \boldsymbol{w}_{1:i-1}$. This adds two dimensions of complexity, the first being that the process of updating a seating arrangement itself is random rather than deterministic and no known closed formula to marginalize over all possible seating arrangements is known.[12] Secondly, the sequential updates make it impossible to marginalize over partial segmentations of u as the exact identity of the words used in a segmentation affects the probability of words that can be used to segment the remainder of u, rendering Dynamic Programming that relies on this kind of overlapping-subproblem structure inapplicable.

There is, however, a straight-forward way of sampling from a slightly different distribution that simply ignores these two issues:

$$Q(\boldsymbol{w}_{1:n} \mid h, u) = \left(\prod_{i=1}^{n} P(w_i \mid h)(1 - p_s) \right) \frac{p_s}{(1 - p_s)} \mathbb{1}[\textsc{Concat}(\boldsymbol{w}_{1:n}) = u]$$

The sole difference to $P(\cdot \mid h, u)$ is that the probability of every word is calculated using h rather than updating h during calculation. This makes it possible to perform efficient sampling using a forward-backward sampling scheme as has been first presented by Mochihashi et al. (2009). As they only give the details for the Bigram model, I present this algorithm for the Unigram model.

Assume an unsegmented utterance $\mathbf{u}$ consisting of n phonemes and a seating arrangement h that defines the posterior predictive distribution $P(w \mid h)$ for every possible word. We build a chart of dimensions $1 \times n$ in which each cell chart[i] contains the marginal probability according to Q of an utterance that consists of the first i segments of $\mathbf{u}$, $\mathbf{u}_{1:i}$ – this is also called the forward-probability of $\mathbf{u}_{1:i}$. This idea is illustrated in Table 3.1 where the n^{th} column contains the sum of the probabilities

11 For simplicity, I assume a fixed stopping probability p_s, rather than integrating this parameter out. Extension along those lines is trivial and automatically dealt with by the re-weighting step.

12 See the discussion in Chapter 2.

function SAMPLE(chart, $\mathbf{u}$)
  $n = |\mathbf{u}|$ ▷ determine length of unsegmented utterance
  $sl \leftarrow n$ ▷ start with full utterance
  $i \leftarrow 1$ ▷ index of generated words
  while $sl > 0$ **do**
    sample $k \propto P(\mathbf{u}_{sl-k+1:sl} \mid h)\text{chart}[sl-k]$ ▷ length of last word
    $w_i \leftarrow \mathbf{u}_{sl-k+1:sl}$ ▷ word spans from $sl-k+1$ to sl
    $i \leftarrow i+1$
    $sl \leftarrow sl-k$ ▷ subtract sampled word from utterance
  end while
  return $\boldsymbol{w} = w_i, w_{i-1}, \ldots, w_1$ ▷ reverse sequence
end function

Figure 3.4: Backward sampling for the Unigram model. At every iteration, a word among all suffixes of $\mathbf{u}_{1:sl}$ (an unsegmented utterance of length sl) is sampled, starting with $sl = n$, i.e. the entire utterance. Once a word has been sampled, the length of this word is subtracted from sl and, until $sl = 0$, the process is repeated. The segmentation is the reversed sequence of words that have been sampled, and sequences will be sampled according to $Q(\boldsymbol{w} \mid h, u)$. The required chart can be built using equation 3.3.

of all possible segmentations of $\mathbf{u}_{1:n}$, the length n prefix of utterance $\mathbf{u}$. We can fill the chart efficiently going from left to right by taking $\text{chart}[0] = 1.0$ by using

$$\text{chart}[k] = \left(\sum_{i=1}^{k-1} \text{chart}[k-i-1](1-p_s)P(\mathbf{u}_{k-i:k} \mid h) \right) + P(c_{1:k} \mid h) \tag{3.3}$$

For the short utterance dog, Table 3.1 illustrates the full chart and how the probability of the individual cells is calculated.

Given such a chart, one can sample a segmentation by making a backwards-pass as follows. Let sl be the length of the entire utterance. Then, sample the *last word* of the segmentation by sampling among all possible suffixes of $\mathbf{u}_{1:sl}$ – the probability that the last word of a segmentation is $\mathbf{u}_{sl-k+1:n}$ is $P(\mathbf{u}_{sl-k+1:n} \mid h)\text{chart}[sl-k]$, i.e. the probability of generating the final word of length k times the marginal probability of the remaining prefix of the utterance. Having sampled a last word, set $sl = sl-k$ and repeat until $sl = 0$, i.e. until there is no unsegmented prefix left. This algorithm is defined in Figure 3.4.

3.6 SEQUENTIAL IMPORTANCE SAMPLING RESAMPLING

With this, everything needed for a *sequential importance resampling particle filter* (Doucet et al., 2000; Murphy, 2012) for word segmentation is in place. I introduce the algorithm and discuss its performance on a

0:ϵ	1:d	2:do	3:dog
1.0	$P(d \mid h)$	$P(do \mid h)+$ $P(o \mid h)(1-p_s)\alpha[1]$	$P(dog \mid h)+$ $P(og \mid h)(1-p_s)\alpha[1]+$ $P(g \mid h)(1-p_s)\alpha[2]$

Table 3.1: Illustration of the chart for the utterance u =dog. For readability, I augment the index with the span of the utterance which is covered by the column – column i spans the initial i segments of u. chart[0] = 1.0 is required for the backward sampling pass in Figure 3.4.

toy example before evaluating its performance on actual child directed speech.

The algorithm is defined in Figure 3.5 and proceeds as follows. At time $i = 0$, initialize n identical empty seating arrangements or particles $\mathbf{h}^{(0)}_{1:N}$. At each time step $i+1$, we update each particle $h^{(i)}_l$ by first sampling a segmentation $s = w_{1:m}$ from $Q(\cdot \mid h^{(i)}_l, u_i)$, i.e. the proposal distribution over segmentations given the previous seating arrangement and the current observation. This is done using the algorithm in Figure 3.4.

A subtle but important detail is that the proposal distribution is only defined over actual sequences of words whereas the hypotheses the particle filter considers are seating arrangements. Rather than modifying the proposal distribution to generate an assignment of words to tables in $h^{(p)}_t$ directly when sampling a segmentation from $P(\cdot \mid u_t, h^{(p)}_t)$, we generate such an assignment 'on-the-fly' upon updating the particle with the proposed segmentation, keeping track of the probability of each individual seating choice made during the update.[13]

This is achieved using the function UPDATE as defined in Figure 3.5 which makes use of the ADDCUSTOMER functions defined in Chapter 2. Note that depending on whether the Unigram or Bigram model is used, a different implementation of UPDATE is used as the way of calculating the probability for a segmentation differs between the two models, as do the ADDCUSTOMER functions used. Except for this, the algorithm is generic and applies to both the Unigram and the Bigram model and can, if the sample and update functions are appropriately changed, applied to any other model as well.

13 For quite technical reasons, modifying the proposal distribution is rather challenging. In particular if the segmentation contains multiple copies of a novel word-type, one needs to ensure that assignments in which these tokens share a table are also generated with non-zero probability, adding considerable complexity. In contrast, my method of generating the seating assignment on the fly side-steps this problem completely.

function PF($n, \mathbf{u}_{1:m}$)
  initialize n empty seating arrangements (=particles) $\mathbf{h}_0^{(1:n)}$
  set all initial weights $\boldsymbol{w}_0^{(1:n)}$ to $\frac{1}{n}$
  for $t = 1 \rightarrow m$ **do**
    for $p = 1 \rightarrow n$ **do**
      sample $\sigma_t \sim Q(\cdot \mid h_{t-1}^{(p)}, u_t)$ ▷ using Figure 3.4
      $h_t^{(p)}, p_{\text{true}} = \text{UPDATE}(h_{t-1}^{(p)}, \sigma_t)$
      $\tilde{w}_t^{(p)} = p_{\text{true}} / Q(\sigma_t \mid h_{t-1}^{(p)}, u_t)$
    end for
    for $p = 1 \rightarrow n$ **do**
      $w_t^{(p)} = \frac{\tilde{w}_t^{(p)}}{\sum_{p'=1}^{n} \tilde{w}_t^{(p')}}$ ▷ normalize weights
    end for
    $\widehat{\text{ESS}} = 1/(\sum_{p=1}^{n} (w_t^{(p)})^2)$
    if $\widehat{\text{ESS}} \leqslant$ threshold **then** ▷ Resampling
      resample all particles according to $\boldsymbol{w}_t^{(1:n)}$
      set all weights to $\frac{1}{n}$
    end if
  end for
end function

function UPDATE(h, σ) ▷ for Unigram model
  $p_{\text{true}} = 1.0$
  for $i = 1 \rightarrow |\sigma| - 1$ **do**
    $p_{\text{true}} = p_{\text{true}} \times P(\text{C} \mid h) \times \text{ADDCUSTOMER}(\sigma_i, h)$
  end for
  $p_{\text{true}} = p_{\text{true}} \times P(\text{S} \mid h) \times \text{ADDCUSTOMER}(\sigma_{|\sigma|}, h)$
  return p_{true}
end function

function UPDATE(h, σ) ▷ for Bigram model
  $p_{\text{true}} = 1.0$
  $w_p = \$$ ▷ preceding word
  for $i = 1 \rightarrow |\sigma|$ **do**
    $p_{\text{true}} = p_{\text{true}} \times \text{ADDCUSTOMER}(w_p, \sigma_i, h)$
    $w_p = \sigma_i$
  end for
  $p_{\text{true}} = p_{\text{true}} \times \text{ADDCUSTOMER}(w_p, \$)$
  return p_{true}
end function

Figure 3.5: Sequential importance sampling resampling particle filter. The algorithm is identical for the Unigram and the Bigram model except for the different UPDATE function that needs to be used. The sole difference in these functions concerns accounting for Bigram dependencies and a slightly different treatment of word-boundaries. The ADDCUSTOMER functions are defined in Figures 2.9 and 2.13.

After having generated the extended particles using UPDATE, the previous weight $w_{i-1}^{(p)}$ of each particle p is updated according to

$$\tilde{w}_i^{(p)} = w_{t-1}^{(p)} \frac{P(u_t \mid h_t^{(p)}) P(h_t^{(p)} \mid h_{t-1}^{(p)})}{Q(\sigma \mid h_{i-1}^{(p)}, u_i)} = w_{t-1}^{(p)} \frac{p_{\text{true}}}{Q(\sigma_t \mid h_{t-1}^{(p)}, u_t)}$$

This is the general weight update formula for a sequential importance sampling particle filter (Murphy, 2012, p. 824) as applied to the specific example of word segmentation. This involves the ratio between the probability of the proposed extension according to the target distribution $P(\cdot \mid h, u)$ and according to the proposal distribution $Q(\cdot \mid h, u)$ – these so called *importance weights* give the algorithm its name.

To calculate $P(h_t^{(p)} \mid h_{t-1}^{(p)})$, i.e. the transition probability of transitioning from latent state $h_{t-1}^{(p)}$ into latent state $h_t^{(p)}$, we use the definition of this distribution in terms of sampling a random sequence of words and adding these words to a seating arrangement (see equations 3.1 and 3.2). Thus, this probability is simply the probability of generating the words that comprise the latent segmentation at time t from $h_{t-1}^{(p)}$ which I write as p_{true} to emphasize that this probability has to be calculated according to the model and not the proposal distribution Q. This probability is automatically calculated by UPDATE and makes up the entire numerator. The reason we can drop the $P(u_t \mid h_t^{(p)})$ factor is that the way in which $h_t^{(p)}$ is generated ensures that it is 1.0: only sequences of words that can be concatenated to yield u_t are assigned non-zero probability by the proposal distribution, and for every such sequence $P(u_t \mid h_t^{(p)}) = 1.0$.

The denominator – the marginal probability of the new observation according to the proposal distribution – can be calculated for any $\sigma = w_{1:n}$ using

$$Q(\sigma \mid h, u) = \frac{Q(\sigma, u \mid h)}{Q(u \mid h)} \tag{3.4}$$

$$= \frac{Q(\sigma \mid h)}{Q(u \mid h)} \tag{3.5}$$

$$Q(\sigma \mid h) = \begin{cases} P(\sigma_1 \mid h)(\prod_{i=2}^{|\sigma|}(1-p_s)P(\sigma_i \mid h))p_s \\ P(\sigma_1 \mid \$, h)\left(\prod_{i=2}^{|\sigma|} P(\sigma_i \mid \sigma_{i-1}, h)\right) P(\$ \mid \sigma_{|\sigma|}, h) \end{cases} \tag{3.6}$$

$$Q(u \mid h) = \text{chart}[n]$$

By construction, chart[n] contains the marginal probability of the utterance, that is, $Q(u \mid h)$. For equation 3.6, one needs to distinguish the Unigram (upper) and Bigram (lower) case.[14] We can simplify $Q(\sigma, u \mid h)$ to $Q(\sigma \mid h)$ because $Q(u \mid \sigma, h) = 1$ for every segmentation that is compatible with u and 0 for every other.

14 For the Unigram model, we need to multiply in the probability of generating the utterance boundary, hence $Q(u \mid h) = \alpha[n]p_s$.

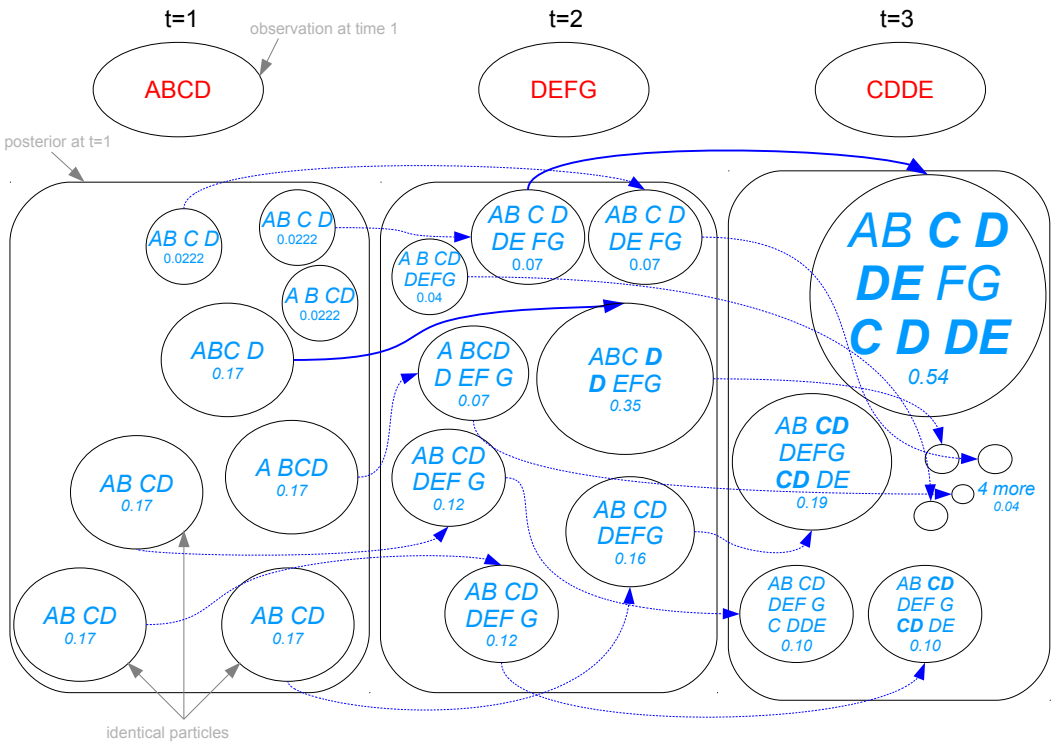


Figure 3.6: Graphical depiction of a simulated run of the sequential importance sampling particle filter.

Finally, we renormalize the particle weights to ensure they characterize a probability distribution, yielding the new set of weights $\boldsymbol{w}_{1:n}^{(i)}$.

For now, I ignore the final resampling step in lines 11 to 14 and will return to it after looking at a worked example of the algorithm described so far which is known simply as *sequential importance sampling*.

3.6.1 *A worked example*

Figure 3.6 depicts in detail the evolution of a set of 8 particles using the sequential importance sampling algorithm on the toy corpus used in Figure 3.1.

Each circle corresponds to a particle, identified by the specific segmentation it corresponds to and its weight.[15] There can be several particles that are identical, as is the case for the three particles at time $t = 1$ that all correspond to the segmentation ab cd. According to this specific set of particles, the approximate posterior probability of ab cd will be $3 \times 0.17 = 0.41$ which is quite different from the true posterior probability of 0.13 (see Figure 3.1).

15 While for ease of visualization, I represent particles through the associated sequence of segmentations, recall that each particle corresponds to a specific seating arrangement rather than a sequence of segmentation choices.

	$P(\langle \text{ab cd}, \text{de fg}, \text{cd de}\rangle \mid \mathbf{u}_{1:3})$	% Particles
True	0.6985	
1,000,000	0.6979	1.55
100,000	0.6967	1.57
10,000	0.7023	1.55

Table 3.2: Posterior probability of the MAP segmentation at $t = 3$ for very large numbers of Particles. Also given is the fraction of particles that correspond to this MAP segmentation which is roughly 1.5% despite these particles accounting for roughly 70% of the posterior probability mass.

This illustrates a general issue of simulations based on random sampling – we may simply be "unlucky" and the most frequent outcome in a set of samples may not coincide with the most probable outcome according to the distribution we try to approximate. Indeed, this particular set of particles does not contain a particle that corresponds to the true MAP hypothesis at $t = 1$, assigning 0 probability rather than 0.53 to the segmentation abcd. Of course, this danger gets smaller as we increase the number of samples and Tables 3.2 and 3.3 show how larger numbers of particles lead to better approximations, getting arbitrarily close to the true posterior in the limit.

Despite its shortcomings, the 8 particle example is useful for understanding the idea of the algorithm. In particular, it allows us to track how individual particles are extended according to the algorithm in Figure 3.5 – the trajectory of each particle is indicated by blue arrows in Figure 3.6.

In this particular run, two of the ab cd particles get extended by choosing def g as the segmentation of the second observation whereas the third one gets extended with defg, resulting in different weights for these particles at time $t = 2$. This shows why it is important to have multiple identical particles as otherwise, only one possible future extensions of every hypothesis could be explored.

We also see that the particle abc d at time $t = 1$ gets extended using d efg, ending up with the highest weight at $t = 2$. This is due to the fact that it is able to 'reuse' the word d, indicated by boldface in the figure, and thus can assign higher probability to the second observation than any of the competing particles who do not 'spot' an already known word in their analysis of the observation. Interestingly, the other two hypotheses that posited a word d at $t = 1$ are updated in a way that does not reuse the word, resulting in lower posterior weights at time $t = 2$. Again, this reflects the random nature of the update – rather than deterministically picking the highest probability extensions, we randomly sample segmentations.

Unsurprisingly, the posterior approximation at $t = 2$ is no better than it was at $t = 1$, assigning the highest weight to a particle that

corresponds to abc d / d efg, a hypothesis that under the true posterior only has a probability of 0.11. It also still assigns 0 probability to the true MAP hypothesis abcd / defg as it could not have generated it, lacking a particle that corresponds to abcd at $t = 1$. This illustrates a general issue with incremental inference algorithms – if a hypothesis is not generated at some point, the algorithm will never be able to consider it even if later evidence speaks strongly in favor of it. Thus, a strictly incremental particle filter can get side-tracked through 'unlucky' choices and never recover from this, a well-known problem for particle filters (see e.g. Murphy, 2012, Fig. 23.6) to which I will return in the discussion.

There are two more points I want to illustrate with this concrete example. The first is that a particle's weight may increase considerably from one time step to the other, as is evident from comparing the particles at $t = 2$ and $t = 3$: a particle corresponding to the segmentation ab c d / de fg with weight 0.07 gets extended to a particle that has a weight of 0.54 at $t = 3$ because its initial segmentation choices – although scarcely supported at $t = 2$ – provide a high-probability analysis of the third observation which reuses the 'words' c, d and de as illustrated by bold-facing. Conversely, a highly weighted particle can end up among the lowest weighted particles, an example of which can also be seen in moving from $t = 2$ to $t = 3$. This up- and down-weighting of particles is how the algorithm can handle the changes in posterior probabilities illustrated in Figure 3.1.

Finally, Table 3.3 compares the most probable 4 segmentations according to the simulation employing 8 particles that I just discussed, a simulation employing 100 particles and the analytically determined posterior distribution. Alongside each hypothesis, I list how many particles correspond to this segmentation.

We see that, unlike the simulation using 8 particles, the one with 100 results in a reasonable (though still far from perfect) approximation to the posterior that correctly identifies the MAP hypothesis at each time. Looking only at the probability of the most probable hypothesis at $t = 3$, Table 3.2 shows that using ever more particles results in a perfect approximation, illustrating the asymptotic correctness of the particle filter – using $1,000,000$ particles correctly identifies the probability of the MAP segmentation up to almost 3 decimal places. Yet, it is striking just how many particles are required to accurately estimate the probability of the MAP hypothesis even in this toy example which only considers 3 observations and where the total number of possible segmentations for the entire corpus is merely 512.

One issue that is apparent from Tables 3.3 and 3.2 is that at $t = 3$, most of the particles correspond to low probability segmentations. Of the 100 particles, only a single one corresponds to the MAP segmentation while 93 particles are used to account for only 17% of the approximation. This is because at $t = 2$, the 'ancestor' of this particle

(corresponding to the segmentation ab cd / de fg) has very low posterior probability and many of the high-probability segmentations at $t = 2$ can only be extended into low probability segmentations at $t = 3$ (see Figure 3.1). Indeed, Table 3.2 shows that the fraction of particles that account for the MAP segmentation at $t = 3$ is consistently below 2% for large numbers of particles.

3.6.2 *Resampling*

This tendency of a small number of particles to attract most mass and, as a result, the weights of all other particles getting closer and closer to 0 is known as the "degeneracy problem, and occurs because we are sampling in a high-dimensional space (in fact, the space is growing over time)" (Murphy, 2012, p. 825), as was illustrated by Figure 3.1.

There are two related reasons why only a few of the particles taking up almost all of the weight is undesirable. First, extending particles with very low weights can be seen as a waste of computation as they contribute very little to the approximation to the posterior; second, having only few particles that represent high probability hypotheses limits the ability to consider alternative extensions of these hypotheses – we'd rather spend the computation wasted on the low probability particles on exploring more possible extensions of the high probability particles.

A straight-forward way of addressing this is to *resample* the n particles according to their current weights. This results in high probability particles having multiple 'descendants' which can be independently extended to explore future possibilites, and in low weight particles being 'weeded out', preventing the algorithm to spend any more resources in exploring how they could be extended. After resampling, the set of particles constitutes an i.i.d. sample from the original approximate posterior distribution and we assign equal weight to every resampled particle. The general idea is illustrated in Figure 3.7.

RESAMPLING USING RESIDUAL SAMPLING There are multiple ways in which one can resample particles. For an experimental evaluation of different strategies, see Douc and Cappé (2005). I follow their suggestion to use *residual resampling*:

First, for every particle $h_i^{(j)}$ with weight $w_i^{(j)}$ we calculate $M^{(j)} = \lfloor n \times w_i^{(h)} \rfloor$. This is the minimal number of descendants that particle $h^{(j)}$ will have and it is simply the (integer part of) the number of times one expects to see particle $h^{(j)}$ if one samples n times from the distribution defined by the current weights. Due to the rounding down, $\sum_{k=1}^{n} M^{(k)}$ may be less than n. This is where *residual sampling* comes in.

	8			100			True	
t	seg	$\hat{P}(\text{seg} \mid u_1)$	n	seg	$\hat{P}(\text{seg} \mid u_1)$	n	seg	$\hat{P}(\text{seg} \mid u_1)$
1	ab cd	0.5	3	abcd	0.50	26	abcd	0.53
	abc d	0.17	1	ab cd	0.18	19	abc d	0.13
	a bcd	0.17	1	abc d	0.13	13	ab cd	0.13
2	abc d /d efg	0.35	1	abcd / defg	0.27	10	abcd / defg	0.36
	ab cd / def g	0.24	2	abcd / def g	0.12	7	abc d / d efg	0.11
	ab cd / defg	0.16	1	abc d / d efg	0.11	7	abc d / defg	0.06
3	ab c d / de fg / c d de	0.54	1	ab cd / de fg / cd de	0.42	1	ab cd / de fg / cd de	0.70
	ab cd / defg / cd de	0.19	1	a b cd / de f g / cd de	0.36	2	a b cd / de fg / cd de	0.05
	ab cd / def g / c dde	0.09	1	abcd / de fg / cd de	0.05	4	ab cd / de f g / cd de	0.05

Table 3.3: Top-3 hypotheses according to the true posterior at each time step and according to two simulated particle filters with 8 and 100 particles, respectively. While the particle filter with 8 particles is already off-track at time $t = 1$, the 100 particles correctly identify the MAP hypothesis at all three times. Yet, even for this toy problem 100 particles do not suffice to closely track the posterior probabilities beyond $t = 1$, underestimating the probability of the MAP hypothesis by a fair bit. We also see that only a single particle corresponds to the MAP hypothesis at time $t = 3$, reflecting the very low posterior probability of the required hypothesis at $t = 2$ (see Figure 2.15), and that 93 particles account for only 17% of the approximation, reflecting the dramatic change that happens upon observing the third observation. This 'inefficient' use of particles can be addressed through resampling, as discussed in the text.

	100			True	
t	seg	$\hat{P}$	n	seg	P
1	abcd	0.59	59	abcd	0.53
	ab cd	0.12	12	abc d	0.13
	a bcd	0.11	11	a bcd	0.13
2	abcd / defg	0.40	40	abcd / defg	0.36
	abc d / d efg	0.11	11	abc d / d efg	0.11
	abcd / de fg	0.09	9	abc d / defg	0.06
3	ab cd / defg / cd de	0.29	29	ab cd / de fg / cd de	0.70
	ab cd / d e fg / cd d e	0.23	23	a b cd / de fg / cd de	0.05
	abcd / de fg / cd de	0.22	22	ab cd / de f g / cd de	0.05

Table 3.4: Top-3 segmentations according to true posterior and to a particle filter with 100 particles and resampling after every observation.

Let $r = n - \sum_{k=1}^{N} M^{(k)}$. We generate the remaining r particles by drawing r times from the distribution defined by the current weights which is equivalent to drawing a vector $\langle \bar{M}^1, \ldots, \bar{M}^N \rangle \sim \mathsf{Mult}(r, \boldsymbol{w}_i^{(1:n)})$, i.e. making a draw from the multinomial distribution over r outcomes defined by the current particle weights (see Table 2.1 for a definition of the Multinomial distribution).

Finally, we let particle $h^{(j)}$ have $M^{(j)} + \bar{M}^{(j)}$ descendants after resampling, resulting in n particles which are likely to include many copies of high probability particles and may completely lack low probability particles.

An additional question concerns when to perform the resampling steps. The approximate effective sample size, defined as

$$\widehat{ESS} = \frac{1}{\sum_{i=1}^{N} (w_{(i)}^{(t)})^2}$$

provides an easy to calculate metric according to which one can decide when to resample. If all particles have the same weight $\widehat{ESS} = N$ which indicates that all of the particles contribute to the posterior approximation. If the weights are heavily skewed, however, $\widehat{ESS}$ will be considerably smaller than N and its magnitude provides a rough estimate of how many of the particles are actually used for the approximation and how many particles have become "useless". Whenever $\widehat{ESS}$ falls below a certain threshold (for example $\frac{N}{2}$), a resampling step is performed and, as all resampled particles have identical weight, after this $\widehat{ESS} = N$.

I experimented with the thresholds N (corresponding to resampling after every observation) and $\frac{N}{2}$, finding little difference between the two thresholds. All experiments that follow have been performed with resampling after every observation.

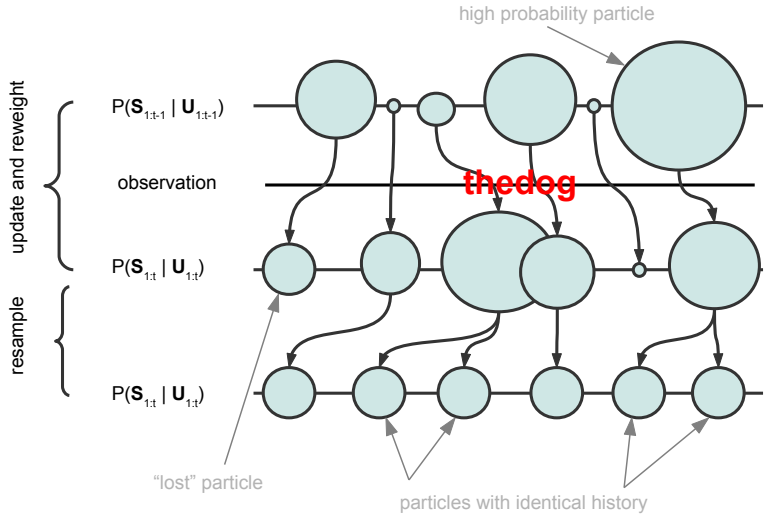


Figure 3.7: Illustration of sequential importance sampling resampling

While this addresses the degeneracy problem, it introduces an orthogonal issue known as *sample impoverishment* (Murphy, 2012, p. 826). This is the issue that low probability hypotheses may be completely lost during a resampling step even though they may be the only hypotheses that could have been extended to high probability hypotheses later on. Basically, the replication of locally high probability particles leads to an *irrecoverable loss of diversity* and this may result in the particle filter getting side-tracked.

For example, consider again the low-probability hypothesis in Figure 3.1 at $t = 2$ which ends up as the highest probability hypothesis at $t = 3$. Having a local probability less than 1%, particles corresponding to this hypothesis are very likely to be weeded out, eliminating the MAP hypothesis completely from the posterior approximations that can be generated at $t = 3$ by extending the current set of particles.

The problem is illustrated by Table 3.4 which provides posterior approximations from a 100 particle particle filter which resamples after every observation. While we see that at $t = 2$, the approximation is reasonable and the number of particles corresponds to the posterior probability of the corresponding hypothesis, at $t = 3$ the posterior approximation is completely off – in fact, the true MAP hypothesis is assigned probability 0 as the locally low probability hypothesis corresponding to ab cd / de fg was 'weeded out' at time $t = 2$, making it impossible to draw the correct conclusion from the observation at $t = 3$. Again, using more particles will address this problem although, as before, the number of particles that is required to perform accurate inference may be impractically high for all but small examples.

3.7 EXPERIMENTAL EVALUATION

After having looked in some detail at the algorithm and its performance on a toy example, I now turn to evaluating the sequential sampling resampling particle filter to actual child directed speech.

There are two questions of interest with respect to the performance of the algorithm. First, we would like to know how faithful the algorithm is to the original model, i.e. whether the approximation to the posterior distribution it infers is close to the true posterior distribution. As it is hard to perform qualitative evaluation of the kind illustrated in Table 3.3 in a setting which comprises hundreds of particles and hundreds of utterances, I evaluate the quality of the posterior approximation by the *expected negative log-probability of the segmentations found by an algorithm.*

The idea behind this is that an accurate posterior approximation ought to concentrate most of its mass on high probability segmentations and thus, result in a better expected negative log-probability. Crucially, we can easily calculate this for every particle (see Chapter 2), and for each particle filter, we can calculate the expectation by taking a weighted sum according to the current particle weights. This is also the metric which Pearl et al. (2010) used to compare inference performance of their incremental DMCMC learner and the batch sampler.

Similarly, as we cannot determine the true posterior I use the batch sampler of Chapter 2 for comparison, considering the expected negative log-probability of the sample segmentations it generates as reference point for the particle filter – ideally, the particle filters should attain an expected negative log-probability that is close to or even better than that of the batch sampler.[16] To get an idea of how important annealing (see Chapter 2) is for the batch Gibbs sampler, I compare a 'vanilla' Gibbs sampler BATCH that does not use annealing and a Gibbs sampler using the simulated annealing schedule discussed in Chapter 2, BATCH-ANNEAL. I initialize each batch sampler randomly by putting a boundary at every possible position with probability 0.5 although preliminary experiments suggested that different initialization schemes (putting no boundaries at all, putting boundaries with at random with higher or lower probability) had no noticeable effects on performance.

Second, we would like to know how well each algorithm performs in terms of the segmentation. As is common, I calculate precision, recall and the harmonic mean of the two, f-measure, for tokens, boundaries and types in the lexicon (see Chapter 2). For the batch samplers, I calculate the maximum marginal segmentation for each individual utterance in the corpus from all the samples collected during a run. For the particle filters, I build the maximum marginal segmentation for each utterance

16 This is essentially the idea Goldwater (2007) used to asses how well a sampler converged onto the true posterior.

by considering the segmentation posited by each particle according to its weight.

Table 3.3 illustrated that particle filters perform better with larger numbers of particles, converging to an exact inference algorithm in the (theoretical) limit where the number of particles goes towards infinity. While there are obvious practical constraints on the number of particles with which one can experiment, I compare a particle filter with a single particle (also called 'Dynamic Programming Sampling' by Pearl et al. (2010)), 100 and 10,000 particles, always running 10 independent simulations for each algorithm. We refer to these algorithms as PF-1, PF-100 and PF-10,000.

For the Gibbs samplers, I run 10 independent simulations for 20,000 iterations, considering the first 10,000 iterations as burn-in. For BATCH-ANNEAL, the temperature was raised from 10 to 1 during the first 10,000 iterations to facilitate convergence on the target distribution, using the same schedule as in Chapter 2. Over the last 10,000 iterations, every 10^{th} sample was collected, for a total of 1000 samples from each simulation.

As a 'baseline' I consider the DPM algorithm which is a heuristic local MAP algorithm (see above).

As test data, I use the Alice section of the Brent-Bernstein-Ratner corpus already used in Chapter 2. While previous work evaluated on the entire Brent-Bernstein-Ratner corpus, as mentioned before this corpus is a concatenation of 9 distinct corpora. Also, "[a] well-known problem with the particle filter is that its performance degrades quickly when the dimension of the state dimension increase" (Gustafsson et al., 2002) which, considering that the state-space grows exponentially in the length of the corpus, suggests that differences between incremental and batch inference are going to become more severe over time. In fact, Börschinger and Johnson (2011), Börschinger and Johnson (2012) and, for different online learners, Pearl et al. (2010) and Phillips and Pearl (in press) evaluate on larger corpora and report large differences between all incremental and batch learners.

In contrast, here I look at natural corpus comprising roughly 1000 utterances directed at a single child rather than concatenating corpora which have been collected across multiple infants (such as the Brent-Bernstein-Ratner corpus) and provide a more detailed analysis. This allows me also to show, for the first time to my knowledge, that incremental inference can, indeed, be 'more efficient' than naive batch inference that does not rely on additional techniques such as simulated annealing.

3.7.1 *Parameter settings*

Both the Unigram and the Bigram model have several parameters which I set to those reported in Goldwater et al. (2009) as yielding the best

performance: $\alpha_0 = 20$ for the Unigram model and $\alpha_0 = 3000, \alpha_1 = 100$ for the Bigram model. For a detailed explanation of the two models and the parameters, see Figure 2.3 on page 35 and Figure 2.10 on page 48. Additionally, I compare a 'restricted' base distribution like the one defined in Figure 2.5 on page 39 with the unrestricted Unigram phoneme distribution of Figure 2.4 on page 37 to see whether this has an impact on incremental inference. Whereas the Unigram phoneme distribution assigns non-zero probability to every possible sequence of phonemes, including obviously non-words such as very long sequences of exclusively consonants, the restricted base distribution assumes a possible word constraint (Norris et al., 1997) that assigns 0 probability to words that lack at least a single vowel.

Thus, I consider two settings for each model, referring to those as UNI-NC (unigram with no possible word constraint), UNI-SC (unigram with possible word constraint), and BI-NC and BI-SC for the Bigram model.

3.7.2 *Unigram Model*

3.7.2.1 *Inference without possible word constraint*

Table 3.5 gives median segmentation scores for the different algorithms, averaged over 10 independent simulations. To get an idea of the variance, Figure 3.9 plots mean token f-score, boundary precision, boundary recall, lexicon precision and lexicon recall for the particle filters and 2 batch Gibbs samplers as box-plots. These plots exclude the DPM algorithm which does not have any variance across runs.

For the UNI-NC setting, we find that the number of particles has a dramatic effect on all metrics, with more particles leading to consistently better performance and, as is evident from Figure 3.11, lower variance.

The batch samplers exhibit very little variance on all metrics, indicating that they reliably converge to a mode of the posterior. Yet, there is a noticeable difference between BATCH and BATCH-ANNEAL, indicating that the two batch samplers are attracted to different modes, with BATCH-ANNEAL getting slightly better token f-score.

In terms of segmentation performance, PF-1 has a median token f-score of only 29% with scores ranging from 5% to 42% whereas PF-100 reaches a median token f-score of 46% with scores ranging from 33 to 60%, worse than either of the batch samplers which reach 66% (BATCH-ANNEAL) and 62% (BATCH), respectively.

PF-10,000's token f-score is close to that of BATCH with 60%, illustrating how using more particles brings performance of the incremental algorithm closer to that of the batch samplers.

The simple DPM algorithm outperforms PF-1 and, perhaps slightly surprising, PF-100 although its 51% token f-score are well below that of PF-10,000 and the batch samplers.

algorithm	tf	bp	br	lp	lr	-logProb×10^3
PF-1	.29	.49	.55	.19	.22	32.35
PF-100	.46	.69	.58	.32	.38	29.18
PF-10,000	.60	.83	.66	.41	.46	27.26
DPM	.51	.86	.48	.29	.38	29.09
BATCH	.62	.92	.64	.60	.63	24.91
BATCH-ANNEAL	**.66**	**.92**	**.74**	**.67**	**.64**	**24.30**

Table 3.5: Segmentation performance for the Unigram model without possible word constraint. "tf", "bp", "br", "lp", and "lr" are short for token f-score, boundary precision, boundary recall, lexicon precision and lexicon recall, respectively.

Looking beyond token f-score, we see that the segmentations inferred by the particle filters and the batch samplers differ qualitatively. Whereas both batch samplers get well above 90% boundary precision with virtually no variance across runs, PF-1 reaches only 49%, PF-100 only 69% and PF-10,000 'only' 83%. Of all the incremental learners, the DPM baseline attains the highest boundary precision with 86%.

PF-1 exhibits higher boundary recall than precision although neither score is particularly good and remains behind those of the other algorithms. Yet, all other algorithms have higher precision than recall, indicating the kind of undersegmentation behavior Goldwater (2007) identified for the Unigram model. This is particularly striking for the DPM learner whose recall is below 50%, accounting for its mediocre token f-score despite the high boundary precision. While we see that using more particles increases boundary precision and recall, PF-10,000's precision is still roughly 10% lower than that of either batch samplers.

Boundary recall explains the difference between BATCH and BATCH-ANNEAL. The former only gets 64% recall to its 90% precision, indicating severe undersegmentation. In contrast, BATCH-ANNEAL gets 74% boundary recall, still considerably less than its precision but noticeably higher than that of BATCH, accounting for the difference in token f-score.

Finally, the lexicon scores indicate that even though PF-10,000 performs very similarly to the batch samplers in terms of token f-score, it both posits more non-word types in its segmentation (lower precision) and identifies fewer of the gold word types (lower precision) than the batch samplers.

With this, I turn to comparing inference rather than segmentation performance according to the expected negative log-probabilities. To get an idea of the variance, Figure 3.8 plots the expected negative log-probability for the segmentations of all 1093 utterances for the difference algorithm which is also given in Table 3.5; here, lower means higher probability and, consequently, better. Again, we see that the batch sam-

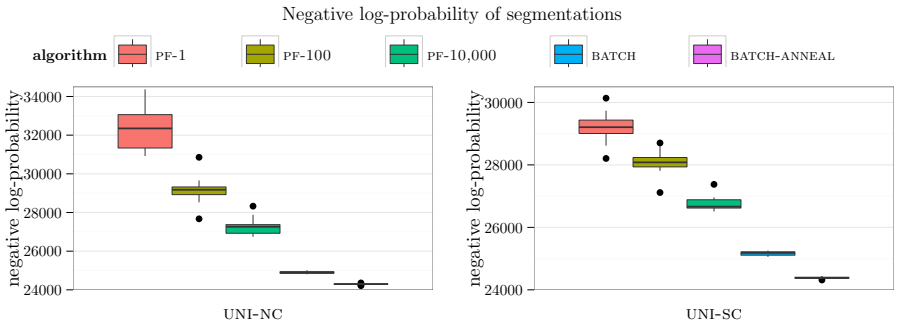


Figure 3.8: Negative log-probabilities for the 1093 segmented utterances according to the particle filters and batch Gibbs samplers with and without annealing; lower means better (see text). We see a clear improvement in going from PF-1 to PF-10000, and then moving towards batch sampling without annealing. Best inference performance is achieved by Gibbs sampling with annealing. Also note that adding the possible word constraint does not affect the ranking but dramatically reduces variance for the 1 particle particle filter.

plers exhibit virtually no variance whereas the particle filters' variance decreases with number of particles.

Not surprisingly, we see that using more particles improves the expected negative log-probability; there is, however, still a noticeable gap between PF-10,000 and the two Gibbs samplers, indicating that offline inference is considerably more efficient. We also see a small if noticeable difference between BATCH and BATCH-ANNEAL, suggesting that even for batch inference, details of the inference algorithm can impact the quality of the posterior approximation.

We also find that even though PF-10,000 is the best performing incremental algorithm, DPM comes in second and outperforms PF-100 in terms of expected negative log-probability, indicating that a rather large number of particles is required to outperform the simple DPM base-line.

3.7.2.2 *Inference with the possible word constraint*

The scores for the UNI-SC setting are given in Table 3.6 and Figure 3.9 and show that adding the possible word constraint changes the picture.

First, there are much smaller differences in segmentation accuracy as measured by token f-score between DPM, PF-1 and PF-100, all of which now achieve between 55% and 58%. In contrast, the constraint has virtually no effect on the performance of the batch samplers whose token f-scores stay roughly the same to UNI-NC. Yet, unlike for UNI-NC PF-10,000 now comes out as the best-performing segmentation algorithm, with 70% token f-score.

Despite the similarities in token f-score the actual segmentations identified by batch and incremental algorithms still differ considerably.

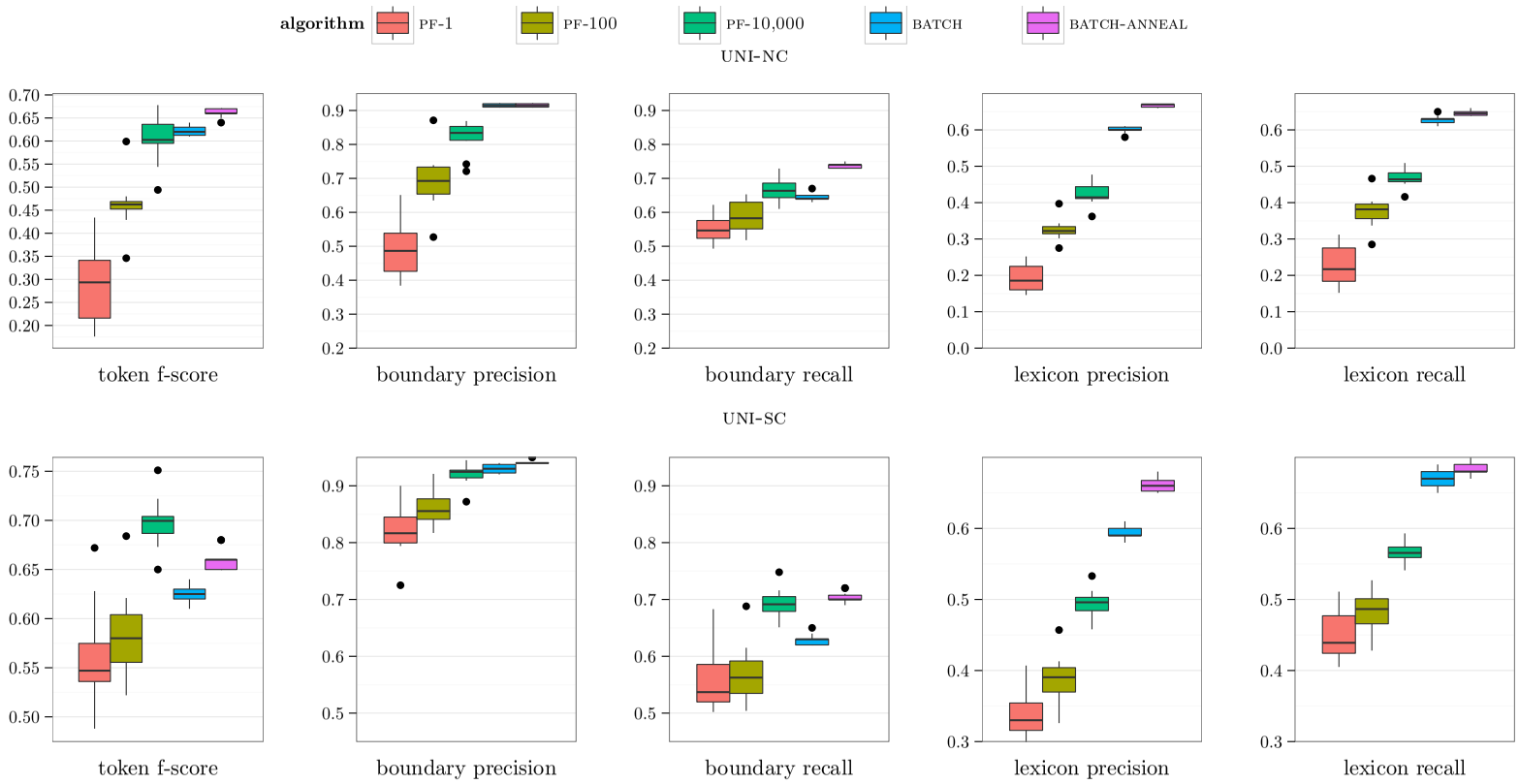


Figure 3.9: Segmentation performance for the Unigram model without possible word constraint (UNI-NC, top) and with possible word constraint (UNI-SC, bottom) on the full Alice corpus for different inference algorithms. The batch Gibbs sampler is run with and without annealing (blue and pink bar, respectively) and the particle filter is run with a single, 100 and 10000 particles. Box-plots summarize 10 independent simulations for each algorithm.

algorithm	tf	bp	br	lp	lr	-logProb×10^3
PF-1	.55	.82	.54	.33	.44	29.21
PF-100	.58	.86	.56	.39	.49	28.08
PF-10,000	**.70**	.92	.69	.50	.57	26.66
DPM	.56	.90	.51	.34	.45	28.86
BATCH	.63	.93	.63	.59	.67	25.18
BATCH-ANNEAL	.66	**.94**	**.70**	**.66**	**.68**	**24.38**

Table 3.6: Segmentation performance for the Unigram model with possible word constraint.

Turning to boundary scores, we first note that compared to UNI-NC, all algorithms get higher precisions. PF-1 and PF-100 now get scores well over 80% (over the low 49 and 69%, respectively). PF-10,000 even reaches 92% boundary precision although the two batch samplers get slightly better still with 93% (BATCH) and 94% (BATCH-ANNEAL). DPM's boundary precision also increases to 90%, outperforming PF-1 and PF-100 but being outperformed by PF-10,000. DPM also gets a minor boost in its already high precision, now reaching 90%, slightly lower than that of PF-10000 and better than that of PF-1 and PF-100.

The increase in boundary precision is not surprising as the possible word constraint rules out many segmentations that posit boundaries at 'impossible' positions (yielding words that lack a syllabic segment). Yet, it is striking that whereas batch learners and DPM were able to identify boundaries with high precision even without such a constraint, the particle filters – in particular those that make use of relatively few particles – benefit tremendously from it.

On the other hand, boundary recall is almost entirely unaffected. BATCH's is still is roughly on par with PF-1 and PF-100, and PF-10000 is virtually identical to BATCH-ANNEAL, with DPM also remaining at roughly 50%.

The lexicon scores show that all the particle filters get a noticeable boost in lexicon precision and recall but still are considerably worse than those of BATCH and BATCH-ANNEAL. This is particularly interesting for PF-10,000 as it attains the overall best token f-score but has considerably lower lexicon precision and recall than either of the batch samplers. Arguably, this is because the incremental learner reliably segments high frequency items.

Turning to inference performance as measured by negative log-probability, we see both from the table and in Figure 3.8 that adding the possible word constraint decreases variance for PF-1 and PF-100 and, overall, makes the gap between the particle filters smaller. It does, however, neither close the gap between PF-10000 and BATCH nor between BATCH and BATCHANNEAL. If anything, the latter gap seems to have widened

algorithm	tf	bp	br	lp	lr	-logProb×10^3
PF-1	.19	.38	.73	.19	.23	33.30
PF-100	.32	.49	.75	.28	.33	30.86
PF-10,000	.44	.58	.77	.35	.39	29.42
DPM	.53	.85	.55	.36	.48	29.88
BATCH	.66	.88	.71	.58	.62	27.47
BATCH-ANNEAL	**.74**	**.89**	**.83**	**.65**	**.63**	**26.55**

Table 3.7: Segmentation scores for the Bigram model without possible word constraint.

slightly, that is, it looks as if with the added constraint the difference between BATCH-ANNEAL and BATCH is more pronounced.

We also see that with the possible word constraint, PF-100 is not only able to outperform DPM on segmentation metrics but also in terms of expected log-probability, indicating that for more constrained models a particle filter with relatively few particles has a chance of outperforming the strong DPM baseline.

Overall, then, we find that the base distribution leaves batch inference rather unaffected but makes a big difference for the incremental algorithms. I return to this point in the discussion.

3.7.3 *Bigram model*

3.7.3.1 *Inference without possible word constraint*

Table 3.7 reports the median scores across the 10 simulations for the algorithms in the BI-NC setting. Figure 3.11 provides the corresponding box-plots to give an idea of the variance. The most striking difference to the UNI-NC setting is the large gap between all of the particle filters and the batch samplers, as well as the rather large difference between BATCH and BATCHANNEAL. In addition, even PF-10,000 is outperformed by DPM, suggesting that for this model, even as many as 10,000 particles do not result in very accurate incremental inference.[17]

With respect to token f-score, BATCH-ANNEAL performs best with a median of 74%. In contrast, BATCH only gets a median of 66%. Thus, annealing makes an even bigger difference for the Bigram than for the Unigram model.

The particle filters now all perform considerably worse, PF-1 with roughly 20%, PF-100 with roughly 32% and PF-10,000 with only 44% token f-score. Thus, while we still see clear improvements in increasing

17 Comparing the respective negative log-probabilities, though, we see that PF-10,000 identifies higher probability-segmentations than DPM, also suggesting the possibility that the posterior distribution is multi-modal and that the particle filter gets attracted by a mode that corresponds to lower accuracy segmentations than DPM.

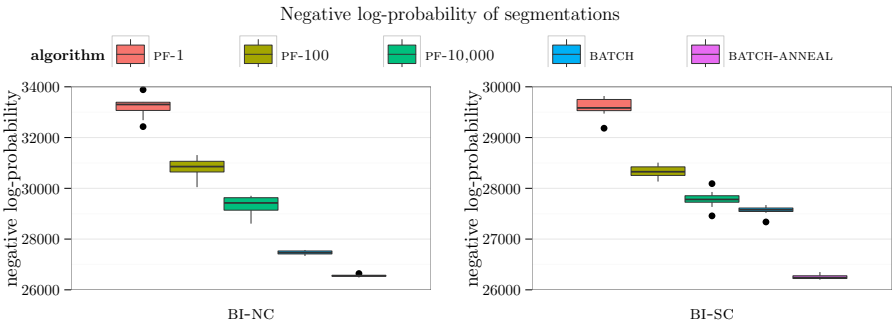


Figure 3.10: Negative log-probabilities for the segmentations of the 1093 utterances found by the particle filters and batch Gibbs samplers with and without annealing. Lower means better. We see a clear improvement in going from PF-1 to PF-1000, and then moving towards batch sampling without annealing. Best inference performance is achieved by Gibbs sampling with annealing.

the numbers of particles, even PF-10000 is outperformed by almost 10% by the DPM algorithm which gets 53%. This is despite the fact that the batch learners actually perform better than under UNI-NC, indicating that the model itself implies higher quality segmentations and that the incremental learners perform inference rather badly.

The boundary scores indicate that the issue is one of over-segmentation for the particle filters, as boundary recall is consistently higher for all of the particle filters than precision. In contrast, the batch samplers still attain high boundary precision and lower recall, indicating under-segmentation that is, however, not as severe as for the Unigram model.

This suggests that the Bigram model generally favors segmentations with more boundaries than the Unigram model, leading to segmentations with higher recalls. Whereas the batch samplers are able to take advantage of this as their precision is high, the incremental learners seem to end up with severely over-segmented solutions which, for PF-10,000, remain below 60% boundary precision but well above 70% recall.

Turning to inference performance in Figure 3.10, we see a similar picture to Figure 3.8 in that increasing the number of particles improves log-probability. Yet, there is still a considerable gap to the Gibbs samplers, and a bigger difference between BATCH and BATCH-ANNEAL than was evident for the Unigram model, consistent with the larger gap between the two samplers in terms of token f-score.

To sum up, then, incremental inference performs much worse for the Bigram model than for the Unigram model, both in terms of identifying accurate word segmentations and high probability solutions.

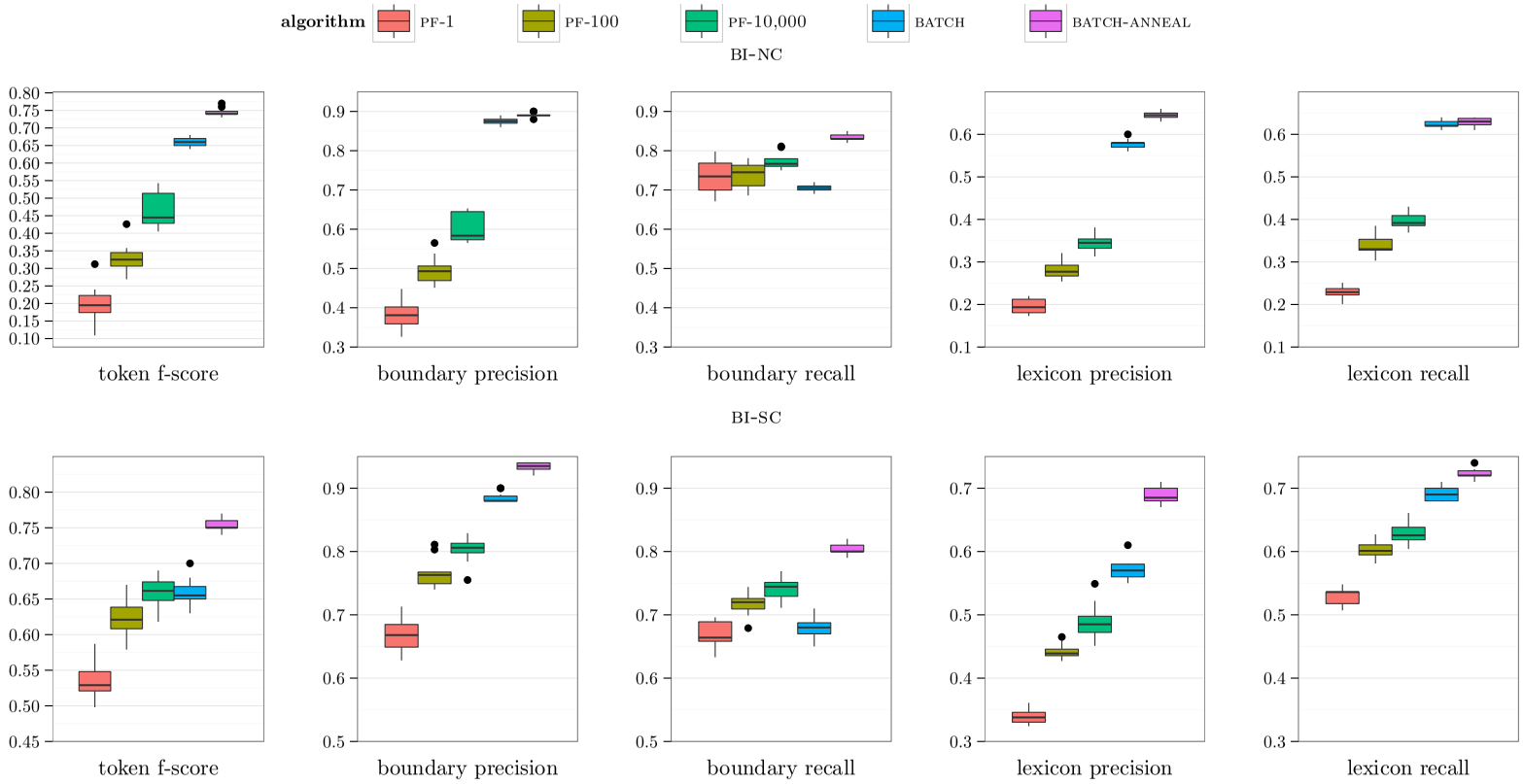


Figure 3.11: Segmentation performance for the Bigram model without possible word constraint (BI-NC, top) and with possible word constraint (BI-SC, bottom) on the full Alice corpus for different inference algorithms. The batch Gibbs sampler is run with and without annealing (blue and pink bar, respectively) and the particle filter is run with a single, 100 and 1000 particles. All scores are averaged across 10 simulations for the particle filters and 10 samples for the batch samplers, with standard errors indicated. See text for discussion.

algorithm	tf	bp	br	lp	lr	-logProb×10^3
PF-1	.53	.67	.66	.34	.54	29.58
PF-100	.62	.76	.72	.44	.60	28.33
PF-10,000	.66	.80	.74	.49	.63	27.78
DPM	.58	.90	.55	.39	.54	29.63
BATCH	.66	.88	.68	.57	.69	27.58
BATCH-ANNEAL	**.75**	**.94**	**.80**	**.69**	**.72**	26.24

Table 3.8: Segmentation performance for the Bigram model with possible word constraint.

3.7.3.2 *Inference with possible word constraint*

Adding the possible word constraint has a similar effect as it had for the Unigram model although it is, overall, less pronounced, as can be seen in Table 3.8.

PF-1's token f-score is now 53%, noticeably worse than PF-100 (62%), and PF-10,000 performs on par with BATCH with 66%. The best algorithm is, still, by far BATCH-ANNEAL with 75%. As for the Unigram model, DPM is largely unaffected by the possible word-constraint and now falls, in terms of token f-score, between PF-1 and PF-100.

Another difference to the Unigram model is the fact that of the batch samplers, only BATCH-ANNEAL gets a noticeable boost in boundary precision, and the boundary precision improvements for the particle filters are much less pronounced than was the case for UNI-SC. Yet, they suffice to have all algorithms but PF-1 clearly undersegment, that is, posit segmentations with higher boundary precision than recall in line with the behavior of the batch samplers.

As for inference performance, the picture is similar to that for the Unigram model although the difference between BATCH and BATCH-ANNEAL is much bigger for BI-SC and, by the same token, that PF-10,000 comes rather close to BATCH. This indicates that annealing is of crucial importance for the batch sampler to perform accurate inference and, in this case.

3.7.4 *Discussion*

The over-arching question raised by the results is why we observe the differences between incremental and batch learners we do and why those differences are larger for the Bigram than for the Unigram model.

Generally speaking, it is not surprising that, despite their asymptotic guarantees, the particle filters do not perform identically to their batch alternatives. The strict incremental character of the algorithm which only allows extensions of segmentations that correspond to one of the finitely many particles at the previous time-step to be considered at all

suggests one obvious reason for the difference: the posterior probability of initial segmentations may change rather dramatically over time, just as we saw that in Figure 3.1 the posterior probability of segmenting the first utterance as abcd drops upon observing the third utterance whereas that of segmenting it as ab cd increases dramatically.

To extend on this reasoning, Figure 3.12 plots the change of token f-scores over time as approximated by running the Gibbs sampler over initial prefixes of the corpus of different sizes, both for UNI-NC and BI-NC, the settings where the particle filters performed worst. The scores for the particle filters are taken from the simulations presented in the previous section whereas the calculated the scores for the batch samplers from simulations on just the corresponding prefix of data.

We see that for BI-NC, there is a sudden increase in token f-score for BATCH-ANNEAL between 200 and 500 utterances, jumping from a mere 55% to roughly 74%. This indicates that very rapidly, the most probable segmentation of many utterances – including utterances observed very early on – changes dramatically to segmentations that result in high token f-scores. And that, conversely, early on the most probable segmentations are segmentations with relatively low token f-scores.

This is reminiscent of the situation depicted in Figure 3.1 where the segmentation of the first two hypotheses changes radically upon observing the third utterance. Here, however, this change is happening over a much longer period and, crucially, after several hundred observations have already been processed. Thus, a strictly incremental learner is unlikely to be able to change its early decisions at so late a point as the required hypotheses will already have been lost due to resampling or simply attained a weight of virtually 0. Again, this is similar to the issue we already observed for the toy example in Table 3.4 although on a considerably larger scale, suggesting that no practically feasible number of particles will overcome it in this case. Thus, in cases like this any incremental learner that can only rely on a finite set of samples at any point in time and never 'revise' earlier analyses will deviate in its conclusions from a batch learner.

Notably, for UNI-NC, there is no jump of comparable magnitude, suggesting a more gradual change of the posterior distribution. Of course, the same principled issue arises – observations made at a very late stage may change the posterior probability of early observations in dramatic ways, making a very low probably hypothesis highly probable and vice versa. Thus, while we also observe deviation between incremental and batch learners in this case it is less pronounced and we see that, at least with 10,000 particles, the particle filter is able to track the segmentation performance of the unannealed batch sampler in terms of token f-score.

Curiously, we also see that the gap between annealed and unannealed batch samplers widens for the Bigram model as a function of input

	UNI-NC		BI-NC	
n	P(seg)	Seg	P(seg)	Seg
10	0.42	yuwanttu siðəbʊk	0.06	yuwan ttu siðəbʊk
20	0.29	yuwant tu siðəbʊk	0.05	yuwan ttu siðəbʊk
50	0.98	yuwant tu si ðəbʊk	0.58	yuwan ttu si ðəbʊk
100	0.96	yuwant tu si ðə bʊk	0.30	yuwant tu si ðə bʊk
200	0.75	yuwant tu si ðə bʊk	0.35	yuwan ttu si ðə bʊk
500	0.99	yuwant tu si ðəbʊk	0.87	yu want tu si ðə bʊk

Table 3.9: Top segmentations according to the marginal posterior over segmentations of the first utterance of the corpus for different amounts of input, according to the run of BATCH-ANNEAL that attained the highest log-probability.

size, indicating that even batch learners can struggle when performing inference over larger amounts of data.

The issue of changes in posterior probabilities of segmentations can be illustrated directly by looking at the most probable segmentation for the first utterance according to the marginal posterior over time. This is shown in Table 3.9, using the samples generated by run of BATCH-ANNEAL that attained the highest log-probability. Note how the Bigram model has very high uncertainty for the first 20 utterances, with the most probable hypothesis getting only a little over 5% of the probability mass. In contrast, the Unigram model already puts 42% on the segmentation "youwantto seethebook" for $n = 10$. Crucially, the Unigram model puts high probability on a 'linguistically meaningful' segmentation early on, and while there are minor changes over time, the overall picture remains surprisingly constant from the beginning.

Not so for the Bigram model which early on favors a somewhat odd segmentation "youwan tto seethebook", being reasonably certain about it at around $n = 50$ but, at $n = 100$, changing its mind briefly to the more accurate "youwant to see the book", only to revert back at $n = 200$ and, finally, at $n = 500$ – the point at which we observed the steep increase – commits to the fully correct segmentation "you want to see the book".

Additional support for my explanation comes from considering actual inference rather than segmentation performance. Figure 3.13 compares the expected negative log-probability of the segmentations identified by the different algorithms for the first 10, 20, 50 and 100 utterances for the BI-NC model. As DPM exhibits no variance, I indicated its log-probability using a horizontal dashed line.

For 10 and 20 utterances, PF-100 and PF-10,000 perform on par or even slightly better than BATCH and BATCH-ANNEAL. Of course, expected negative log-probability is only a coarse proxy for inference performance as a posterior with high uncertainty will attain higher negative

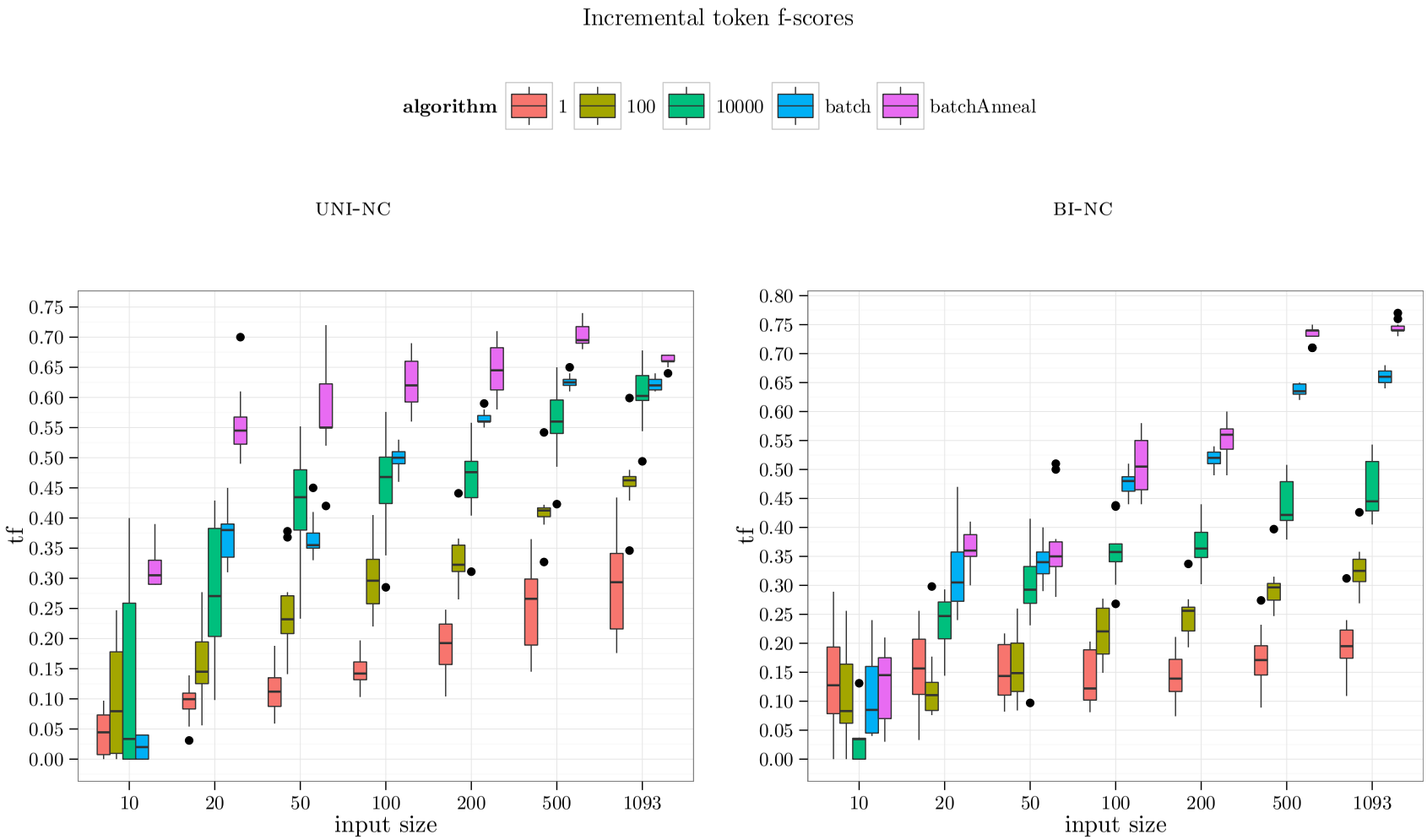


Figure 3.12: Expected token f-scores over time for the different algorithms for BI-NC and UNI-NC over time. See text for discussion.

expected log-probability than a posterior that concentrates more mass on a single high probability hypothesis – a case where more accurate inference would be indicated by a higher than a lower scores. While it is conceivable that the particle filter is 'more efficient' for small corpora as the particles are extended independently whereas the Markov Chain Monte Carlo samplers can suffer from high auto-correlation and poor mixing, we should not over-interpret the fact that PF-100 and PF-10,000 attain 'better' expected negative log-probabilities for 10 and 20 utterances. The relevant point is that as the number of observation increases, the gap between the particle filters and the batch learners grows larger, with the particle filters falling behind more and more. Again, this suggests that the particle filters suffer from the fact that the posterior distributions change quite radically over time in the sense that segmentations that have high posterior probability given only a few initial observations may get very low posterior probability conditional after having observed several more utterances, and vice versa. Thus, even though at $t = 20$ they may have succeeded in providing a good approximation to the posterior up to $t = 20$, their approximation may lack particles that correspond to hypotheses that, at a much later time point, will get high posterior probability.

Incidentally, this also offers an explanation of the fact that the DPM algorithm outperforms all particle filters for BI-NC with respect to token f-score. It is precisely *because* of its reliance on a simple local MAP heuristic which makes it ignore the initially 'misleading' conclusions actually implied by the data – note how its expected log-probability is well below that of PF-100 and PF-10,000 initially, reflecting that its initial choices are not really supported by the data.[18] Despite this, it overtakes PF-100 at around 50 utterances and comes close to PF-10,000 at around 100 utterances, indicating that its initial choices are 'justified' to a certain degree in hindsight by future observations.

This explanation is somewhat different to the explanation Pearl et al. (2010) adduce to account for the difference between incremental and batch learners. They suggested that this difference is due to the fact that the former may simply be unable "to recover from mistakes made early on". Yet, I think it is misleading to simply talk about 'mistakes' here, as the goal of incremental inference is to approximate the posterior at every time-step. And we have just seen that, at least for the first 10 or 20 utterances, inference performance of the particle filters is not dramatically worse (if worse at all) than that of the batch learners. The issue is, in a sense, not that the particle filter fails early on in providing good approximations but that the *true posterior* may, early on, assign a very low probability to segmentations that, given many more observations, will get high probability. This can be called a 'mistake' in

18 Actually, this reflects the fact that DPM relies on an approximate MAP segmentation algorithm which, initially, prefers to not posit any boundaries. If it were determining the actual MAP segmentations, one would expect its early inference performance to be rather well.

hindsight but, from the view of incremental inference, this terminology is misleading.

This both accounts for the general difference between the particle filters and the batch learners, and, in pointing out the sudden change for BI-NC, for the fact that the particle filters perform worse for the Bigram model than for the Unigram model.

Interestingly, it is not just the incremental learners that seem to struggle. For both UNI-NC and BI-NC we find that a batch sampler without annealing is unable to take advantage of the additional utterances to the same extent as BATCH-ANNEAL. This shows, not very surprisingly, that the inference problem becomes harder as a function of the input size and that, somewhat surprisingly, this leads to noticeable differences even between batch learners that process the entire corpus in an offline fashion. Hence, even for batch learners accurate inference is far from trivial for these models, and while this effect is much more pronounced for the Bigram model it is also obvious for the Unigram model. Overall, then, the fact that incremental learners perform worse than their batch counterparts is not too surprising, considering that even the latter have to rely on techniques such as annealing to improve convergence.

Finally, it is worth stressing again that upon adding the substantive possible word constraint, the segmentation performance gap between incremental and batch learners gets much smaller. This is because ruling out segmentations that include 'words' without at least one syllabic segment reduces posterior uncertainty early on and, crucially, prevents particle filters from committing to hypotheses that posit single consonants or short sequences of consonants as words. While a batch learner may draw very similar conclusions from a large set of utterances with and without the possible word constraint, incremental learners depend heavily on the initial choices they make on the basis of only very few observations. By cutting down the number of possible choices there and ruling out hypotheses that will have devastating effects on future segmentations – such as treating highly frequent segments such as ð (the "th" in "the", "that", etc) – the possible word constraint can be essential to their performance.

3.8 PARTICLE FILTER WITH REJUVENATION

The previous discussion suggests that strict incrementality is at the heart of the problem. In fact, there are two related issues that need to be dealt with. First, there is the general issue of *loss of sample diversity* already alluded to above – very quickly, almost all particles will agree exactly on the analyses of all previous observations. While this can be addressed to some extent through the use of large numbers of Particles, for all but very small numbers of observations this is practically impossible, as "all the particles will collapse to a single point within a few iterations" (Murphy, 2012). This is not only a problem in

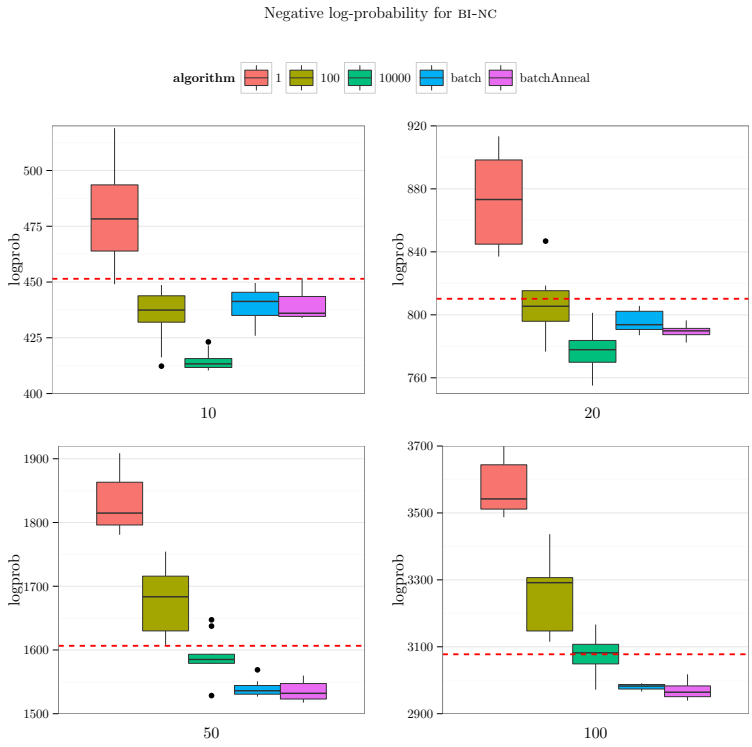


Figure 3.13: Expected negative log-probability of the segmentations identified by the different algorithms for the first 10, 20, 50 and 100 utterances of the data. DPM is represented by a dashed red line. Note that even though PF-100 and PF-10,000 yield rather good expected negative log-probabilities early on, they start falling behind rather early. In contrast, even though DPM initially is considerably worse than PF-100 and PF-10,000, it overtakes them as more observations are considered. See text for discussion.

so far as it makes the approximations at later time points degenerate; it also, to come to the second issue, makes it impossible to revise analyses of earlier observations. In cases where alter observations are strongly informative about earlier segmentations by making much more probable segmentations that were very low probability when considering only the initial observations, this makes it impossible for the particle filter to ever get close to the true posterior distribution over segmentations and, in so far as those early segmentations impact the analysis of novel segmentations, processing more data will emphasize rather than overcome the differences.

To address these two, I use a strategy known as *rejuvenation*[19]. Whenever the particles are resampled and there are likely to be many identical

19 There is some variation in the use of 'rejuvenation'. For example, Murphy (2012) refers to the resampling step as 'rejuvenation'. I follow Canini et al. (2009) in using 'rejuvenation' to refer to the act of resampling previous analyses. In any case, this is the preferred meaning of 'rejuvenation' within Computational Linguistics and Natural Language Processing judging from recent publications such as May et al. (2014).

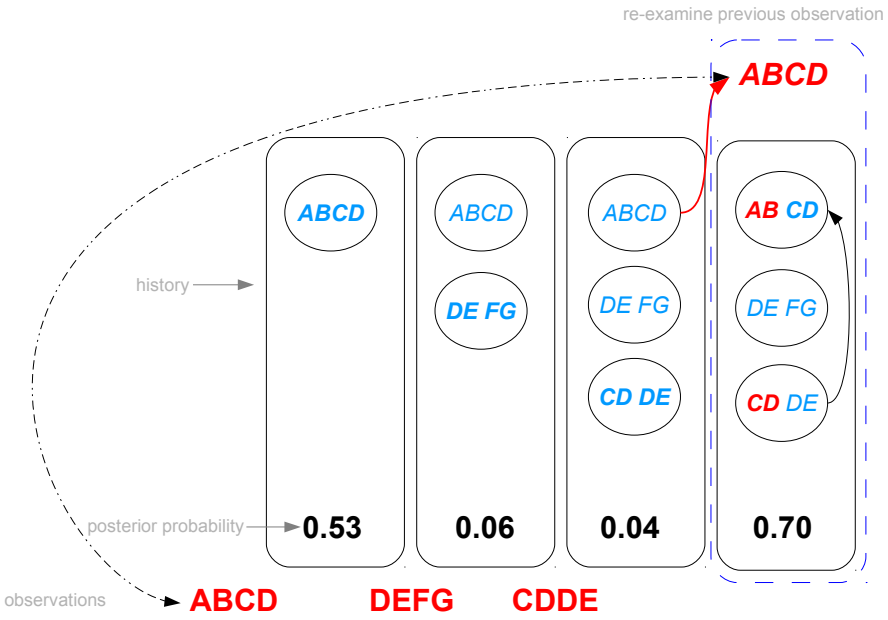


Figure 3.14: Schematic illustration of rejuvenation of a single particle. At $t = 3$, the hypothesis corresponding to the particle has low posterior probability because its initial guess – despite being highly probable at $t = 1$ – is not very likely given the other two observations. Rejuvenation allows a particle to revise earlier decisions by treating past observations as novel data which can be re-evaluated in the light of the other observations that have been processed in the meantime.

copies, we have each particle reanalyze some of the observations it has made so far conditional on its analyses of all other observations (Canini et al., 2009; Murphy, 2012). This 'rejuvenation' (an 'aged' particle is made young again) can be viewed as performing Gibbs sampling over some of the previously observed utterances and leads to a larger variety of particles.

The idea is illustrated in Figure 3.14 for the toy corpus on which I discussed incremental inference previously. It depicts the segmentation choices a single particle may make over time, referred to as its *history*, together with the posterior probability of these choices according to the model. Note how the posterior probability of the segmentation according has dropped to a mere 4% at $t = 3$ even though it has, by making a 'lucky' decision at $t = 2$, correctly segmented the last two observations. In a strictly incremental setting, there is no possibility for this particle to ever revise the analysis of the first observation even though, given its analyses of the other two observations, the original analysis made at $t = 1$ is very improbable. Rejuvenation allows the particle to re-examine

the first observation in the light of the other observations, treating it as if it were a new observations which will be segmented anew, ignoring its original analysis but taking into account the knowledge gained from the other observations.

Before I go on to describe rejuvenation in more detail, note that adding it to the algorithm requires that previous observations (and their analyses according to each particle) be stored. Thus, a particle filter with rejuvenation is no longer online in the sense that it can discard every observation after it has processed it. However, it still performs incremental inference, that is, it provides a posterior approximation after every observation has been observed, rather than only considering the full set of observations and delaying inference until then.

3.8.1 *Rejuvenation step using Metropolis-Hastings blocked sampling*

Let us associate with every particle $h_t^{(p)}$ at time t a history of segmentations $\boldsymbol{\sigma}_{1:t}^{(p)}$ which records the segmentations this particle posited for observations $\mathbf{u}_{1:t}$. A single rejuvenation step for a particle $h_t^{(p)}$ with history $\boldsymbol{\sigma}_t^{(p)}$ is defined algorithmically in Figure 3.15.

It randomly chooses an utterance u_j among the utterances observed up to time t. By optionally restricting the ability to re-examine previous observation to the most recent win (mnemonic for 'window') utterances, I add a memory constraint along the lines of Pearl et al. (2010) to my algorithm. While I sample uniformly among all utterances in a given window rather than sampling with probability proportional to how far in the past any utterance is, making this change to the algorithm is trivial. Preliminary experiments found no interesting difference arising from the use of 'decayed' rather than uniform sampling using a fixed-size window and I restrict attention to this. I return to the relationship of my algorithm to Pearl et al. (2010)'s Decayed Markov Chain Monte Carlo learner in the discussion.

After having determined an utterance to be re-analyzed, we pretend that this utterance hasn't been observed before but only became available after having processed the remaining $t-1$ observations, resampling an analysis for u_j and updating the particle accordingly. This is possible because the word segmentation model is exchangeable, and thus we can reuse the algorithm to sample a segmentation for an utterance given a particle h that we needed for the particle filter to begin with. However, because this algorithm samples from a distribution that is slightly different from that defined by the model, we need to correct for this difference.

For the particle filter, this correction step is part of calculating the particle weight. Here, we hold the weights constant and, instead, perform a Metropolis-Hastings accept/reject step. Rather than always using the proposed new segmentation σ', we calculate an acceptance probability p_a and only change the segmentation with probability p_a, keep-

function REJUVENATE($h_t, \mathbf{u}_{1:t}, \mathbf{sigma}_{1:t}, win = -1$)
  if $win = -1$ **then**
    uniformly sample $j, 1 \leqslant j \leqslant t$ ▷ unrestricted memory
  else
    uniformly sample $j, \max(1, t - win) \leqslant j \leqslant t$ ▷ restricted
  end if
  $h_t^{-j}, p_{old} = \text{REMOVE}(h_t, \sigma_j)$ ▷ remove old analysis
  sample $\sigma' \sim Q(\cdot \mid h_t^{-j}, u_j)$ ▷ sample novel analysis
  $q_{old} = Q(\sigma_j \mid h_t^{-j}, u_j)$
  $q_{new} = Q(\sigma' \mid h_t^{-j}, u_j)$
  $h_t, p_{new} = \text{UPDATE}(h_t^{-j}, \sigma')$
  calculate $p_a = \frac{p_{new} q_{old}}{p_{old} q_{new}}$ ▷ MH acceptance probability
  if NEXTDOUBLE $\leqslant p_a$ **then** ▷ accept novel analysis
    $\sigma_j = \sigma'$
  else ▷ reject novel analysis
    REMOVE(h_t, σ') ▷ undo update
    UPDATE(h_t, σ_j) ▷ add original segmentation back
  end if
end function

function REMOVE($h, \boldsymbol{w}_{1:n}$) ▷ for Unigram model
  $p_{true} = 1.0$
  for $i = 1 \rightarrow n - 1$ **do**
    $p_{true} = p_{true} \times \text{REMOVECUSTOMER}(w_i, h) \times P(\text{C} \mid h)$
  end for
  $p_{true} = p_{true} \times \text{REMOVECUSTOMER}(w_n, h) \times P(\text{S} \mid h)$
  return $\langle h, p_{true} \rangle$
end function

function REMOVE($h, \boldsymbol{w}_{1:n}$) ▷ for Bigram model
  $p_{true} = 1.0$
  $w_p = \$$
  for $i = 1 \rightarrow n$ **do**
    $p_{true} = p_{true} \times \text{REMOVECUSTOMER}(w_p, w_i, h)$
    $w_p = w_i$
  end for
  $p_{true} = p_{true} \times \text{REMOVECUSTOMER}(w_p, \$)$
  return $\langle h, p_{true} \rangle$
end function

Figure 3.15: Algorithm to perform rejuvenation and required helper functions to update particles.

ing the original segmentation with probability $1 - p_a$. I briefly walk through the algorithm given in Figure 3.15 and the involved helper functions.

To rejuvenate a particle h_t with associated history $\sigma_{1:t}$, we begin by randomly sampling an utterance u_j among the previously observed utterances $\mathbf{u}_{1:t}$ to be re-examined, optionally considering only the last win utterances (see above). Then, we 'move u_j to the end' of the observation sequence by first updating h_t as if it had never processed u_j so far. This is valid because the utterances are *exchangeable* and corresponds to removing all customers associated with words in the current segmentation σ_j for u_j, yielding an updated particle h_t^{-j}. The superscript indicates that this particle's seating arrangement is identical to that of h_t except for lacking all customers associated with σ_j.

The update is performed using the REMOVE helper function which removes the words (or, for the Bigram model, bigrams) in σ_j from h_t. As we will require $p_{old} = P(\sigma_j \mid h_t^{-j})$ in calculating the acceptance probability, we calculate it while actually removing σ_j from the particle's history.[20]

Once the particle has 'forgotten' everything about σ_j, we first we propose a new analysis σ' for u_j using the algorithm in Figure 3.4. Then, we calculate q_{new} and q_{old}, i.e. the probabilities of σ_j and σ' according to the proposal distribution $Q(\cdot \mid h_t^j, u_j)$.[21]

To calculate $p_{new} = P(\sigma' \mid h_t^{-j})$ we actually update the particle with the new segmentation, using the UPDATE function we already made use of in the particle filter. Again, note that in addition to the probabilities of the individual words those responsible for the utterance boundary need to be included in this calculation.

Finally, we calculate $p_a = \frac{p_{new} q_{old}}{p_{old} q_{new}}$ and, with probability p_a, accept the new proposal by replacing σ_j with σ'. Otherwise, we reject the new proposal and set the particle to its original state by first removing σ' and then updating it again using σ_j.[22]

Thus a rejuvenation step is, essentially, identical to a single sampling step in the blocked Metropolis-within-Gibbs samplers of Mochihashi

20 A detail that is easy to overlook is that this probability needs to include the probabilities governing utterance boundaries. Even though these are observed, in the Bigram model they affect the probability of the final word of a segmentation; and in the Unigram model, they affect the number of words in a segmentation.

21 Note that we only need to know these probabilities up to a normalization constant. Thus, in this case we can ignore conditioning on u_j and directly use equation 3.6 rather than 3.4.

22 Again, the fact that we treat table assignments implicitly through the addCustomer and removeCustomer functions means that we may recover a seating arrangement that differs slightly from the original seating arrangement. I found this to make no difference to ensuring recovery of the original seating arrangement. See the related discussion about intermediate updates in the algorithm in Figure 2.8 on page 45.

for $q = 1 \rightarrow r$ **do** ▷ perform r rejuvenation steps
  for $p = 1 \rightarrow n$ **do** ▷ for each particle
    REJUVATE($h_i^{(p)}, \mathbf{u}_{1:i}, \boldsymbol{\sigma}_{1:i}^{(p)}, win$)
  end for
end for

Figure 3.16:]
Code for adding rejuvenation to the particle filter described in Figure 3.5. Add these lines after line 12 of this algorithm. win is the size of the window in which the incremental learner can perform rejuvenation steps – setting it to -1 results in a particle filter that can re-segment every previously observed utterance, setting it to w restricts it to only the most recent w observations.

et al. (2009) and, for the slightly more complex Adaptor Grammar framework, Johnson et al. (2007b).[23]

3.8.2 *Adding rejuvenation to the particle filter*

We can add rejuvenation to the particle filter described in Figure 3.5 by adding the code in Figure 3.16 after line 16, i.e. after the particles were resampled.

Obviously, the number of rejuvenation steps r impacts runtime of the algorithm. In particular, if we set r very large and resample after every observation the particle filter essentially turns into a batch sampler over the entire corpus upon observing each observation. Also, while leaving runtime unaffected the choice of window size win is likely to affect performance in limiting the algorithms ability to revise earlier observations.

Considering the case where $r \rightarrow \infty$ also shows that adding rejuvenation to the particle filter leaves unaffected its asymptotic guarantee to converge on the true posterior in the limit as in this case, we just recover a batch sampler (indicating 'unlimited' memory). By setting $r = 0$, we get the strictly incremental particle filter, showing that the particle filter with rejuvenation includes both strict online learning and batch learning as boundary cases.

Setting the number of particles to 1 and sampling previous utterances according to a decay function my algorithm recovers the DM-CMC learner of Pearl et al. (2010), showing that it is a special case of a particle filter with rejuvenation that only uses a single particle.

23 It is worth pointing out, though, that Mochihashi et al. (2009) omit the accept/reject step in the description of their algorithm. While it is true that, for most problems, the acceptance probabilities are essentially 1.0 and omitting this step makes no difference, for small amounts of data this can make a huge difference.

3.9 EXPERIMENTAL EVALUATION

As before, we are interested in understanding how the particle filters compare to the batch samplers and how their performance varies as a function of the number of particles and, as another parameter, the number of rejuvenation steps. I limit myself to considering particle filters that use 1 and 100 particles. For r, I consider the values $1, 10, 100$ and 1000. To refer to the different algorithms, I extend the previously used PF-N notation which indicated the number of particles by adding the number of rejuvenation steps. Thus, PF-1-1000 is a particle filter with a single particle, performing 1000 rejuvenation steps after each resampling step.

Finally, I investigate the impact of window size for PF-1-1000 and PF-100-1000, considering 500 and 100. PF-1-1000-100 refers to a particle filter with a single particle, 1000 rejuvenation steps and a window size of 100.

3.9.1 *Unigram model*

As can be seen from Table 3.10, adding rejuvenation drastically improves performance of the particle filters, consistently letting them outperform the batch learners in terms of token f-score. The picture differs somewhat between UNI-NC and UNI-SC, and as before, I discuss these settings separately.

3.9.1.1 *Without possible word constraint*

Not surprisingly, the overall best performance is achieved by PF-100-1000, the particle filter that uses the most particles and rejuvenation steps. With 77%, its token f-score is roughly 10% higher than that of the best performing batch sampler, and looking at its boundary precision and recall scores we find that rather than over- or undersegmenting, it seems to strike a good balance between the two.

We also see that by only adding 10 rejuvenation steps to PF-100 we can boost its token f-score by more than 20% to 68%, slightly higher than that of BATCH-ANNEAL, illustrating that rejuvenation is indeed a very effective strategy to improve word segmentation. For PF-1, considerably more rejuvenation steps are required to yield good performance, with even 100 steps remaining well below 60% token f-score. Yet, with 1000 rejuvenation steps, even PF-1 attains higher token f-score than the batch learners. Unlike PF-100 and its rejuvenated variants, though, it is very prone to over-segmenting, attaining the overall highest boundary recall of 92% and remaining at a quite low boundary precision of 79%.

Looking at the impact of the window size, we find next to no difference in limiting the window to the most recent 500 utterances. Only being allowed to consider the most recent 100 utterances, however, results in a noticeable drop of roughly $5-6$% in token f-score which remains,

nevertheless, on par with or above that of the best performing batch learner. While this shows some sensitivity to the window size, it is consistent with the previous observation in Figure 3.12 that for the Unigram model, there is a rather gradual improvement in token f-score over time rather than the huge jump we observe for the Bigram model between 200 and 500 utterances.

Finally, though, it needs to be noted that despite the good segmentation scores, in terms of log-probability none of the particle filters can close the gap to either BATCH or BATCH-ANNEAL. This indicates that the particle filters are still performing worse inference than their batch counterparts. I return to this point in the discussion.

3.9.1.2 *With possible word constraint*

For UNI-SC, PF-1-100 and PF-1-1000 attain the best token f-scores, *not* PF-100-1000. This is somewhat surprising as the latter makes use of more particles but looking at log-probabilities, we see that, again, better token f-score is not correlated with higher log-probability. Indeed, PF-100 and its variants consistently get higher log-probabilities than PF-1.

A possible explanation for the exceptionally high token f-score for PF-1 – 86% which is a full 20% more than that of BATCH-ANNEAL – is again provided by boundary precision and recall. Whereas for UNI-NC, PF-1 had to suffer from comparatively low boundary precision, adding the possible word constraint results in boundary precisions that are consistently above 90%, starting from as little as 10 rejuvenation steps. Coupled with PF-1's preference for high boundary recall, this improved precision results in a considerable boost in token f-score.

Ironically, the reason that PF-100 does not benefit as strongly is that it is able to explore the hypothesis space more effectively, correctly zoning in on solutions that have a slightly lower boundary recall and mimicking, to some extent, the undersegmentation behavior observed for the batch samplers. Yet, the boundary recall for PF-100 is still noticeably higher than that of the batch samplers, explaining why it is able to attain a token f-score of 81% and why its log-probability is still lower than that of either of the batch samplers.

Turning to the impact of window size, again we see that limiting the window to the most recent 500 utterances has little to no impact. Yet, limiting it to the last 100 utterances yields noticeable drops in token f-score of $4-5\%$ which are accompanied by a drop in boundary recall rather than precision.

Thus, PF-100-1000-100 only attains 77% boundary recall as opposed to the 85% of PF-100-1000 with a virtually unchanged precision. While the segmentation metrics for PF-100-1000-100 are, in a sense, closer to that of BATCH and BATCH-ANNEAL, its inference performance is worse rather than better than that of PF-100-1000. While this is not surprising in as much as one would expect a limited window to yield worse infer-

		UNI-NC						UNI-NC					
algorithm	R	tf	bp	br	lp	lr	-logProb×10^3	tf	bp	br	lp	lr	-logProb×10^3
PF-1	0	.29	.49	.55	.19	.22	32.35	.55	.82	.54	.33	.44	29.20
	10	.38	.53	.86	.39	.24	29.93	.83	.91	.88	.67	.66	25.82
	100	.55	.65	.89	.51	.36	27.71	**.86**	.92	**.93**	.73	.68	25.67
	1000	.73	.79	**.92**	.65	.51	26.27	**.86**	.92	**.93**	**.76**	**.69**	25.54
	W=500	.72	.78	**.92**	.63	.51	26.43	**.86**	.91	.92	.73	**.69**	25.71
	W=100	.66	.75	.89	.54	.48	27.16	.81	.91	.86	.64	.66	26.15
PF-100	0	.46	.69	.58	.32	.38	29.18	.58	.86	.56	.39	.49	28.08
	10	.68	.80	.82	.55	.50	26.31	.79	.93	.82	.63	.64	25.69
	100	.72	.83	.86	.64	.56	25.65	.81	.93	.85	.69	.68	25.39
	1000	**.77**	.88	.87	**.69**	.61	25.32	0.81	.94	.85	.71	**.69**	25.20
	W=500	.77	.88	.86	.67	.61	25.42	.80	.94	.84	.68	.68	25.40
	W=100	.72	.85	.79	.54	.55	26.28	.76	.93	.77	.57	.61	26.06
DPM		.51	.86	.48	.29	.38	29.09	.56	.90	.51	.34	.45	28.86
BATCH		.62	**.92**	.64	.60	.63	24.91	.63	.93	.63	.59	.67	25.18
BATCH-ANNEAL		.66	**.92**	.74	.67	**.64**	**24.30**	.66	**.94**	.70	.66	.68	**24.38**

Table 3.10: Scores for the Unigram model with rejuvenation.

ence, it also shows that the relationship between segmentation scores and log-probabilities is not straight-forward for UNI-SC.

As a last point, for UNI-SC we find that limiting the window to the most recent 100 observations yields worse results than using only 100 rejuvenation steps on the entire history. This is something we could not observe for UNI-NC where, irrespective of window size, all learners making use of 1000 rejuvenation steps outperformed the corresponding learner that only used 100 rejuvenation steps.

3.9.2 *Bigram model*

I now turn to the Bigram model, again examining BI-NC and BI-SC separately.

3.9.2.1 *Without possible word constraint*

For BI-NC, even adding 1000 rejuvenation steps to PF-1 doesn't bring it up to the performance of either of the batch samplers although it considerably improves performance. Its token f-score is only slightly lower than that of BATCH, and its expected log-probability improves to 27.53×10^3 compared to 27.47×10^3 for BATCH. Overall, it is striking how effective adding a few rejuvenation steps is in improving performance even for a particle filter with only a single particle. For example, PF-1-100 outperforms PF-10,000 on all metrics by a fair margin even though, in a certain sense, it uses much less memory and performs less computation: whereas PF-10,000 has to extend $10,000$ individual particles for each utterance, thus performing $10,000 \times N$ utterance samples for a corpus of size N, PF-1-100 only extends a single particle but performs 100 additional sampling steps for each of the N utterance, totaling in $101 \times N$ samples.

Of course, this way of quantifying computational effort makes strong assumptions about the underlying architecture. For example, if there were no limit to the number of parallel computation steps that can be performed, using a very large number of particles with no rejuvenation would still, essentially, amount to only one computational step per utterance (as, by assumption, all updates can be performed in parallel). In contrast, rejuvenation requires sequential processing as it can only ever resample an utterance *conditional on the segmentations on all other utterances*. Rather than speculating on the proper way of quantifying computational effort in ways that are relevant to understanding language acquisition, I refrain from pushing this point (see also the discussion of cognitive plausibility below). For practical purposes, however, these kinds of considerations may be very relevant, and my findings seem to suggest that trading off rejuvenation against number of particles ought to be preferred (though see May et al., 2014, for discussion of rejuvenation and particle filters as applied to LDA).

Yet, all PF-1 variants suffer from rather low boundary precision with comparatively high recall, suggesting that despite their similarity in token f-score to BATCH the segmentations it identifies are still systematically different to the undersegmented segmentations identified by the batch learners. While we see that more rejuvenation steps lead to higher boundary precision, rather than increasing the number of rejuvenation steps even further I turn to see how a larger number of particles benefits from rejuvenation steps.

As expected, using 100 particles always leads to better results than using a single particle with the same number of rejuvenation steps. Somewhat surprisingly, though, we now see PF-100-100 outperform BATCH with respect to inference performance and, in terms of token f-score, even BATCH-ANNEAL. Interestingly, this is despite the fact that its segmentation metrics are slightly worse and, as before, the segmentation inferred by the incremental learner indicates slight oversegmentation rather than undersegmentation.

Turning to PF-100-1000 we find an even better log-probability and with 76% the best token f-score for the BI-NC model, 2% higher than that of BATCH-ANNEAL. While its log-probability is slightly lower with 26.74×10^3 vs 26.55×10^3, it is noticeable higher than that of BATCH.

Also, even though PF-100-1000 has a higher boundary recall than precision, there is only a 3% difference between its boundary precision of 86% and BATCH-ANNEAL's 89%. Thus, we find that using 100 particles coupled with 1000 rejuvenation steps allows an incremental algorithm to outperform BATCH, the vanilla batch sampler, and, with respect to token f-score, even BATCH-ANNEAL.

Finally, looking at the impact of window size, we see that smaller windows yield worse performance on all metrics. For PF-1-1000, we find that restricting the learner to the most recent 500 observations has only a very minor (but still noticeably larger than for UNI-NC) impact, resulting in drops of roughly $2-3\%$ on all segmentation metrics and a slightly worse log-probability. Limiting it to the most recent 100 utterances results in a much larger drop, bringing token f-score down from 65% to 51%. In fact, we find that PF-1-1000-100 performs worse than PF-1-100, showing clearly that it is not only the number of rejuvenation steps performed but also the 'memory' of the learner in the form of the window that impact performance and that this is much more pronounced for the Bigram than for the Unigram model.

The trend is similar for PF-100-1000. Here, limiting the window to 500 observations results in a relatively larger drop in token f-score from 76% to 70%, and again PF-100-1000-100 performs worse than PF-100-100.

The impact of window size is expected given the previous discussion of Figure 3.12. There was a major jump between 200 and 500 utterances, indicating that the ability to re-examine observations that are more than 100 utterances in the past is important. Further support for this comes from Figure 3.17 which compares the token f-score for different

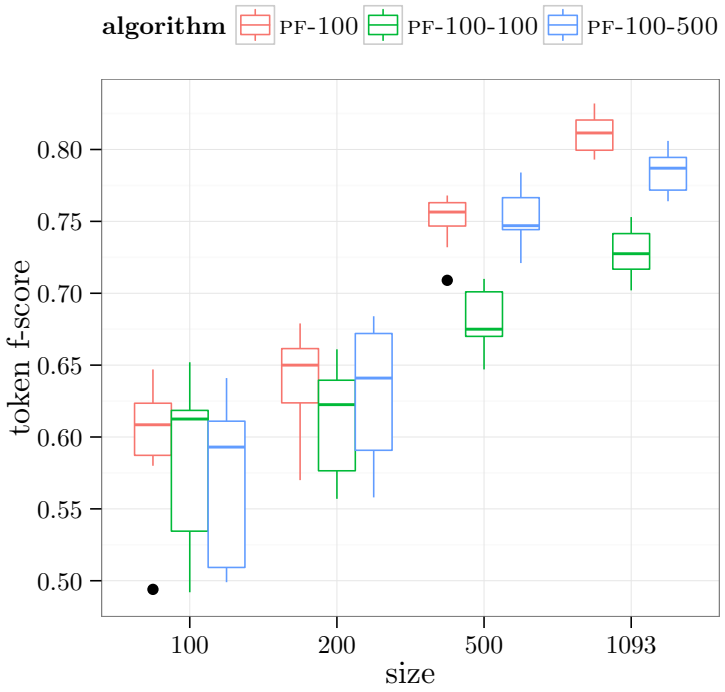


Figure 3.17: Comparing the impact of window size for PF-100-1000 for BI-NC.

input sizes for PF-100-1000, PF-100-1000-100 and PF-100-1000-500, and we see that, as expected, there is no noticeable difference between the three for $n = 100$. Yet, PF-100-100 begins falling behind at $n = 200$ and, finally, PF-100-500 falls behind after $n = 500$ despite performing the same number of rejuvenation steps as PF-100-1000.

3.9.2.2 *The effect of the possible word constraint*

Adding the possible word constraint changes the picture in an interesting way. As before, we witness an overall improvement in scores for the incremental learners. In fact, for BI-SC even adding as little as 10 rejuvenation steps to either PF-1 or PF-100 boosts segmentation accuracy to 73%, outperforming BATCH and coming close to BATCH-ANNEAL's 75%. Somewhat surprisingly, perhaps, BATCH is not only outperformed in terms of token f-score but also in terms of log-probability, indicating that for the thus constrained model incremental inference with rejuvenation is more efficient than batch sampling without annealing.

Increasing the number of rejuvenation steps keeps improving token f-score for both PF-1 and PF-100, and they are outperforming BATCH-ANNEAL starting from $r = 100$. Yet, as for BI-NC the gap between BATCH-ANNEAL's log-prob and that of the incremental learners is not closed completely for BI-SC, showing that, with respect to inference performance, BATCH-ANNEAL is the best choice.

Another rather striking point is that the overall best token f-score is achieved not by PF-100-1000 but by PF-1-1000, with 83% against 81%. Yet, PF-100-1000 attains better lexicon scores that are on par with those of BATCH-ANNEAL and, indeed, its boundary precision of 93% is only 1% lower than that of BATCH-ANNEAL whereas its recall is 6% higher, accounting for the better token f-score. In contrast, despite slightly higher token f-score PF-1-1000 shows signs of mild oversegmentation with a boundary precision of slightly less than 90% versus a rather high recall of 92%, thus indicating that it deviates more strongly from BATCH-ANNEAL and thus the posterior implied by the model than PF-100-1000.

This fits the pattern we already observed for UNI-SC where we also found PF-1 rather than PF-100 to yield the highest token f-scores despite worse inference according to the negative log-probability. I believe that the same explanation applies in these two cases: incremental learning generally favors segmentations with higher boundary recall than batch inference, even if the global solution preferred by the model has lower recall. Whereas PF-100 is slowly approaching the globally preferred solution, PF-1 seems to use be unable to 'overcome' its preference for high-recall solutions even using rejuvenation. Thus, whereas PF-100-1000 shows evidence for mild undersegmentation PF-1-1000 presumably attained its slightly higher token f-score precisely because it does not exhibit this behavior.

Finally, for the impact of window size we see a similar picture as for BI-NC: while limiting the window size to 500 now has a very small impact, limiting it to 100 results in a noticeable drop in performance although, in this case, even the so-constrained learners attain a token f-score that is higher than that of BATCH if lower than that of BATCH-ANNEAL.

In conclusion, even when incremental learners are provided with the ability to perform rejuvenation a constraint like the possible word constraint boosts segmentation performance considerably.

3.10 DISCUSSION

The discussion of the algorithms so far has focused on a technical analysis of the performance of the different algorithms. Summing up the detailed discussion of the preceding paragraph, the main findings of my experiments are that

- all else being equal, incremental particle filters perform better with more particles
- without rejuvenation, even large numbers of particles perform considerably worse than non-incremental batch inference
- adding a constraint on possible words has a huge positive effect on particle filter performance (both with and without rejuvenation) but hardly affects batch inference

		BI-NC						BI-SC					
algorithm	R	tf	bp	br	lp	lr	-logProb$\times 10^3$	tf	bp	br	lp	lr	-logProb$\times 10^3$
PF-10,000	0	.44	.58	.77	.35	.39	29.42	.66	.80	.74	.49	.63	27.78
PF-1	0	.20	.38	.73	.19	.23	33.30	.53	.67	.66	.34	.54	29.58
	10	.32	.49	.88	.29	.26	30.40	.73	.80	.87	.57	.63	27.36
	100	.53	.62	**.92**	.40	.36	28.57	.78	.84	.91	.63	.66	26.94
	1000	.65	.73	**.92**	.51	.46	27.53	**.83**	.89	**.92**	.65	.67	26.58
	w=500	.63	.71	.89	.47	.45	27.82	.82	.88	.91	.63	.67	26.69
	w=100	.51	.62	.83	.36	.39	29.16	.71	.81	.81	.51	.63	27.50
PF-100	0	.33	.49	.75	.28	.33	30.85	.62	.76	.72	.44	.60	28.32
	10	.55	.67	.82	.43	.44	28.15	.73	.86	.82	.59	.66	26.88
	100	.65	.77	.84	.52	.50	27.20	.79	.89	.85	.65	.68	26.50
	1000	**.76**	.86	.88	.63	.60	26.74	.81	.93	.86	**.69**	**.72**	26.35
	w=500	.70	.81	.85	.56	.56	27.11	.79	.92	0.84	.64	.69	26.46
	w=100	.61	.73	.82	.43	.48	28.25	.73	.87	.79	.53	.65	27.13
BATCH		.66	.88	.71	.58	.62	27.47	.66	.88	.68	.57	.69	27.58
BATCH-ANNEAL		.74	**.89**	.83	**.65**	**.63**	**26.55**	.75	**.94**	.80	**.69**	**.72**	**26.24**

Table 3.11: Scores for particle filters with and without rejuvenation on BI-NC and BI-SC. For PF-1-1000 and PF-100-1000, I also give scores for window sizes of $w = 500$ and $w = 100$.

- in terms of segmentation but not log-probability, a particle filter with rejuvenation and the possible word constraint can outperform batch inference

Here, I will address more directly the question of what my findings have to say about language acquisition.

While concerns about the inference *mechanism* are of only secondary interest from the general Bayesian modeling perspective outlined in the first chapter, incremental algorithms are interesting in at least two ways. First, they make it possible to connect Bayesian models rather directly to psycho-linguistic experiments, and the particle filter algorithm discussed here has in fact been used two studies of human word segmentation (Meylan et al., 2012; Kurumada et al., 2013).

Moreover, in so far as it is an asymptotically correct inference algorithm, study of a 'constrained' incremental algorithm can shed some light on how processing limitations impact the conclusions that can be drawn from specific input. As a concrete example, we saw that a requirement for incremental processing seems to dampen the preference for undersegmentation exhibited by the word segmentation models. This the idea of the *rational process* approach of Sanborn et al. (2010), also taken up for word segmentation in Pearl et al. (2010) and, more recently, Phillips and Pearl (in press).

Generally speaking, the idea is that differences between the conclusions drawn by human learners and those implied by a Bayesian model may be due to the former performing inference in a constrained setting. Although not mentioned in the rational process literature, this idea bears some resemblance to that of *type-II rationality* as defined by Good (1971), that is, "maximization of expected utility taking into account deliberation costs". I discuss this idea in some more detail giving some concrete examples before pointing out possible problems with these kinds of interpretations.

3.10.1 *The role of processing constraints*

One can interpret my findings in this framework by identifying the dimensions along which the incremental algorithm can be characterized as being more constrained than the 'ideal' batch learners.

3.10.1.1 *Order effects*

For example, Figures 3.1 and 3.2 illustrated how in such a setting, 'order effects' may arise because the distributions induced by different subsets of the entire data may be dramatically different even though the posterior given the full data is identical. Indeed, due to the requirement of making local decisions, particle filters can be used to derive order effects from models that are, strictly speaking, order insensitive due to the

Particles	% Correct ORDER1	% Correct ORDER2
1	0	4
10	6	67
100	47	100
1000	100	100

Table 3.12: Comparison of how often a particle filter with different number of particles correctly identified the MAP segmentation of the two 3 utterance corpora in Figure 3.1 (ORDER1) and Figure 3.2 (ORDER2).

exchangeability assumption built into most Bayesian models (Sanborn et al., 2006).

To illustrate this for word segmentation, consider again the example in Figure 3.1. I run 100 independent simulations of particle filters with 1, 10 and 100, and 1000 particles on this toy corpus and count the number of times in which the correct global MAP segmentation was identified. The ORDER1 column of Table 3.12 indicates that for this corpus, 1000 particles are required to reliably identify the MAP segmentation. Even using 100 particles only correctly identifies the MAP segmentation of the corpus in roughly 50% of the simulations.

We have already seen that increasing the number of particles addresses leads to better performance (see Tables 3.3 and 3.2). Here I show how simply changing the order to that in Figure 3.2 also improves performance. Recall that due to the exchangeability of the word segmentation models, the MAP segmentation for both corpora segments all three utterances identically. Thus, from the perspective of the model there is no difference between these orderings, Yet, the ORDER2 column of Table 3.12 shows that performance is much better on this alternative ordering for particle filters that use only a few particles.

From an acquisition point of view, this raises the possibility that certain ordering properties exhibited by child-directed speech might be crucial for constrained learners such as infants to adequately solve word segmentation and related acquisition problems. Of course, making this connection requires some way of linking the constraints inherent in a particle filter to those human learners are subject to. One possibility would be to argue that 'number of particles' and 'window size' correspond to the memory limitations of human infants. In this thesis, however, I will not further investigate this possibility for reasons detailed below.

3.10.1.2 *Importance of substantive constraints*

Another interesting observation may be derived from the huge performance difference we observe for the particle filters for the models with and without the possible word constraint. Recall from both Chapter 2

and the experiments in this chapter that for batch learners, there was virtually no difference in performance between the model with and without possible word constraint – according to the segmentation model, the same segmentations are assigned high posterior probability irrespective of whether or not a possible word constraint is assumed.

Yet, for all incremental learners I considered we found that the constraint makes a huge difference. This suggests that even though, as shown by performance of the batch learners, the input by itself contains sufficient information to identify reasonable segmentations, the incremental learners I considered are unable to leverage this information unless a substantive constraint on the shape of possible words (here, that each word has to contain at least one syllabic segment) is built into the model.

Assuming that infants are incremental learners as well and that they are constrained in ways similar to particle filters, this suggests that for human (as opposed to 'ideal statistical') learners a substantive constraint on the form of possible words is necessary to solve the segmentation problem. I am not aware of any experimental work addressing the exact nature of such a constraint in infants although Norris et al. (1997) reports evidence for a possible word constraint in adult segmentation.

3.10.1.3 *Less is more?*

Turning to the learners with rejuvenation, we note on the one hand a relaxation of processing constraints in adding 'memory' and the ability to revise earlier decisions to the incremental learner. On the other hand we get two additional parameters which control just how constrained the algorithm is: number of rejuvenation steps and size of the window within which the algorithm can resample previous observations.

This results in a learner that is a generalization of the Decayed Markov Chain Monte Carlo learner of Pearl et al. (2010) and, more recently, Phillips and Pearl (in press). Not surprisingly, then, my results are largely in line with theirs.[24] Like them, I find that the 'constrained' incremental learners (with rejuvenation) can outperform their batch counterparts in terms of segmentation performance, and that this trend is more pronounced for the Unigram than for the Bigram model. Indeed, this observation isn't new, at least for the Unigram model: Goldwater (2007) already pointed out that the DPM algorithm proposed by Brent (1999) results in higher token f-scores but lower log-probabilities according to the Unigram model than her batch Gibbs sampler.

24 A subtle difference is that they also consider pre-syllabified input but only consider an 'unrestricted' base distribution. In a sense, however, one can view my use of a constrained base distribution as a slightly weaker version of their pre-syllabifying the corpus – while they ensure through pre-processing that posited words will always be made out of valid syllables, I merely enforce that every posited word contain at least a single syllabic element.

An attractive interpretation proposed by Pearl et al. (2010) and Phillips and Pearl (in press) of this kind of finding – an algorithm that is more 'constrained' than another algorithm performs better – makes use of the "Less is More" hypothesis (Newport, 1990): the idea that certain kinds of 'cognitive limitations' can aid rather than hurt infants in acquiring their first language. Concretely, the idea is that by not having sufficient memory to represent examples in a way that makes it possible to detect 'global' patterns that might also be a source of confusion, infants avoid many mistakes and identify more useful regularities than adult language learners.

This can, indeed, be seen as providing a possible explanation of the surprisingly good performance of the rejuvenated particle filters for the Unigram model – we've already pointed out that the incremental learners have much less of a tendency to undersegment than their batch counterparts, and that this seems to fall out of the requirement of incremental processing, introducing a (soft) greedy strategy in which words acquired early on are consistently segmented out of the input, avoiding undersegmentation. In contrast, the batch learners take a 'global view' of the data and pick up frequently co-occurring words much more directly, ironically leading to undersegmentation that results in lower token f-scores. Similarly if somewhat less pronounced, for the Bigram model we find the best performing particle filters to attain higher boundary recall than the batch learners.

Taken together, then, my findings seem to also speak in favor of a "Less is More" effect in word segmentation. Yet, I am reluctant to draw strong conclusions along those lines as the best performing algorithms are not the most constrained I considered – in contrast, they are those that come rather close to the batch learners with which they were contrasted, requiring substantial amounts of rejuvenation steps and, as is clear from the experiments on window size, a rather large memory. Thus, in a sense I observe a clear "more is more" effect for the learners, probably missed by Pearl et al. (2010) and Phillips and Pearl (in press) because the Decayed Markov Chain Monte Carlo learner is only one of the many possible particle filters one can consider.

3.10.2 *Problems for 'Rational Process' interpretations*

While it is tempting to take the incremental algorithms of this chapter as proposals about actual inference mechanisms and, as just done, derive predictions from a 'Rational Process' perspective, I believe there to be several problems with this view.

For one thing, it is hard to properly judge the 'cognitive plausibility' of an algorithm. In our case, some of the obvious questions are how many particles counts as cognitively plausible – 1, 2, 1000 or even 1,000,000?

'Intuitively', 1 particle sounds more plausible than 1,000,000 but other than appealing to intuition, I don't see a principled way of de-

ciding this question. One possible way is to compare performance as a function of particles to actual human performance, as in (Meylan et al., 2012). Yet, in a sense this pre-judges the question by assuming that modifying the number of particles *is* the right way of modeling constraints; but perhaps it is another determinant such as window size or the number of rejuvenation steps. A related problem is that it is currently not clear how 'high-level' notions such as a particle or a rejuvenation step could be mapped onto properties of the brain. In light of these issues and in the absence of detailed studies along those lines, I believe the most reasonable stance to be to simply not relate number of particles and cognitive processing in overly strong ways.

Similarly with respect to rejuvenation steps – 'intuitively' an algorithm that performs 1000 additional samples per observation looses a lot of the appeal it got for performing incremental inference but how many – if any – rejuvenation steps are cognitively plausible? With respect to the Decayed Markov Chain Monte Carlo learner, Phillips and Pearl (in press) assume that it is to perform 20,000 additional samples per observation, arguing that it still requires "approximately 74% less processing than the [Batch sampler], a significant processing reduction" because the batch samplers performs 20,000 iterations over the entire corpus. It is, however, neither clear that the batch sampler really requires as many iterations as it is usually run for[25] nor that this is a meaningful dimension along which to compare the algorithms – the major 'cognitive plausibility' issue of batch algorithms is their processing data in large batches rather than the number of iterations they perform over it.

Relatedly, then, the kind of resampling steps performed by the DMCMC learner and the particle filters with rejuvenation are nothing but partial batch samples. If this kind of processing is deemed undesirable because of cognitive implausibility for the batch samplers, why isn't this considered to raise the same issue for incremental learners? Phillips and Pearl (in press) are correct to point out that limiting the learners memory adds an additional constraint to batch processing but this just raises the next question how this constraint ought to be implemented properly. They chose a decay function, I use a fixed window size – both choices raise an additional question, namely which parameter for the decay function to choose and which window size to use. I found that a window size of 500 and 100 lead to rather different results, and that larger memory is better – but how large exactly is plausible?

25 For example, I found no difference between running the batch sampler for 2000 or 20,000 iterations for all the word segmentation models. The reason I used 20,000 iterations for my experiments is that I was interested in both generating a reasonably large number of posterior samples and, to some extent, the fact that this is the number of iterations Goldwater (2007) and Goldwater et al. (2009) reported for the Gibbs samplers I used, rather than having determined this to be the minimal number required for the algorithm to converge.

3.10.3 *Suboptimal model and suboptimal algorithm*

Finally, there is something odd about taking the observation that *worse inference* can lead to *better segmentation* as evidence for a particular segmentation *model*. This is how one might be tempted to interpret Pearl et al. (2010) and Phillips and Pearl (in press) – "cognitively plausible learners outperform the ideal" (Phillips and Pearl, 2012). While I do believe that these kinds of findings are interesting, I think there are reasons to be cautious with this kind of interpretation.

In particular, the 'constrained-is-better-than-ideal' interpretation presupposes some variant of the idea that humans may apply a 'suboptimal' algorithm to a particular model and, *precisely because of the suboptimal algorithm*, exhibit good performance. Of course, one cannot fully rule out this possibility; talk about how particular models relate human performance are notoriously difficult, and we always face the problem that human behavior only imperfectly reflects the underlying mechanisms that give rise to it. Indeed, Kripke (1982) claims that it is impossible to empirically distinguish between "someone assumes a particular model but makes a mistake in applying it" from "someone assumes a different model and correctly applies it" and raises this as a general challenge to cognitive science (for a critical discussion, see Chomsky (1986)).

Even ignoring this philosophical problem, I find it odd to assume suboptimal algorithms rather than trying to understand in what sense the model is inadequate.

At this point, I believe that rather than raising more questions to which, admittedly, I do not have the answers, I should elaborate on how I believe my findings ought to be interpreted.

3.10.4 *Interpreting results from incremental algorithms*

Given that I am reluctant to interpret the experimental results of the algorithms 'mechanistically', what do they tell us about human language acquisition? As discussed in Chapter 1, I believe the strength of computational modeling to be its ability to evaluate specific proposals and their consequences in a principled fashion and this is how I believe my findings ought to be viewed as well. Thus, rather than taking my algorithms as specific proposals about actual mechanism I take them as *tools to study what kind of issues may arise in incremental inference* in principle, providing useful information about the properties of models which can then be explored in the Bayesian framework.

To provide two examples which will be taken up – again within a modeling framework that does not focus on inference algorithms – I consider briefly the impact of the possible word constraint and the fact that sub-optimal inference leads to better segmentation.

3.10.4.1 *Sensitivity to input size*

With respect to the possible word constraint, I noticed that without it incremental learners performed considerably worse on the Bigram model than on the Unigram model. My analysis suggested that the major issue was that the Bigram model does, indeed, early on assign high probability to segmentations which are not linguistically plausible; and that because of this, an incremental learner may *correctly commit* to segmentations with low segmentation accuracy early on and simply be *unable to recover from this later*.

I also found that this trend was less pronounced for the Unigram model, suggesting – not implausibly – that *a more complex model may simply require more input to induce segmentations which correspond to our expectations*. As this issue was somewhat alleviated by adding the possible word constraint, this raises the question whether a more complex model may actually require a substantive constraint to perform well not only on large amounts of data but also on little amounts. In the next chapter, I will examine this question in a detailed way, comparing several models of differing 'complexity' as to how their performance depends on 'possible word constraints'.

3.10.4.2 *Modeling assumptions*

Relatedly, I believe the finding that sub-optimal inference results in better segmentation to be informative not about the mechanism humans might use but about what kind of model might be more adequate. Thus, rather than trying to 'fit' a bad model to human performance by positing a 'bad' inference algorithm (an idea which I criticized above), one can use properties of the inference algorithm as informing construction of more adequate models.

For example, the observation that using more particles with rejuvenation lowers segmentation score while increasing inference performance provides additional evidence that *undersegmentation is an effect of the ability to identify 'global' patterns*. A model that avoids this, then, needs to have some means of either 'explaining these patterns away' or somehow limiting the amount of input over which patterns can be detected. I will explore the former idea – allowing models to explain away patterns – in the next chapter but believe that my findings suggest an interesting question for future work. In particular, is there a way of defining a model that enforces the kind of 'locality constraint' that arise in incremental processing and allow the particle filter with rejuvenation to identify high accuracy segmentations under the Unigram model?

BEYOND EXCHANGEABILITY Recalling that part of the reason why incremental and batch learners perform so differently is that the former are simply unable to properly handle the kind of 'long range' relations that, according to the underlying model, hold between obser-

vations that are arbitrarily far apart in the input. This is a consequence of *exchangeability*, i.e. the assumption that the order of observations should not affect the conclusions drawn from them.

My observation that it is precisely *because* the incremental algorithm is unable to connect observations that are very far apart in the input that it performs better suggests that assuming a non-exchangeable model may be an alternative way of addressing the problem of under-segmentation. For example, recent work in Machine Learning on non-exchangeable models using the *distance dependent Chinese Restaurant Process* (Blei and Frazier, 2011) may point towards the construction of models that imply the kind of 'useful' locality constraint directly, rather than deducing it from processing limitations.

While I leave exploration of this idea for future work, it is worth pointing out that for non-exchangeable models particle filters may prove to be the only feasible inference algorithms as common Markov Chain Monte Carlo algorithms require exchangeability to be computationally tractable. In contrast, the correctness of a strictly incremental particle filter is independent of any assumption of exchangeability and can directly be applied to such a model.

To conclude, then, I believe the results in this chapter to contribute in multiple ways to the study of word segmentation in a Bayesian framework. One can take the algorithms as proposals about mechanism, following work such as Pearl et al. (2010) and Phillips and Pearl (in press); my preferred interpretation, though, is in terms of suggesting novel research questions about *models*; in particular how input size and model assumptions interact and whether assumptions such as exchangeability are adequate for our purposes. The next chapter directly addresses the first of these two questions and studies how a large class of different models performs on different amounts of input.

4

STUDYING THE EFFECT OF INPUT SIZE FOR BAYESIAN WORD SEGMENTATION

Studies of computational models of language acquisition depend to a large part on the input available for experiments. In this chapter, I study the effect that input size has on the performance of word segmentation models embodying different kinds of linguistic assumptions. This directly addresses two questions raised by the findings of the previous chapter:

1. Do complex models such as the Bigram model rely more heavily on additional constraints to yield good performance on little data than 'simple' models such as the Unigram model?

2. How can undersegmentation behavior of models be addressed by changing the assumptions built into the model?

Because currently available corpora for word segmentation are not suited for addressing this question, I perform my study on a novel corpus based on the Providence Corpus (Demuth et al., 2006).

I find that, indeed, input size can have dramatic effects on segmentation performance and that, somewhat surprisingly, models performing well on smaller amounts of data can show a marked decrease in performance when exposed to larger amounts of data. I also find that moving towards more complex models requires the addition of strong constraints on possible words to yield good performance on even large amounts of data, suggesting that the answer to our first question is yes. As for the second question, I show that combining constraints on possible word forms with modeling word-dependencies using collocations successfully addresses the undersegmentation problem.

In addition, I present the data-set on which I perform the experiments comprising longitudinal data for six children. This corpus makes it possible to ask more specific questions about computational models of word segmentation, in particular about intra-language variability and about how the performance of different models can change over time.[1]

4.1 INTRODUCTION

Segmenting a stream of sounds into discrete words is one of the first tasks that children acquiring their native language have to tackle. Computational models of word segmentation enable us to study this problem

1 The corpus and the code to run the experiments is available at http://web.science.mq.edu.au/~bborschi/.

in a controlled and detailed manner, allowing for example for an examination of the usefulness of different kinds of cues or different learning strategies, as discussed in more detail in Chapter 1.

Just as important as the actual models, however, is the adequacy of the input used to evaluate them — if we are interested in answering questions about human language acquisition, the data we evaluate our models on needs to be comparable to what children are likely to have access to. To this end, several datasets of phonemically transcribed child directed speech (CDS) have been constructed in several languages, ranging from English to Italian, Polish, Sesotho and Chinese (Brent and Cartwright, 1996; Gervain and Erra, 2012; Boruta and Jastrzebska, 2012; Johnson, 2008a; Johnson and Demuth, 2010). In addition to cross-linguistic variation adequate computational models also need to handle language-internal variation along several dimensions, a topic that has so far received little interest.

In this chapter, I look at one of the most basic points of variation: the actual size of the input to the learner. It is common to evaluate models on a single large dataset. In fact, size seems to be so important that often several corpora are concatenated to yield a larger input set. Yet, to my knowledge little work has looked at understanding how input size affects segmentation performance. The longer children are exposed to language, the more data they are exposed to and the better at their language they become, something one would expect from adequate models of language acquisition as well.

I run my experiments on a novel dataset that contains longitudinal data for six children from the Providence Corpus (Demuth et al., 2006). It has two advantages over the current defacto standard for word segmentation studies for English, the Bernstein-Ratner-Brent corpus (Brent, 1999, in the following, BRB Corpus).

First of all, it cleanly separates CDS that is directed at different children with one separate corpus per infant; in contrast, the BRB Corpus contains data from 9 different children with no clear indication of the different portions.[2] In addition, recording for some of the children in the BRB corpus began as late as month 21 and for others as early as month 13, raising a potential issue with respect to the comparability of the individual corpora.

In contrast, the Providence Corpus provides data for all of the children starting from month 16 at the latest and starting from month 11 at the earliest and thus constitutes a much more homogeneous data set. Finally, the BRB corpus with its roughly 10,000 utterances is too small to systematically address questions about input size.

My transcription of the Providence Corpus contains more than 90,000 CDS utterances in total and spans a period of several months for all of the children. This makes it possible to both compare inter-child vari-

2 Thus, in Chapter 2 and 3 I restricted attention to a sub-part of this corpus that corresponds to a single child.

ability in word segmentation across comparable situations and to study developmental changes in individual children over a period of several months. As such, the resource will allow researchers to ask a wider range of questions than is currently the norm.

For the experiments on the effects of input size I focus on one of the six sub-corpora of the dataset, yet I describe and make available the full data so as to enable other researchers to take advantage of this new resource as well. Other studies that make use of my data are Börschinger and Johnson (2014) (forming the basis for Chapter 5 of this thesis) and Synnave et al. (2014).

The outline of the chapter is as follows. First, I provide background about the original Providence Corpus, my way of phonemically transcribing it and the properties of the new data-set I created. In section 4.3, I introduce the models of word segmentation which I examine with respect to the effect of input size in section 4.4. Section 4.5 discusses my findings and the final section concludes.

4.2 THE PROVIDENCE CORPUS

The Providence Corpus (Demuth et al., 2006) was collected during 2002-2005 from participants in southern New England. It contains longitudinal audio/video recordings of 6 monolingual English-speaking mothers and their children from approximately 1-3 years during spontaneous interactions at home. The children included 3 boys (Alex, Ethan, William) and 3 girls (Lily, Naima, Violet). Each was recorded for approximately 1 hour every 2 weeks beginning at the onset of first words. Two of the girls have denser corpora, with weekly recordings from 1;3-2;10 (Naima) and 2;0-3;0 (Lily), and Naima's recordings tended to be 1.5 hours long. There is therefore more data for this mother and child. Lily's mother also talked quickly; there is therefore much data from Lily's mother as well. Recording began around one year or once the parent reported that the child was producing approximately four words.

Digital audio/video recordings took place in each child's home. In most cases a research assistant came to set up the recording equipment and then left, encouraging naturalistic spontaneous speech interactions between parent and child. The children and parent (usually the mother) wore a wireless Azden WLT/PRO VHF lavalier microphone pinned to the collar. The child's radio transmitter was stored in a child-sized backpack. The radio receiver was attached to the top of a small Panasonic PV-DV601D-K Mini digital video recorder placed on a tripod nearby. Although parent and child could move freely about, the video information was useful in determining the context of what was being discussed, including possible target words. The availability of video would allow future work along the lines of Frank et al. (2009) and Jones et al. (2010) although so far, we haven't made direct use of the video recordings.

The digital audio/video recordings were downloaded onto a computer, and both adult and child speech were orthographically transcribed using CHAT conventions (cf. MacWhinney (2000)). The child data — but unfortunately not the caregivers' — were then also transcribed in phonemic transcription. All mother and child transcriptions, as well as audio/video files, can be found on the CHILDES database http://childes.psy.cmu.edu/. I used the XML version of the data for the transcription process.

The transcripts include both the child's and the adults' utterances. As potential input to computational models of word segmentation, however, we are exclusively interested in the CDS parts of the transcripts which we focus on in the following. Note that all corpus statistics mentioned in the rest of the paper exclude child utterances.

4.2.1 *Producing a phonemically transcribed version*

To find CDS utterances, for all six children we extract the orthographic transcriptions for all utterances made by caregivers from the XML transcripts of the Providence corpus, starting from 11 months up to and including 22 months.[3] This makes the data qualitatively comparable to the BRB Corpus that includes CDS from between 13 and 21 months of the children's age. In total, we extract 101,451 utterances with a 9,395 distinct (orthographic) types but the number of utterances some utterances are not transcribed (see section 4.2.1.1). I also ignore all child utterances as I am interested in generating child-*directed* input for the models.

To turn the orthographic representations into a phonemic format that is suitable for studying language acquisition, we perform a four step process of filtering, dictionary look-up, heuristic construction of pronunciations for unknown types and manual transcription of unknown types not covered by the heuristic as well as correction of mistakes made during earlier steps.

4.2.1.1 *Filtering words*

We manually remove types that we consider to be obvious non-words, in particular interjections such as *hmmmhmmm* or *mmmmhmmm*, obvious onomatopoeic wordplay such as *nananana* and unintelligible words which are transcribed in the Providence Corpus as *xxx* and *yyy*. This is consistent with the procedure followed by Brent (1999) and, more recently, Boruta and Jastrzebska (2012), making the corpus comparable in this respect to theirs. We do not, however, remove these items in cases where the resulting utterance would have been rendered fully unintelligible or where a word that should have been excluded according

3 Available at http://childes.psy.cmu.edu/data-xml/Eng-USA/Providence.zip

to the above criteria was used as an actual word in a large number of cases.[4]

In total, we identify 785 such non-words and we remove all occurrences of these types from the transcript, leaving the remaining words in the utterance:[5] utterances including any of the non-words are still transcribed as long as there is at least one word left after removing all non-words.

A total of 7,362 utterances are thus completely ignored, with 6,123 utterances consisting of exactly one of these filtered elements, in particular *xxx* (unintelligible, 2215), *oh* (525) and *hmmm* (521).

4.2.1.2 *Dictionary lookup*

After filtering, we perform a simple dictionary-lookup transcription using a phonemic dictionary. We use the VoxForge dictionary which uses a standard phone set for American English, corresponding to the DARPABET coding.[6] We also provide a script that maps this representational scheme into one-character-per-phoneme representations that are required by some of the currently common word segmentation tools.[7] If there are multiple pronunciations available for a type, we always pick the first one. While this constitutes an idealization we believe that an explicit idealization is to be preferred over an overly simplistic method of artificially introducing variability such as randomly choosing a pronunciation.

In total, the VoxForgeDictionary covers 7035 of the 8610 remaining types in the data, leaving 1575 of the types untranscribed. We transcribe these words manually, using a simple pre-processing heuristic to aid the process.

4.2.1.3 *Heuristically constructing pronunciations for unknown words*

Many of the unknown words are either forms of types that already are in the lexicon, e.g. possessives (*Elmo's*) or plurals (*Legos*), or compounds of two types that are both in the lexicon individually (*frenchtoast*, *teddybear*). We handle the former case by simple rules operating on the orthographic forms directly.

---

4 The former applies mostly to cases where an item is mentioned rather than used, e.g. "Does the baby say 'Wah wah'?"; the latter, for example, applies to 'bonk' which, in addition to its onomatopoeic use, also occurs as a verb in the corpus, including its preterite and participle. The data includes the full list of filtered items as well as the scripts that perform the automatic steps of transcription from the original xml-data so that researchers can easily make their own decisions about which items to exclude.

5 While this may seem like a lot of items to exclude, most of these are hapaxes like *bumpoopadoompadadooboom* or *doodleuhdoo*.

6 The dictionary is available at http://www.repository.voxforge1.org/downloads/SpeechCorpus/Trunk/Lexicon/VoxForge.tgz. The lexicon with which the experiments were performed was retrieved on July 19th, 2012.

7 E.g. dpseg (Goldwater et al. 2009).

If we encounter an orthographic form w for which the VoxForgeDictionary does not provide a pronunciation and which ends in either *s*, *es* or *'s*, we try to automatically construct a pronunciation as follows.

First, determine whether the *stem* $\mathtt{s}$ of the orthographic word that remains if we strip the ending $\mathtt{e} \in \{es, 's, s\}$ has a pronunciation in the dictionary. If not, we cannot automatically construct a pronunciation. Otherwise, let $\mathtt{b}$ be the pronunciation of $\mathtt{s}$.

We then construct a pronunciation $\mathtt{p}$ for w based on the following rules, depending on the identity of the ending $\mathtt{e}$:

- $\mathtt{e} = es$: $\mathtt{p} = \mathtt{b} + /@z/$
- $\mathtt{e} = s$ or $\mathtt{e} = 's$:
  - if $\mathtt{b}$ ends in a voiced segment, $\mathtt{p} = \mathtt{b} + /z/$
  - else $\mathtt{p} = \mathtt{b} + /s/$

To identify potential compounds, we try to decompose a word w into a prefix $\mathtt{p}$ and a maximal suffix $\mathtt{s}$ such that the dictionary provides pronuncations for both $\mathtt{p}$ and $\mathtt{s}$ are known forms.

Taken together, the heuristic applies to 924 cases which we then manually correct for mistakes.[8]

4.2.1.4 *Manual transcription*

The remaining 651 word types are labeled manually, using where available the form-annotation in the XML files as guide-line.[9]

4.2.2 *Statistics*

The final corpus comprises a total of 94,089 phonemically transcribed utterances and consists of six distinct sub-corpora, each corresponding to the CDS directed at one of the six children. Each sub-corpus is, in turn, subdivided into individual files corresponding to the age of the child at which recording took place, ranging from 11 up to 22 months. Both within these individual files and within the overall corpus, the order of the CDS utterances corresponds to the order in which these utterances were actually made, making them suitable for studies that look at changes over time.

Table 4.1 gives summary statistics over the full amount of data for each individual child. Looking at total number of utterances, we can broadly identify two groups: for Alex, Violet and William there are considerably less CDS utterances than for Ethan, Lily and Naima. This is presumably mostly due to recording beginning at different ages and

8 An alternative way of constructing pronunciations is to use letter-to-sound rules, a strategy that may be more appropriate for large corpora with many unknown words.

9 While not always provided for caregiver utterances, some of them include phonetic markup for individual words, in particular if the words were names.

Name	#Utt	#Tok	#Type	∅ Utt. Len.	∅ Tok. Len	∅ Type Len.
Alex	8,330	29,423	1,877	11.00	3.12	4.58
Violet	9,024	39,135	2,343	13.43	3.10	4.68
William	10,697	45,689	2,061	13.01	3.05	4.59
Ethan	18,020	75,564	2,999	13.11	3.13	4.69
Lily	20,641	94,696	3,946	14.77	3.22	4.96
Naima	27,377	141,990	4,579	16.51	3.18	5.05

Table 4.1: Statistics about the different sub-corpora of the Providence corpus, including total number of utterances, tokens, types, as well as average utterance, token and type length (in phonemes). I use ∅ as shorthand for 'average'. While roughly comparable, I focus on the Naima corpus which is the largest corpus and provides the most data for my experiments.

different numbers of sessions having been recorded for different children, as discussed above.

Yet, there also seem to be noticeable differences in terms of utterance-, token- and type-length. Although I will not do so in this paper, performing comparative evaluation of models across the children may lead to the discovery of interesting predictors of model performance and perhaps even actual language ability on behalf of the children.

For the rest of the chapter, I will focus on the Naima part of the corpus and take a closer look at how the segmentation performance of different segmentation models changes as a function of the size of the input.

4.3 BAYESIAN WORD SEGMENTATION

The word segmentation models I study in this paper are Goldwater's Unigram model (Goldwater, 2007; Goldwater et al., 2009) and Johnson (2008b)'s collocation-syllable Adaptor Grammar models. A detailed review of the mathematics for these kinds of models is provided in chapter 2. Here, I only give an intuitive idea of Bayesian word segmentation models although I will discuss, in some detail, the different collocation-syllable models below. I also want to remind the reader that I use adaptor grammars as a modeling framework because this allows us to easily specify a huge variety of models. As discussed in Chapter 2, I do not want to suggest that context-free rules are required for word segmentation; indeed, the string languages generated by the adaptor grammars I use are regular and the structured objects generated by them could also be generated by a simple probablistic branching process such as a Hidden Markov Model. Despite this, using the context-free grammar format makes both implementing, presenting and reasoning about the models much easier.

4.3.1 *Intuition for Bayesian Word Segmentation*

All the models are Bayesian probabilistic models that define a generative process for the target of learning, in this case segmented utterances or sequences of words. This generative process is defined with the help of the Dirichlet Process (DP):[10] at an intuitive level, the DP is useful for word segmentation because it biases models to identify compact ways to represent the observed unsegmented utterances, trading off the number of both tokens and types used in an analysis of the data.

This trade-off is a consequence of the way that probabilities are assigned to tokens under a DP model: the probability of hypothesizing a word token[11] depends on the number of times that its type has previously been hypothesized, and the probability of a full segmentation of the data is the product of the probabilities for all the tokens used in the segmentation. This tends to make solutions in which a small number of words is used relatively frequently yet not over-excessively (as would be the case if every individual segment were a type) the most probable which, in most cases, also leads to linguistically reasonable results. What differentiates the different models from one-another are the specific assumptions about *the nature of possible word types* and *the relationships between word tokens*. I will elaborate on these points, thus introducing all models used in the experiments.

4.3.2 *Assumptions about possible words*

The base distribution which specifies a model's prior expectations about the form of words plays is a crucial part of any Bayesian segmentation model. Here I compare how different assumptions about the base distribution interact with the assumptions the model makes about word-to-word dependencies.

A naive assumption about possible words is that they can be any arbitrary sequence of segments. Under such a Unigram phoneme distribution (see also Figure 2.4) both *dog* and *qfx* would be equally good candidates for possible words a priori. While obviously not true of human languages (*bnik* isn't a possible English word), this was the base distribution built into the original Unigram and Bigram models (Goldwater et al., 2009) that initiated research on Bayesian word segmentation. While it has been shown to work reasonably well on in specific settings, we already saw in Chapter 2 and, considering incremental inference, in Chapter 3 that without additional constraints this kind of base distribution can lead to surprisingly bad performance. Staying close to

10 Adaptor Grammars actually use the Pitman-Yor Process, a strict generalization of the DP. I gloss over this detail.

11 Or a token of another entity, e.g. a syllable or a multi-word expression, if the model incorporates these notions.

the original models, I showed how adding a possible word constraint along the lines of Norris et al. (1997) can partially address these issues.

Extending the idea of such a possible word constraint in a linguistically motivated way and utilizing Adaptor Grammar's ability to easily specify rich hierarchical models, I now consider the even more constrained assumption that words have to be sequences of syllables. Indeed, there is strong evidence that even very young infants track probabilities defined over syllables (Saffran et al., 1996), and recent work such as Phillips and Pearl (in press) argues that the syllable has advantages over the phoneme as a primitive unit in word segmentation.

4.3.2.1 *Phonemic or syllabic input*

While this may be viewed as arguing for using syllabified rather than phonemic input, Schrimpf and Jarosz (2014) argue that proper syllabification presupposes successful word segmentation. Their example is that a phoneme sequence such as /l ʊ k æt/ ("lookat") can only be correctly syllabified as /l ʊ k - æt/ if /l ʊ k/ has already been identified as a word; otherwise, syllabification principles such as *onset maximization* (see, e.g. Hayes, 2009, p.) will syllabify it as /l ʊ - k æt/.

I share Schrimpf and Jarosz (2014)'s feeling that this suggests that syllabification and word segmentation need to be performed *jointly* rather than separately and use phonemic input and models that can perform joint segmentation and syllabification.

4.3.2.2 *Unconstrained base distribution*

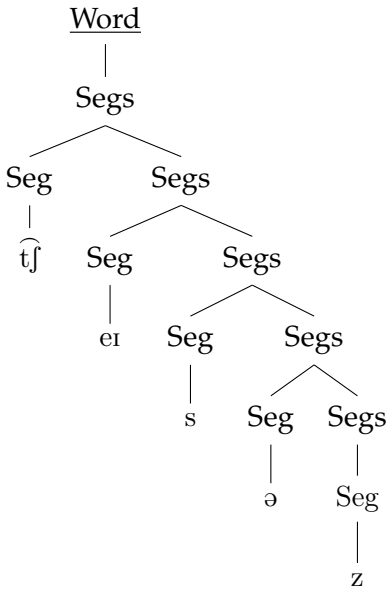
Chapter 2 described an extension of the unconstrained base distribution of Goldwater (2007) and Goldwater et al. (2009) as a probabilistic finite state automaton. Here, I re-express the same distribution as a Probabilistic Context-Free Grammar. This can be done concisely using the three rules in Figure 4.1.

Rule 4.1 defines that words are arbitrary sequences of segments as generated by rules 4.2-4.4. Here, rule 4.4 is actually a rule schemata which abbreviates an entire set of re-write rules, one for every phoneme $x \in \Sigma$ where Σ is the phoneme inventory of the data.

Word is underlined, indicating that it is an *adapted non-terminal* (see Chapter 2). Recall that this means that the model will learn an inventory of and a distribution over complete subtrees dominated by a Word non-terminals which, essentially, corresponds to the lexicon inferred by the segmentation model.

As not every suffix of a word is itself another word, we want to keep separate the non-terminal that implements the recursion, Segs, from the adapted non-terminal that enables the model to learn words, Word, explaining why we need the unary rule 4.1.[12]

12 There also is the issue that it is not clear what recursion involving adapted non-terminal means and ought to be avoided for technical reasons, see Johnson (2007).



$$\underline{\text{Word}} \rightarrow \text{Segs} \quad (4.1)$$
$$\text{Segs} \rightarrow \text{Seg} \quad (4.2)$$
$$\text{Segs} \rightarrow \text{Seg Segs} \quad (4.3)$$
$$\text{Seg} \rightarrow \text{x} \quad (4.4)$$

Figure 4.1: Adaptor Grammar rules to define an unconstrained base distribution over all possible sequences of segments. An example tree generated by these rules is depicted on the left.

The distribution over sequences of phonemes defined by these rules is equivalent to that defined by the automaton in Figure 2.4. In particular, the probabilities associated with the rules corresponding to 4.4 directly correspond to Θ, the probabilities governing transitions from state 0 to state 1; and the probability of rule 4.2 corresponds to Φ, the stopping probability with which the automaton transitions into its final state.

4.3.2.3 *Syllable base distribution*

We can put constraints on possible words by requiring words to be sequences of syllables rather than arbitrary sequences of segments. The rules in Figure 4.2 define a base distribution that enforces this constraint which is, essentially, that of Johnson (2008b). I use brackets to indicate optionality of a category on the right hand side, allowing us to compactly present multiple alternatives. For example, rule 4.5 abbreviates the 2 rules

$$\text{SyllIF} \rightarrow \underline{\text{OnsI}}\ \text{Rhyme}$$
$$\text{SyllIF} \rightarrow \text{Rhyme}$$

Also note that I explicitly limit the length of possible words to 4 syllables, following Johnson (2008b) and Johnson and Goldwater (2009). I do this for reasons of efficiency as avoiding a recursive rule at this level greatly speeds up the parsing that is required by the adaptor grammar inference program. In preliminary experiments I found no noticeable

differences between using a fully recursive alternative in which words could consist of any number of syllables extend for noticeable increased runtime. This is not very surprising, considering that most words in English child directed speech tend to be mono-syllabic.

To explain the base distribution these rules define in more detail, I now discuss the assumptions about syllable structure they encode which correspond to standard assumptions made in phonology (Hayes, 2009, e.g.). To begin with, it requires each syllable to contain a vowel, as syllables need to have a vocalic core also called *nucleus*.[13] This requires us to pre-specify the set of phonemes which are vowels $\mathscr{V}$ and the set of phonemes which are consonants $\mathscr{C}$.

Preliminary experiments showed that it is also possible to infer which phonemes are which although, as vocalic and consonantal segments differ considerably in their acoustic properties (Ladefoged, 2012), I believe that assuming this difference is unproblematic.[14]

Optionally, the obligatory vowel which each syllable needs to contain can be preceded by an optional sequence of consonants called *onset* and followed by an optional sequence of consonants called *coda*. Nucleus and coda are grouped into a constituent called *rhyme*.

Thus, all rules defining syllables conform to the general schema

$$\begin{aligned} \text{Syll} &\rightarrow (\,\text{Ons}\,)\,\text{Rhyme} \\ \text{Rhyme} &\rightarrow \text{V Coda} \end{aligned}$$

The specific rules, taken from Johnson and Goldwater (2009), make additional assumptions in order to allow the model to notice aspects of syllable structure that are particular to an individual language and can help in word segmentation.

For example, in order to allow it to learn which sequences of consonants are likely to occur in onsets and codas, I treat both onsets and codas as adapted non-terminals. This makes it possible for the model to capture aspects that are specific to individual languages – for example, that onsets such as /sp/ are dispreferred in Spanish but are fine in English – from the data, relying only on (arguably universal) general knowledge about syllable structure.

Moreover, I allow the model to learn certain aspects of the phonotactics of the language by identifying which specific onsets are limited

13 Because diphtongs are treated as single segments, there is no need to allow a sequence of vowels in nucleus position. In fact, this is also true for the data on which Johnson (2008b) evaluated, rendering his mention of such a rule superfluous.

14 A potential complication is that in languages such as Tamil Berber, consonants may play the role of syllabic nuclei in particular contexts but not others. To some extent, English exhibits a similar phenomenon in words such as /b ɑ t l̩/ ("bottle") which may be analyzed as containing a syllabic /l̩/. Here, I ignore these issues and make the simplifying assumption that the role each segment can play is specified ahead of time (although it is possible for phonemes to count as both consonantal and syllabic, allowing the model to choose the analysis of each token depending on the context).

to or strongly indicative of the beginning of words; and which codas are limited to or strongly indicative of the end of words. For example, a complex onset such as /s t r/ as in /s t r ɛ ŋ θ/ ("strength") occurs most frequently word-initially, and the coda /ŋ θ/ occurs almost exclusively word-finally.

To enable the model to exploit these kinds of phonotactic cues, two kinds of onsets and codas are distinguished, OnsI for word-initial onsets, Ons for word-internal onsets, CodaF for word final codas, and Coda for word-internal codas. Crucially, these different non-terminals are adapted as, otherwise, the model would not be able to learn the preferred role of entire sequences of consonants.[15]

As a result of the two way distinction of onsets and codas, we have to distinguish 4 syllable non-terminals. SyllIF corresponds to the single syllable in a mono-syllabic word in which the (optional) onset has to be OnsI and the (optional) coda has to be CodaF because the same syllable is both initial and final. SyllI is a word-initial syllable in a multi-syllabic word, thus its onset (if any) has to be a OnsI but its coda (if any) a word-internal coda Coda. Similarly, SyllF corresponds to the word-final syllable in a multi-syllabic word which uses Ons and CodaF. Finally, Syll corresponds to a word-internal syllable and (optionally) uses both Ons and Coda.[16]

4.3.3 *Assumptions about relations between word tokens*

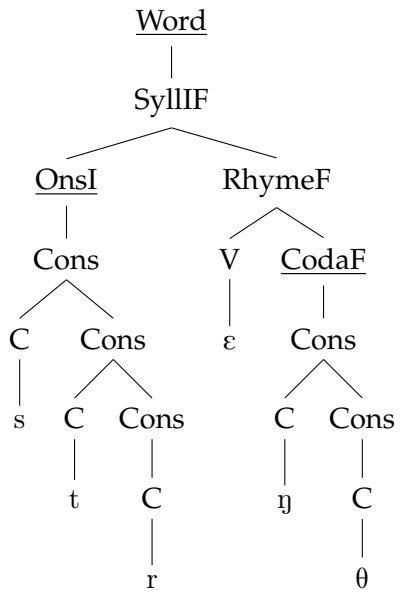
So far, I have only discussed the base distribution. Another aspect with respect to which one can distinguish different models are the distributional assumptions they encode about the words in an utterance.

The simplest such assumption is, arguably, that all words are independent, resulting in a 0^{th}-order Markov or Unigram model in which the probability of a sequence of words is just the product of the marginal probability of each word. As a result, the probability of "*the dog barks*" is indistinguishable from "barks dog the", showing that these assumptions clearly do not suffice to capture semantic or syntactic relations between words. Although popular in early work on segmentation (Brent, 1999; Venkataraman, 2001; Goldwater, 2007), this Unigram assumption has been shown to also be problematic for word segmentation. In particular, Goldwater (2007) and Goldwater et al. (2009) demonstrated that the Unigram assumption encourages undersegmentation, an issue I also noticed in the discussion in the previous chapter.

The Unigram model can be expressed as an Adaptor Grammar (Johnson, 2007), using the following rules which generate sequences of rules

15 An investigation into how much these particular cues help is provided in Chapter 5.

16 Note that unlike onsets and codas, syllables themselves are not adapted. In Börschinger et al. (2012), I discuss a variant of the model in which entire syllables are adapted. Subsequent experiments indicate that this does not make a difference to the overall findings but that adapting syllables leads to even more severe undersegmentation behavior.



$$
\begin{aligned}
\underline{\text{Word}} &\rightarrow \text{SyllIF} \\
\underline{\text{Word}} &\rightarrow \text{SyllI}\,(\,\text{Syll}\,)(\,\text{Syll}\,)\,\text{SyllF} \\
\text{SyllIF} &\rightarrow (\,\underline{\text{OnsI}}\,)\,\text{RhymeF} \qquad (4.5) \\
\text{SyllI} &\rightarrow (\,\underline{\text{Ons}}\,)\,\text{Rhyme} \\
\text{SyllF} &\rightarrow (\,\underline{\text{Ons}}\,)\,\text{RhymeF} \\
\underline{\text{OnsI}} &\rightarrow \text{Cons} \\
\underline{\text{Ons}} &\rightarrow \text{Cons} \\
\underline{\text{CodaF}} &\rightarrow \text{Cons} \\
\underline{\text{Coda}} &\rightarrow \text{Cons} \\
\text{Cons} &\rightarrow \text{C}\,(\,\text{Cons}\,) \\
\text{RhymeF} &\rightarrow \text{V}\,(\,\underline{\text{CodaF}}\,) \\
\text{RhymeF} &\rightarrow \text{V}\,(\,\underline{\text{Coda}}\,) \\
\text{C} &\rightarrow \mathrm{c}\,, \mathrm{c} \in \mathscr{C} \\
\text{V} &\rightarrow \mathrm{v}\,, \mathrm{v} \in \mathscr{V}
\end{aligned}
$$

Figure 4.2: Adaptor Grammar rules to define a base distribution that constrains words to be sequences of syllables. In addition, this grammar enables the model to use phonotactic cues to word-boundaries by learning word-initial onsets ($\underline{\text{OnsI}}$) and word-final codas ($\underline{\text{CodaF}}$). An example tree is given to the left.

which generate sequences of adapted $\underline{\text{Word}}$ non-terminals using a simple 0^{th}-order Markov process:

$$
\begin{aligned}
\text{Words} &\rightarrow \underline{\text{Word}} \\
\text{Words} &\rightarrow \underline{\text{Word}}\ \text{Words}
\end{aligned}
$$

The kind of structure generated by these rules is illustrated in Figure 4.3. By using either the rules in Figure 4.1 or those in Figure 4.2 to expand the $\underline{\text{Word}}$ non-terminal, one recovers the UNIGRAM model described in Chapter 2 with the unconstrained base distribution of Figure 2.4, or a UNIGRAM-SYLLABLE model which makes use of the more constrained base distribution described above.

One way in which Goldwater (2007) proposed to address this is by moving from a Unigram to a Bigram model. In such a model, the probability of a sequence such as "the dog barks" is the product of the probabilities of the *bigrams*, i.e. all pairs of adjacent words. How this can be done is reviewed in detail in Chapter 2, and although under certain circumstances this results in noticeably improvements over the Unigram assumption, increasing the order of the language model even further showed no noticeable gains. In particular, Mochihashi et al. (2009) considered a *Trigram* model in which the probability of a sequence is the

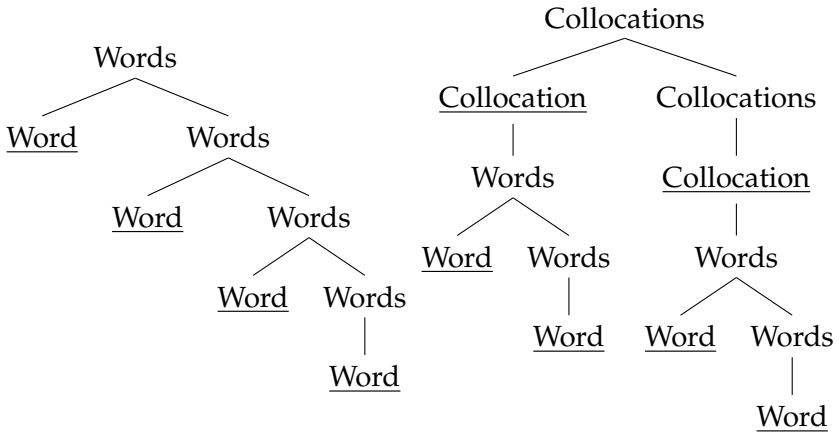


Figure 4.3: Illustration of the kinds of structures that the Unigram and the Collocation model generate for an utterance.

product of the probabilities of every triple of adjacent words in an utterance and reported, essentially, the same performance as the Bigram model.

Interestingly, the Bigram model *cannot* be expressed as an Adaptor Grammar, at least in the current instantiation of Adaptor Grammars that assume a finite set of non-terminal symbols. This is because unlike the 0^{th}-order Markov process of the Unigram model, the 1^{st}-order Markov Process of the Bigram model would need to be described using a particular adapted non-terminal for every possible word (also see the discussion at the end of Chapter 2). Thus, there is no straightforward way in which the syllable base distribution of Figure 4.2 can be combined with a Bigram model – while it is, theoretically, possible to re-code this base distribution, the rich hierarchy it includes and the latent sub-word structure make this challenging.

An alternative way of relaxing the independence assumption between words has been proposed by Johnson (2008b), employing a hierarchical instead of a sequential notion of context. His *collocation* model assume that sentences are sequences of *multi-word sequences* or collocations. Consequently, the model not only learns words but an additional kind of entity, entire chunks of words. Importantly and in contrast to the Unigram and Bigram models, these chunks are stored in addition to and not at the expense of the words that make them up — a collocation model can infer that /ð ə d o g i/ (*thedoggie*) is a high-frequent sequence that is made up of the distinct words /ð ə/ (*the*) and /d o g i/ (*doggie*).

The COLLOCATION model can be 'derived' from the Unigram model by adding the rules

$$\text{Collocations} \rightarrow \underline{\text{Collocation}}\ (\ \text{Collocations}\ )$$
$$\underline{\text{Collocation}} \rightarrow \text{Words}$$

$$
\begin{aligned}
\text{Collocation3s} &\rightarrow \underline{\text{Collocation3}}\ (\ \text{Collocation3s}\ )\\
\underline{\text{Collocation3}} &\rightarrow \text{Collocation2s}\\
\text{Collocation2s} &\rightarrow \underline{\text{Collocation2}}\ (\ \text{Collocation2s}\ )\\
\underline{\text{Collocation2}} &\rightarrow \text{Collocations}\\
\text{Collocations} &\rightarrow \underline{\text{Collocation}}\ (\ \text{Collocations}\ )\\
\underline{\text{Collocation}} &\rightarrow \text{Words}\\
\text{Words} &\rightarrow \underline{\text{Word}}\ (\ \text{Words}\ )
\end{aligned}
$$

Figure 4.4: Adaptor grammar rules for a collocation3-model. To expand the adapted Word non-terminal, either the rules in Figure 4.1 or those in Figure 4.2 can be used. The former yields the COLLOC3 model, the latter the COLLOC3-SYLL model.

and letting Collocations be the start-symbol of the Adaptor Grammar. Thus, just like the Unigram model one can combine the collocation model with the different assumptions about words discussed above. The kind of structures generated by the collocation model is illustrated in Figure 4.3 – note that the expansion of each Collocation uses the kind of structure the Unigram model uses to analyze an entire sentence.

Adaptor Grammars make it easy to further extend the collocation model by adding additional levels above the Collocations non-terminal. Just as one derives the collocation model from the Unigram model by adapting the non-terminal that spans the whole sentence in the Unigram model, one can derive a collocation2 model from the collocation model by adding the following rules and using as start symbol the Collocation2s non-terminal:

$$
\begin{aligned}
\text{Collocation2s} &\rightarrow \underline{\text{Collocation2}}\ (\ \text{Collocation2s}\ )\\
\underline{\text{Collocation2}} &\rightarrow \underline{\text{Collocations}}
\end{aligned}
$$

Johnson and Goldwater (2009) found that using yet another level, i.e. a collocation3-model, yields best performance. Following this observation, in this chapter I examine models using up to 3 levels of collocations, and I consider for each model whether or not it assumes an unconstrained (Figure 4.1) or a syllable-constrained (Figure 4.2) base distribution. For concreteness, the grammar for a collocation3-model is given in Figure 4.4 down to the adapted Word non-terminal which can either be expanded using the rules in Figure 4.1 or those in Figure 4.2.

I refer to the Unigram model with unconstrained base distribution simply as UNIGRAM, and analogously, to the collocation models with unconstrained base distribution as COLLOC, COLLOC2, and COLLOC3. To distinguish these models from those with the syllable base distribution,

I refer to the latter as UNIGRAM-SYLL, COLLOC-SYLL, COLLOC2-SYLL, and COLLOC3-SYLL, for a total of 8 models.[17]

4.4 EXPERIMENTS

As human language learners tend to get better at their native language with longer exposure, one would expect adequate computational models to exhibit something similar, initially improving as more data is observed and, at some point (probably beyond the size of samples one usually can look at in practice), asymptotically approaching some upper bound.

Also, I want to address one of the questions raised in the previous chapters by the bad performance of the Bigram model on small amounts of data, namely whether more complex models require stronger inductive biases (as encoded in the syllable base distribution) to perform well on smaller amounts of data.

4.4.1 *Corpus*

The longitudinal data available in the Providence Corpus suggests a natural setup for studying these questions by constructing inputs that consist of all CDS utterances directed at an individual child up to a certain point of time. For my experiments, I use the Naima section of the Providence Corpus and collect CDS utterances from when Naima was 11 months old through to when she was 21 months old to construct 11 differently sized inputs, each input consisting of all CDS utterances in the corpus up to and including a given month.

I refer to the different input sets by the last month from which it includes data. For example, data set 11 includes all utterances in the Naima corpus which were collected up to her 11^{th} month; data set 12 will add to this corpus the utterances collected ruing Naima's 12^{th} month, and so forth. Thus, another way of viewing the different datasets is as larger and larger prefixes of the entire Naima corpus. Table 4.2 presents high-level summary statistics for the different input sets.

For ease of discussion, I use "language exposure" and "input size" exchangeably, a simplifying yet justified choice as is evident from Table 4.2 that shows how the number of utterances grows over time.

17 Börschinger (2012) includes a discussion of the original Bigram model. As one cannot combine it with the syllable base distribution and increasing the order of the Markov language model has not shown to yield improvements (Mochihashi et al., 2009), I exclude it from discussion here.

Month	Utterances	Tokens	Types	Avg. Utt. Len
11	973	3,756	589	12.40
12	2,683	10,932	1,086	12.83
13	3,553	14,640	1,275	12.96
14	4,794	20,466	1,612	13.47
15	7,533	33,283	2,073	13.97
16	11,338	52,573	2,566	14.67
17	13,435	63,603	2,864	15.03
18	15,990	78,023	3,250	15.49
19	18,984	94,797	3,666	15.88
20	21,887	111,322	3,978	16.15
21	24,327	124,341	4,214	16.24

Table 4.2: The properties of the different input sets. Note that each later month properly includes all utterances from the earlier months, e.g the 2683 utterances of the input at month 12 include the 973 utterances of the input at month 11, and so forth.

4.4.2 *Evaluation*

4.4.2.1 *Evaluation data*

Word segmentation is an instance of *unsupervised* learning and as such, it is common to simply evaluate on the input that model performed inference over. However, this does not allow us to systematically compare performance of models across largely different inputs. To illustrate, performance on a very large corpus may – when summarized by a single score calculated with respect to this corpus – be worse than that on a smaller corpus (again, summarized by a single score calculated with respect to this smaller corpus) simply because the two corpora vary considerably.

To address this, I also evaluate on a held-out test-set which is constant across all experimental conditions, irrespective of the input. The scores on this held-out data can be compared directly across conditions, making a comparison of how overall segmentation quality changes over time meaningful.

I construct the test-set by sampling 200 CDS utterances from the 22nd month of each of the six children's sub-corpus in the Providence Corpus, for a total of 1,200 held-out utterances.[18]

The segmentation on the held-out data is calculated after inference has been performed on the input, thus implicitly defining a (probabilistic) lexicon according to which one samples a segmentation for each

18 Including CDS utterances directed at other children also introduces some amount of variation so that one can ensure that benefits of more complex models are not simply due to over-fitting the peculiarities of the Naima corpus.

utterance in the test-set. Note that during this process, no novel words are added to the model's lexicon; in this sense, we evaluate the knowledge the learner has acquired after having had access to the input.

4.4.2.2 *Evaluation metrics*

As evaluation metrics, I use the standard token and boundary scores going back to Brent and Cartwright (1996) and reviewed in Chapter 2. The single most representative score for overall segmentation performance is token f-score, the harmonic mean between token precision (number of correct tokens identified by the model over the number of total tokens predicted by the model) and token recall (number of correct tokens identified by the model over the true number of tokens in the input) as a measure of segmentation accuracy.

4.4.2.3 *Experimental setup*

My experimental procedure follows closely the one outlined in Johnson and Goldwater (2009). I used the August 2012 version of Mark Johnson's Adaptor Grammar implementation[19] to run two Markov Chain Monte Carlo chains for each of the models for 1000 iterations, collecting sample analyses for the held-out test-set with a lag of 5 after a burn-in of 800 iterations.

I then determine the maximum marginal posterior segmentation (Johnson and Goldwater (2009), see Chapter 2 for a review) for each individual utterance on the basis of the 80 samples that were collected for each condition.

An overall summary of the experimental results is given in Figure 4.5 which plots token f-score on both the input and the held-out test-set for different amounts of input size, with the size of the input getting larger from left to right. I indicate the base distribution used by each model by the line-type, using solid lines for the syllable base distribution and dashed lines for the unconstrained Unigram phoneme base distribution. The different (in)dependence assumptions about words are indicated by color.

4.5 DISCUSSION

Overall, one can see two broad patterns of behavior across the different models which are largely consistent for evaluating on the input and on the held-out data. One group of models exhibits *a degradation in performance for larger amounts of inputs*, in particular the UNIGRAM, UNIGRAM-SYLL, COLLOC-SYLL; and to a lesser degree the COLLOC2-SYLL and the COLLOC3-SYLL. The other group comprises the collocation

19 Available at http://web.science.mq.edu.au/~mjohnson/code/py-cfg-2012-08-16.tgz.

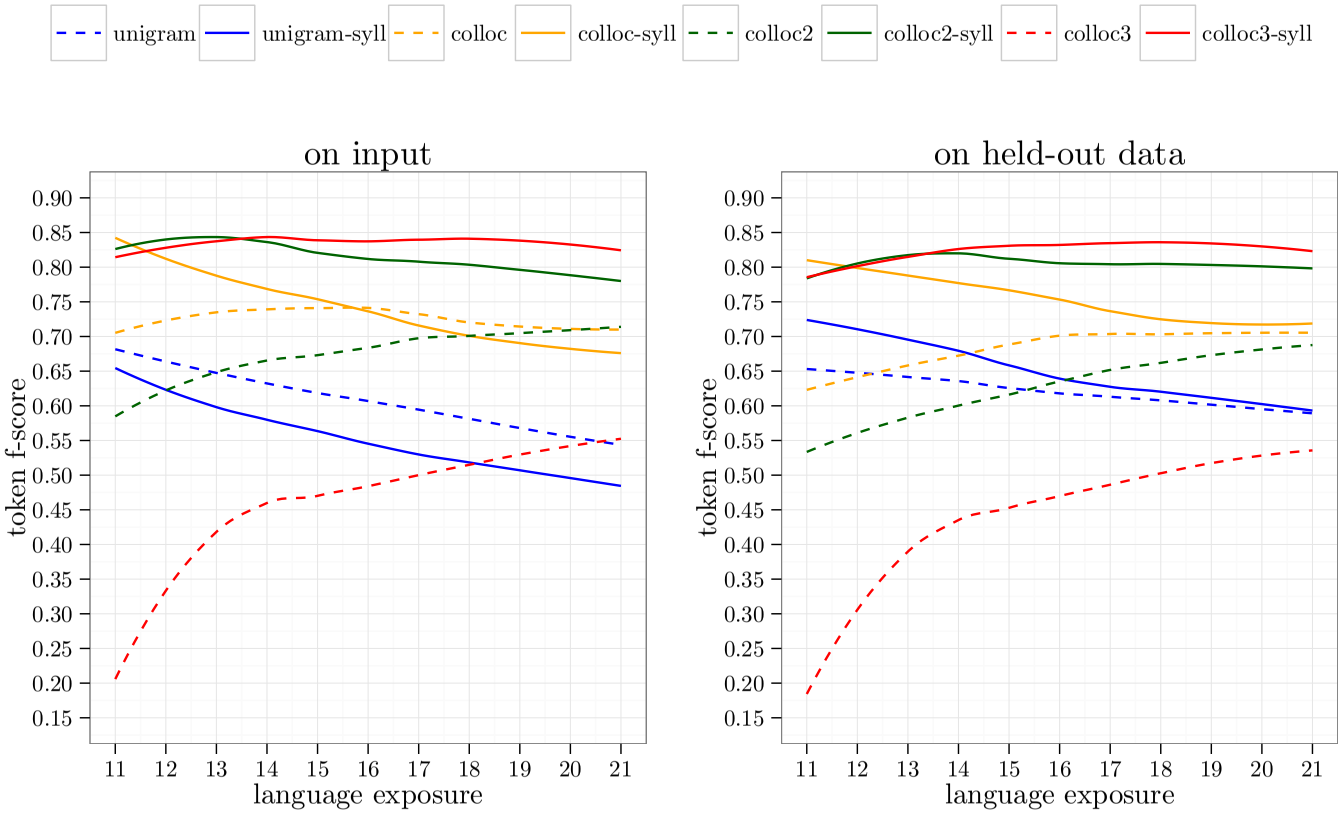


Figure 4.5: Word segmentation performance on input and held-out test-set for the different models, as a function of the input size. The models incorporating syllable structure all exhibit some degree of performance decrease for larger inputs although the COLLOC3-SYLL model consistently remains well above 80%. In contrast, the models without syllable structure exhibit an increase in performance over time although their segmentation accuracy is considerably worse than that of the best performing models with syllable-structures. Performance is very similar on both input and held-out data although, somewhat surprisingly perhaps, the downwards trend is more pronounced on the input. See text for discussion.

models with no syllable structure, i.e. COLLOC, COLLOC2, and COLLOC3, show signs of improvement over time.

4.5.1 *Differences between evaluation on input and held-out data*

Overall, the trends as well as segmentation performance are by and large identical on both the input and the held-out data; because, as I have argued before, it is not clear how a token f-score for a corpus of 900 utterances ought to be compared to one for a corpus of 25,000 utterances, I focus the discussion on the held-out data where all the scores are calculated across the same utterances, irrespective of the input size.

Yet, it is interesting to note that a model that has seen as little as roughly 1,000 to 2,500 utterances (which corresponds to months 11 and 12) performs almost as well on its input as on held-out data – thus, there is no evidence for 'over-fitting' in the classical sense that good performance on the 'training data' entails considerably worse performance on held-out data.

4.5.2 *Performance of different models*

The COLLOC3-SYLL model clearly emerges as the most accurate with a token f-score of 84% at peak performance at around 18 months (15,990 utterances) that drops to around 82% at month 21 (24,327 utterances). Second best is the COLLOC2-SYLL model which peaks at around 82% at month 13 and also drops by 2% to about 80% at month 21. The third place is taken by COLLOC-SYLL which peaks already at month 11 with 81% token f-score. However, it exhibits a much more dramatic drop than the higher-order collocation models, steadily dropping in token f-score by almost 10% to 72% at month 21.

Determining a clear fourth place is harder. Thus, even though up until roughly 14 months the UNIGRAM-SYLL model is outperformed only by the higher-order collocation syllable models, at this point it is overtaken by COLLOC, the simplest collocation model that does not use syllable structure; and at 16 months, by COLLOC2. This is because UNIGRAM-SYLL exhibits the most dramatic drop in performance, from around 72% at month 11 (973 utterances) to slightly less than 60% for month 21; in contrast, even though COLLOC starts out with only 60% at onth 11, its performance increases over time and seems to asymptote to slightly above 70% starting from month 16.

The largest performance improvement is seen for COLLOC3 which jumps from only 18% for month 11 to about 53% for month 21. Despite this, even after this huge increase it remains the worst model.

To sum up, there are two types of behaviour. UNIGRAM, COLLOC-SYLL, COLLOC2-SYLL and COLLOC3-SYLL exhibit what I call *overlearning*: they reach their peak performance for relatively small amounts of

input and gradually get worse as the size of the input grows larger. This is much more pronounced for UNIGRAM-SYLL and COLLOC-SYLL than for COLLOC2-SYLL and COLLOC3-SYLL, with the latter remaining above 80% accuracy even for the largest amount of input. Despite overlearning, COLLOC3-SYLL and COLLOC2-SYLL perform word segmentation the most accurate for all sizes of input. On the other hand, the collocation models lacking the assumption of syllable structure do not exhibit overlearning, at least not on the amount of data I tested them.

To my knowledge, I am the first to identify these two different trends which I will now attempt to explain.

4.5.3 *Overlearning*

I begin with a detailed explanation of "overlearning", starting from an original observation going back to Goldwater (2007) who noticed that the Unigram model tends to identify undersegmented solutions where the predicted (incorrect) words often consist of several of the (correct) words. Her explanation for this is that "groups of words that frequently co-occur violate the unigram assumption in the model since they exhibit strong word-to-word dependencies", and that "[t]he only way the model can capture these dependencies is by assuming that these collocations are in fact words themselves." (Goldwater, 2007, p.72)

Why is it, however, that these "misleading" co-occurrences occur in the data in the first place and, apparently, become more problematic the larger the input is?

4.5.3.1 *Explaining overlearning*

I suspect that many of the "collocations" a model such as UNIGRAM is susceptible to arise from *principled regularities governing language* which are not accounted for by the model. I discuss this idea first for the UNIGRAM model and then argue that it extends to the collocation-syllable models as well.

As a concrete example, note that English syntax requires (most) prepositional phrases to begin with a preposition-determiner sequence. As both prepositions and determiners are a small closed class, there is only a small number of sequences such as "in the" or "of a". Crucially, the occurrence of a sequence such as *in the* is largely independent of what is actually being talked about (as it excludes the head-noun of the prepositional phrase) and the number of its occurrences can therefore be expected to just keep growing with the amount of input.

This is supported by figure 4.6 which plots the change in relative frequency of several function word bigrams and a 'content' bigram such as *the baby* which is, indeed, the highest frequency bigram including a noun. Whereas occurrences of the content bigram are restricted to contexts in which a baby plays any role, the function word bigram are necessi-

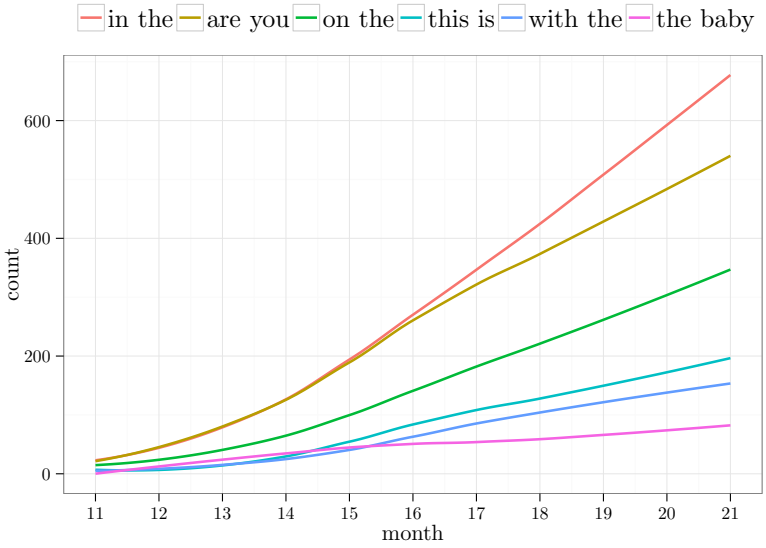


Figure 4.6: Frequency of several high frequency function word bigrams over time as well as that of the highest-frequency content word bigram *the baby*. See Table 4.3 for further discussion of these patterns.

tated not by conversation topic but English syntax. Consequently, their frequency in the input grows much more dramatically over time.

Recalling that the UNIGRAM model was shown to undersegment high-frequency items, this observation directly leads to the prediction that it will perform worse when trained on larger amounts of data: the evidence for spurious 'words' such *in the* grows steadily with the input size.

The experimental results strongly suggest that this reasoning is correct. The drop in performance for the UNIGRAM model is clear from figure 1: it reaches peak performance of around 66% when its input consists of a mere 973 utterances, and its segmentation accuracy steadily drops as it processes larger inputs down to around 59% for an input of 24,327 utterances. When syllable structure is taken into account, the drop is even more pronounced as it starts from 73% and also drops to around 59%.

More direct support for explaining the drop by the negative impact of the increasing frequency of patterns like the one in figure 4.6 comes from figure 4.7 that plots how well the model is able to identify word types of different frequencies in the test set as a function of input size.

Note how for higher-frequency types, the Unigram model's performance decreases more dramatically than for low frequency types for larger amounts of input. To investigate whether this difference in performance could actually be due to high-frequency items getting "absorbed" into larger units as I suggest following Goldwater (2007), in Table 4.3 I perform a qualitative evaluation of a number of actual examples of patterns involving high-frequency items that are themselves of different frequencies. As is clear, cases that are analyzed correctly at month 11 are almost consistently misanalysed as a single word at month 21, showing that the "loss" of high-frequency items is a major reason for the overlearning.

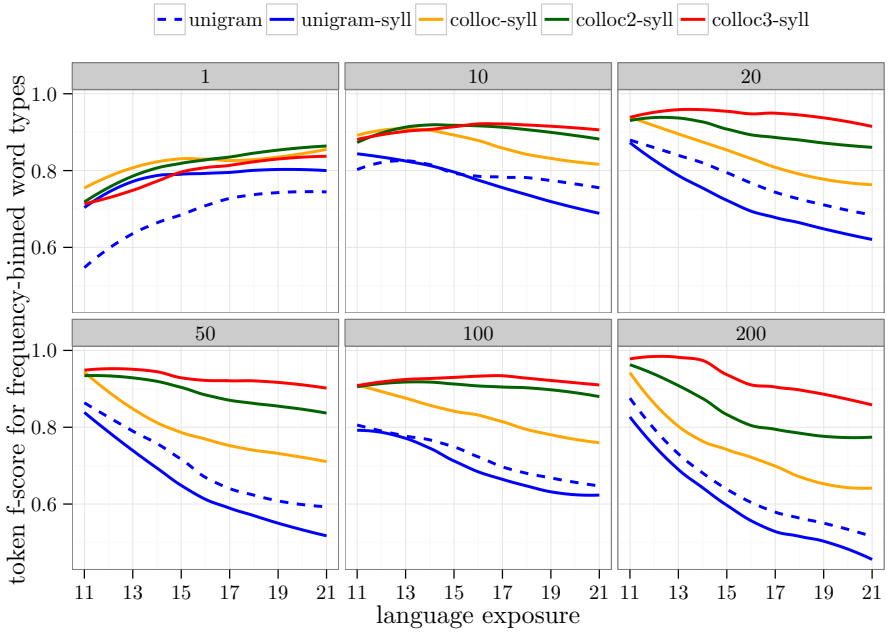


Figure 4.7: Token f-scores for word types of different frequencies in the test-set as a function of the size of the input. Note that the Unigram and the collocSyll model already show clear decreases in accuracy over time for types of frequency larger than 10 whereas the colloc3Syll model shows a relatively robust performance up to the highest-frequency bin of types with frequency larger than 200. The drop in performance shows that even the colloc3Syll model suffers from "overlearning" although this behavior is much less pronounced than for the other models and only occurs dramatically for the highest-frequency types in the data.

Surprisingly, perhaps, the same kind of explanation seems to apply to the collocation-syllable models.

While originally proposed by Johnson (2008b) to specifically address the problem of undersegmentation, looking at figure 4.7 indicates that collocation models do not solve the problem of high-frequency words completely, although it seems to get less problematic with the number of collocational levels the model has at its disposal. Looking at the kind of units learned by the collocation-models explains why that is: among the high-frequency collocations learned by both the COLLOC-SYLL and the COLLOC2-SYLL model from the largest input is, for example, the two 'word' sequence *doyou remember*.

While this is a better solution than the UNIGRAM model's *doyouremember*, it still involves the undersegmented collocation *do you* which, incidentally, is another example for a high-frequency function word bigram. The COLLOC3-SYLL model analyses *this specific case correctly* as a three-word collocation *do you remember*, but it fails to acquire the

pattern	#test	month 11 (973 utterances)					month 21 (24,327 utterances)				
		#input	UNIGRAM		COLLOC3-SYLL		#input	UNIGRAM		COLLOC3-SYLL	
			% cor	% col	% cor	% col		% cor	% col	% cor	% col
in the	18	14	50%	50%	94%	0%	671	0%	89%	0%	94%
are you	19	14	100%	0%	100%	0%	536	0%	71%	47%	42%
on the	21	11	100%	0%	86%	0%	347	0%	86%	10%	81%
this is	9	5	100%	0%	100%	0%	196	0%	67%	100%	0%
with the	4	2	100%	0%	100%	0%	153	0%	75%	100%	0%
the baby	4	0	100%	0%	75%	0%	80	0%	100%	100%	0%

Table 4.3: A qualitative look at how bigram-patterns of different frequency are handled by the UNIGRAM and the COLLOC3-SYLL model at month 11 and month 21, respectively. The %cor and %col columns give the percentage of occurrences of a pattern in the test-set that were handled correctly or were misanalysed as constituting a two-word collocation. Note that %cor and %col need not add up to 100% as the models can also make other errors than simply undersegmeting. At month 11, both the UNIGRAM and the COLLOC3-SYLL model handle all the cases nearly perfectly; but unlike the UNIGRAM model, the COLLOC3-SYLL model handles lower-frequency patterns at month 21 with increasing ease, making no mistake for patterns that occur fewer than 200 times in its input such as *the baby* but still mis-analyzing a pattern such as *in the*.

collocation *do you* on its own and prefers to use the "word" *doyou* in most other cases.

This shows that collocations do not solve the undersegmentation problem but only push it back a level, in line with figure 4.7: while the COLLOC3-SYLL model behaves relatively stable for word-types with frequencies smaller than 200, it also shows a marked drop for the high-frequency word types from around 95% at month 11 to just below 80% for month 21. Some concrete examples of patterns which the COLLOC3-SYLL model is and isn't able to handle correctly are given in table 4.3, alongside the performance of the Unigram model for these cases. This clearly illustrates how, even though COLLOC3-SYLL handles some of the patterns which UNIGRAM misanalyzes correctly, its ability to handle word-dependencies breaks for high-frequency patterns.

4.5.3.2 *A general problem for Bayesian models?*

With this, the fact that even for collocation-models the undersegmentation problem gets worse for larger inputs shouldn't be too surprising. As pointed out above, many of the patterns leading to undersegmentation errors are due to syntactic regularities that, for example, require prepositions to be followed (in almost all cases) by articles. Figure 4.6 indicates that these kinds of patterns grow continuously with the size of the input, suggesting that models that "merely" model co-occurrence statistics are bound to fail at some point.

This may almost seem like a general problem for Bayesian probabilistic models of the kind discussed here that, in a sense, simply try to identify high-frequency patterns in the input. Yet this is not so.

For one thing, the lack of detailed linguistic structure is not inherent to the Bayesian framework that is fully unrestricted as to what kind of structures a model is defined over. This much is clear from the ease with which syllable structure can be incorporated into the models.

Secondly, even without additional linguistic structure the relative robustness of the COLLOC3-SYLL model shows that while not fully solving the problem of misleading co-occurrences, a sufficiently rich collocational structure goes a long way in alleviating the problem for input sizes that go well beyond 20,000 utterances. It suffices to handle most cases involving content words such as nouns, correctly learning for example that despite its (relatively) high frequency, *the baby* consists of two individual words. Figure 4.6 shows that for patterns like this, frequency grows much slower over time (although still too fast for a model lacking any ability to model larger-than-word-units such as the UNIGRAM model), not too surprising considering that the occurrence of content words — unlike function words — is mainly dependent on what is actually being talked about and that conversation topics tend to change over time.

Finally, we also see that – at least on a corpus of up to roughly 25,000 utterances – using two or three levels of collocation is suffi-

cient to prevent overlearning for all but the highest frequency items. That is, in Figure 4.7 one only observes clear instances of overlearning for COLLOC3-SYLL and COLLOC2-SYLL for words of frequency 200 and larger. Thus, these kinds of models may actually be sufficient to prevent overlearning in 'realistic' scenarios of large but finite amounts of input despite the fact that, theoretically, they too will start undersegmenting when exposed too enough input.

UNDERSEGMENTATION AS DESIDERATUM? A final point which I briefly want to raise is that from a certain perspective, undersegmentation may be viewed as a 'feature' rather than a shortcoming of a segmentation model. Thus, Lignos (2012) argues that "a good model of infant word segmentation" should be able to replicate "at early stages of learning, undersegmentation of function word collocations" (p. 240). In particular, he points out that in his seminal work on language acquisition, Brown (1973) reports that infants tend to analyze pure function word sequences such as *that a* or verb-article collocations such as *get a*, *have a*, etc. as monomorphemic (see p. 391ff).

This raises a general issue with the kind of evaluation common in computational modeling of word segmentation – treating the orthographic transcript as gold standard is questionable from a developmental point of view where infants do, indeed, undersegment. Generally speaking, I believe this criticism to be valid; yet, addressing it properly requires establishing an alternative gold standard which, given that evidence about the kinds of undersegmentations infants do perform, is currently at best anecdotal, even in the language acquisition literature.

While I am looking forward to the development of such a standard by language acquisition researchers, at the current stage I am skeptical towards viewing particular patterns of undersegmentation as 'beneficial'. In the absence of a well-established standard against which to compare, this may result in situations where the relative merit of different models depends on the subjective judgment of the researcher as to which undersegmentations are 'good' and which are 'bad'. For this reason I exclusively evaluate against the orthographic transcript although I am aware of the limitation of this approach. In forthcoming work, Phillips and Pearl (in press) discuss in some more detail the limits of this approach and show how additional detailed evaluation that looks at specific attested error patterns can address some of its shortcomings.

Coming back to the question whether the overlearning exhibited by the models may not be viewed as an argument in favor of them, I also want to point out that – counter to Brown (1973)'s observation and Lignos (2012)'s demand, the collocation syllable models exhibit severe undersegmentation *not in the early but late stages of learning*, i.e. only once they have been exhibited to huge amounts of input. I believe this to cast general doubt on the idea that overlearning – at least as exhibited

by the models under discussion here – is desirable from a cognitive point of view.

4.5.4 *Performance of unconstrained models*

What remains to be given is an explanation why the collocation models without syllable structure lead to an overall worse performance but seem to exhibit a positive relation between amount of input and segmentation accuracy. The key to this, I believe, lies again in considering the kinds of regularities the model is sensitive to; this also provides an answer to the question raised by the previous chapter, i.e. whether *richer models require stronger constraints to perform well on little data.*

4.5.4.1 *Constraints on possible words*

With no restriction on what an actual word may look like, *high-frequency patterns of any kind* — including individual phonemes and short n-gram like sequences of phonemes — can be employed by a probabilistic segmentation model to explain the input they get. In particular, for little input with overall few word tokens and, consequently, relatively few repetitions for each of the actual word types, the evidence for high-frequency non-words (e.g., simply treating an individual phoneme as a word) is extremely high, possibly leading to over- rather than undersegmentation.

For example, inspecting the segmentations generated by COLLOC3 at month 11 we find that most frequent "word" is learned is *t* which is used in "collocations" like *t o*, *ge t* or, illustrating the problem very nicely, *j us t*. The reason these segmentations make sense for the model is that, of course, every individual phoneme occurs frequently in the input and, as such, is a plausible word; in particular as the fact that certain sequences occur more frequently than would be expected if all 'words' were independent, such as the preposition *to*, can be explained by positing the 'collocation' *t o*.

While 'statistically reasonable', segmentations involving 'words' such as *t* should be ruled out on the grounds that *t* simply is not a possible word in any language. And indeed, as can be seen from the performance of the collocation syllable model, adding a constraint that rules out these kinds of analyses forces the model to identify units that match closely with actual words.

Of course, this immediately raises the question why the UNIGRAM model does not require a possible word constraint. To understand this, recall that under the UNIGRAM model, all tokens in a segmentation are fully independent (see above and Chapter 2). Thus, it can only consider the *marginal probabilities* of the words in a sequence to predict its overall probability whereas a collocation (or a Bigram) model can take into account contextual dependencies.

With this, consider again a segmentation such as *t o* as posited by the COLLOC3 model. The reason the Unigram model does not posit this analysis is that the phoneme sequence /t u/ (corresponding to the word *to*[20]) occurs much more frequently than would be expected if it really were the sequence of two independent 'words' *t* and *u*. To wit, the relative frequency of the bigram /t u/ among all possible phoneme bigrams at month 11 is 0.6%. In contrast, the marginal frequencies of the phoneme /t/ is 0.07% and that of /u/ is 0.04%, hence according to a UNIGRAM model the expected frequency of /t u/ ought to be 0.3%, half of what is actually observed. Consequently, the Unigram model will posit the single word *to* as the alternative analysis is at odds with the empirically observed frequencies.

In contrast, a collocation model can account for the mismatch by assigning a probability directly to the collocation *t o* which need not be identical to the product of the marginal probabilities of *t* and *o*. In addition, by assigning independent probabilities to *t* and *o* which capture the occurrence of these phonemes in other contexts, the overall data can be fit much better, making the undersegmented *t o* the preferred solution.

Ironically then, the oversegmentation behavior is worst for models with a lot of additional structure such as the COLLOC3 model that, *when combined with a syllable structure constraint*, leads to the best performance. Without such a constraint, however, it uses the structure it has at its disposal to identify "statistically plausible" but linguistically meaningless segmentation.

Increasing the amount of input leads to the same overall change in segmentations – larger and larger units will be posited by the model the more and more input it processes. Yet, in this case this actually leads to more accurate rather than worse segmentation because the models tendency to oversegment is so large initially that it takes a huge amount of data to overcome it.

In line with this, at month 21 the top-5 word list is *ing*, *you*, *z*, *the* and *to*. This shows, that, as expected, more input does lead to larger units being segmented; but also that even for well over 20,000 utterances, the COLLOC3 model prefers to posit short 'words' as it has another three levels of collocations that it can use to assemble them into larger units.

The COLLOC2 and COLLOC model are less extreme in their oversegmentation behavior as they have fewer "levels" at their disposal, discouraging the excessive use of one-phoneme 'words' more severely. Thus, the oversegmentation behavior is less pronounced early on and, consequently, less input is required to achieve reasonable segmentation performance of around 70% token f-score. For these models, however, one also sees evidence for beginnings of undersegmentation for the COLLOC model, consistent with the observation that it seems to ceil at around 70% token f-score starting from month 15. Also, its highest-frequency

20 This bigram may, of course, also occur as part of another word, as in *altogether*.

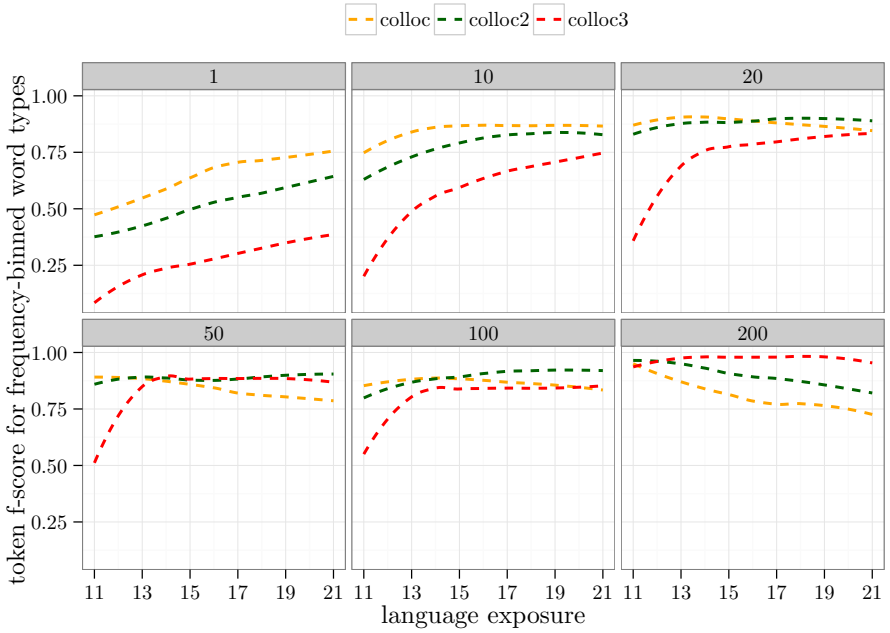


Figure 4.8: Token f-scores for word types of different frequencies in the test-set as a function of the size of the input. Note that the COLLOC model exhibits overlearning for words of frequency larger than 20, and COLLOC exhibits clear overlearning of words of frequency 200, indicating that even these models will overlearn as they see more and more input.

words at month 21 include "overlearned" units like *areyou* and *doyou*, demonstrating that the COLLOC model is beginning to overlearn just like the syllable-structure models.

This is also suggested when we look at Figure 4.8 which, just as Figure 4.7 for the models that overlearn, plots token f-score for words of different frequencies for the collocation models without syllable structure. In Figure 4.7, we observe an overlearning pattern already for words of frequency 10 and 20. Here, for COLLOC3 we see no such behavior at all although, starting at 50 for COLLOC, one also sees the suspicious drop; and for COLLOC2, one clearly sees overlearning on words of frequency 200, demonstrating that if we were to increase the input amount further, these models would also begin to undersegment more and more severely and, ultimately, exhibit an overall drop in segmentation performance.

In conclusion, I think this shows that the intuitively attractive behavior of such unconstrained models to get better over time is an artifact of their strong preference for short units that, for small amounts of data, masks the overlearning inherent in the model at the expense of segmentation accuracy.

To summarize this discussion then, the fact that models lacking syllable structure do not exhibit overlearning in my experiments should not

be taken as evidence that they are more adequate than their overlearning relatives. Rather, and consistent with the discussion in chapter 3, it demonstrates that richer models which include notions such as collocations require stronger constraints on what counts as a possible word to perform good segmentation on small amounts of data at all, highlighting the importance of constraints on the form of possible words.

4.5.5 *Parameter settings*

I want to briefly mention the possibility that overlearning may be addressed by manually choosing appropriate values for the models' hyper parameters for differently sized inputs. Goldwater (2007) observed and the discussion in Chapter 2 confirmed that the segmentations proposed by her Bigram model depend on the choice of hyper-parameters, and Johnson (2008b) observed a similar sensitivity to hyper parameters in Adaptor Grammar models.

Thus, it may be possible to identify different hyper parameter settings for different amounts of input that, to some extent, prevent or alleviate overlearning. I cannot rule out this possibility and it may, indeed, be a reasonable way of addressing these kinds of problems as they arise in practical applications of Bayesian models which exhibit similar kinds of behavior. From a computational modeling point of view, however, I find this to be an undesirable strategy.

First, the number of hyper parameters in an Adaptor Grammar is twice the number of adapted non-terminal symbols, making the identification of well-performing hyper parameters very challenging. Indeed, Johnson and Goldwater (2009) found that their automatically inferred hyper parameters (see Chapter 2, hyper parameter sampling) led to better performance than any of the values Johnson (2008b) had manually considered. As it is reasonable to expect that more realistic models of human language will be more complex and therefore will have even more hyper-parameters than the models investigated here, manually choosing hyper parameter values seems infeasible.

More importantly, I believe manual setting of hyper parameters or, as it is know in Machine Learning, parameter tuning to be a questionable move for computational models of language acquisition. It is certainly conceivable that (things loosely corresponding to) hyper parameters are indeed fixed by human biology to values that result in good segmentations. Yet, cross-linguistic evaluation of segmentation models has not suggested that a single set of hyper parameters performs well across different languages – the settings that work for English have so far worked considerably worse for other languages. Also, as argued in chapter 2, models which do not require hyper-parameters to be fixed to specific values should be preferred on general simplicity grounds over models that do require such prespecification.

4.6 CONCLUSION AND PERSPECTIVES

I have presented a novel corpus of English CDS derived from the Providence Corpus for studying models of word segmentation. This corpus makes it possible to address a wider range of questions than is currently common, for example with respect to the study of how the segmentations predicted by a model may change over time.

The primary contribution of this chapter is the identification of a so far unreported "overlearning" effect for state-of-the-art word segmentation models on large amounts of data and an explanation of this behavior. In particular, I argue that unless linguistic regularities are modeled explicitly, they will give rise to 'misleading patterns' which lead Bayesian word segmentation models to undersegment more as they see more input and, consequently, perform worse rather than better segmentation. Yet I found that by adding a sufficiently rich 'collocational' structure, this problem can be pushed back considerably to yield stable performance up to roughly 25,000 utterances.

I have also demonstrated that adding the power to capture more distributional dependencies to a segmentation model also adds the need for a stronger constraint on possible word forms. In particular, even though the COLLOC3-SYLL model emerges as the overall best and most robust segmentation model, the COLLOC3 model exhibits severe oversegmentation behavior and requires very large amounts of inputs to identify words rather than sub-word units such as very frequent phoneme n-grams. This provides a positive answer to the question raised in chapter 3 whether richer segmentation models also require more substantive constraints to perform well on small amounts of data.

I conclude this chapter by pointing out possible ways of extending this work, also noting two recent examples which can be seen as taking a first step in either of these directions.

4.6.1 *Future work*

I hypothesized that the tendency to undersegment needs to ultimately be addressed by modeling additional levels of linguistic structure that properly explain the high frequency of certain patterns in ways that do not require treating them as single units. A first step in this direction was taken in Synnave et al. (2014) where we[21] added a kind of semantic annotation capturing the idea of *activity contexts* (Roy et al., 2012). We found that, as expected, trying to account for the topicality of words – their tendency to occur more frequently in particular contexts as others – addresses overlearning and results in significant improvements.

Additional possible ways of adding semantics is to incorporate learning of simple word-object correspondences as in Jones et al. (2010) and Johnson et al. (2010) to the model. As the Providence corpus makes

21 I am third author of this paper.

available video recordings for all the transcripts, it should theoretically be possible to annotate each utterance with a set of salient objects. Following the strategy outlined in Johnson et al. (2010), it would then be straight-forward to add the ability to associate particular words or collocations with salient objects in context, providing a further way in which some notion of semantics could be incorporated into word segmentation.

In addition to semantics, incorporating syntactic dependencies is likely to help overcome overlearning of, in particular, function word sequences. A general challenge is choice of an appropriate set of syntactic notions that is not tailored directly at the language under investigation and, of course, additional information that could be used to determine syntactic dependencies, e.g. high-level meaning representations for utterances or parts of them. Focusing on English, Johnson et al. (2014) shows how incorporating the abstract knowledge that mono-syllabic function words can occur at the edges of collocational units improves segmentation and it would be interesting to see whether their models, evaluated on the Brent-Bernstein-Ratner corpus, also address overlearning on the Naima corpus or other corpora of comparable size.

5

EXPLORING THE ROLE OF STRESS IN BAYESIAN WORD SEGMENTATION

Stress has long been established as a major cue in word segmentation for English infants. In this chapter I show that enabling a current state-of-the-art Bayesian word segmentation model, the Johnson and Goldwater (2009)'s COLLOC3-SYLL model that was also examined in the previous chapter, to take advantage of stress cues noticeably improves its performance.

I find that the improvements range from 10 to 4%, depending on both the use of phonotactic cues and, to a lesser extent, the amount of evidence available to the learner. I also find that in particular early on, stress cues are much more useful for the model than phonotactic cues by themselves, consistent with the finding that children do seem to use stress cues before they use phonotactic cues.

Finally, I study how the model's knowledge about stress patterns evolves over time. I not only find that the model correctly acquires the most frequent patterns relatively quickly but also that the Unique Stress Constraint that is at the heart of a previously proposed model does not need to be built in but can be acquired jointly with word segmentation.

5.1 INTRODUCTION

Among the first tasks a child language learner has to solve is picking out words from the fluent speech that constitutes its linguistic input.[1] For English, stress has long been claimed to be a useful cue in infant word segmentation (Jusczyk et al., 1993, 1999b), following the demonstration of its effectiveness in adult speech processing (Cutler et al., 1986).

Several studies have investigated the role of stress in word segmentation using computational models, using both neural network and "non-statistical" approaches (Christiansen et al., 1998; Yang, 2004; Lignos and Yang, 2010; Lignos, 2011, 2012) which will be reviewed below. Bayesian models of word segmentation (Brent, 1999; Goldwater, 2007), however, have until recently completely ignored stress. The sole exception in this respect is Doyle and Levy (2013) who added stress cues to the Bigram model (Goldwater et al., 2009, also see Chapter 2), finding small but statistically significant improvements when applying the model to pre-syllabified data.

In this chapter, I extend on this work and show how to integrate stress cues into the flexible Adaptor Grammar framework (Johnson et al.,

1 The datasets and software to replicate the experiments are available from http://web.science.mq.edu.au/~bborschi/

2007b, see Chapter 2 for a brief review). This allows me to both start from a stronger baseline model and to systematically investigate how the role of stress cues interacts with other aspects of the model. In particular, I find that phonotactic cues to word-boundaries interact with stress cues, indicating synergistic effects for small inputs and partial redundancy for larger inputs.

Overall, I find that stress cues add roughly 6% token f-score to a model that does not account for phonotactics and 4% to a model that already incorporates phonotactics. Relatedly and in line with the finding that stress cues are used by infants before phonotactic cues (Jusczyk et al., 1999a), I observe that phonotactic cues require more input than stress cues to be used efficiently.

A closer look at the knowledge acquired by the models shows that the Unique Stress Constraint of Yang (2004) can be acquired jointly with segmenting the input instead of having to be pre-specified; and that the models correctly identify the predominant stress pattern of the input but underestimate the frequency of iambic words, which have been found to be missegmented by infant learners.

The outline of the chapter is as follows. In section 5.2 I review prior work. Section 5.3 introduces the adaptor grammar segmentation models and section 5.4 explains the experimental evaluation and its results. Section 5.5 discusses my findings, and Section 5.6 concludes and provides some suggestions for future research.

5.2 BACKGROUND AND RELATED WORK

5.2.1 *Lexical stress in word segmentation*

Lexical stress is the "accentuation of syllables within words" (Cutler, 2005). Following Cutler and Carter (1987)'s observation that stressed syllables tend to occur at the beginnings of words in English, Jusczyk et al. (1993) investigated whether infants acquiring English take advantage of this fact. Their study demonstrated that this is indeed the case for 9 month olds, although they found no indication of using stressed syllables as cues for word boundaries in 6 month olds. Their findings have been replicated and extended in subsequent work (Jusczyk et al., 1999b; Thiessen and Saffran, 2003; Curtin et al., 2005; Thiessen and Saffran, 2007), identifying a by now well-established set of findings about the role stress plays in early word segmentation:

1. English infants treat stressed syllables as cues for the beginnings of words from roughly 7 months of age, suggesting that the role played by stress needs to be acquired, and that this requires antecedent segmentation by non-stress-based means (Thiessen and Saffran, 2007)

2. English infants exhibit a preference for low-pass filtered stress-initial words from this age, suggesting that it is indeed stress and not simply other phonetic or phonotactic properties (which are missing in low-pass filtered speech) that are treated as a cue for word-beginnings (Jusczyk et al., 1993)

3. phontactic cues seem to be used later than stress cues by infants, evidence for their being used occurring only at around 9 months and later (Mattys et al., 1999; Mattys and Jusczyk, 2000; Jusczyk et al., 1999a)

4. once stress start being used from around 7 months, they seem to outweigh other (e.g. distributional or phonotactic) cues to word boundaries up until roughly 10 to 11 months, illustrated by frequent mis-segmentations of words that do not conform to the dominant stress-initial pattern such as /g i ˈt ar/ (guitar) (Thiessen and Saffran, 2003)

The experiments in this chapter address these points from a computational modeling perspective and shows that joint Bayesian models that incorporate distributional, phonotactic and stress cues exhibit similar behavior as will be discussed below.

It is worth noting at this point that the English stress system is in fact quite complex. The linguistic knowledge required to correctly predict the stress of a word involves more than merely knowing that most words tend to be stressed on their first syllable – common linguistic analyses of stress systems make use of ideas such as syllable weight and foot structure, all of which will be ignored in this chapter (for an accessible introduction, see Hayes, 2009, chapter 14). The reason for this is that there is little evidence that the subtle aspects of the English stress system that require a deeper analysis play an important role at the stage of word segmentation.

5.2.2 *Prior modeling work on stress in word segmentation*

The earliest computational model for word segmentation incorporating stress cues I am aware of is the recurrent network model of Christiansen et al. (1998) and Christiansen and Curtin (1999). Segmentation using neural networks has fallen somewhat out of favor recently, and its performance is, indeed, considerably lower than that of the other *lexical segmentation models* which incorporate the idea of an explicit lexicon in which word forms are stored; in this case, the best segmentation accuracy reported by Christiansen et al. (1998) and Christiansen and Curtin (1999) is a mere 44% whereas the base-line models I consider in this chapter already attain an accuracy of well over 80%. For this reason, I do not discuss these models in more detail.

A highly influential segmentation model that explicitly incorporated stress cues is presented by Yang (2004). It is a simple incremental algo-

rithm that embodies an allegedly universal substantive constraints on the stress-patterns of possible words called the *Unique Stress Constraint* (USC) which was defined by Yang and Gambell (2005) as follows:

> UNIQUE STRESS CONSTRAINT A word can bear at most one primary stress[.] (p. 19)

A constraint like this is made plausible by the idea that one of the roles played by stress is its *culminative function*, i.e. the idea that it identifies the single most salient syllable of a word(Fromkin, 2001; Cutler, 2005). In a sense, the USC can be viewed almost as a tautology – primary stress is *defined as* the most prominent stress in a word, and of course there is at most a single most prominent stress in any word. Yang (2004) argued, rather influentially, that statistical word segmentation are outperformed by a simple algorithm that relies on a 'Universal Grammar' principle such as the USC rather than distributional regularities of the input; and indeed, he reported very good segmentation results of 85.6% segmentation accuracy on a corpus of pre-syllabified child directed speech, outperforming a transitional-probability learner of the kind proposed by early work such as as Saffran et al. (1996) by a huge margin.

However, the high scores Yang (2004) and Yang and Gambell (2005) reported depend on the questionable assumption that, in fact, every word token contain a stressed syllable, including function words. While this assumption has more recently also been made explicitly by Doyle and Levy (2013) (discussed below), Lignos further explored Yang's original algorithm, taking into account that function words should not be assumed to possess lexical stress cues. While his scores are in line with those reported by Yang, the importance of stress for this learner were more modest, providing a gain of around 2.5% (Lignos, 2011).

Working in the Bayesian framework I use as well, Doyle and Levy (2013) extend the Bigram model of Goldwater et al. (2009) by adding *stress-templates* to the base distribution which defines the prior expectations a model has about possible words (see Chapter 2, 3, and 4 for a discussion about how assumptions built into the base distribution affect word segmentation). A stress-template indicates how many syllables the word has, and which of these syllables (if any) are stressed. Thus, unlike in the Yang/Lignos the model of Doyle and Levy (2013) can, at least in theory, form explicit expectations about the stress pattern of its language by learning a distribution over stress-templates from the input.

Interestingly, Doyle and Levy (2013) do not directly examine the probabilities their model infers for the different stress-templates but they do report, on the basis of calculating the fraction stress-initial words in the proposed segmentation, that their model does slightly prefer stress-initial words over the baseline model which does not make any use of stress cues. The ability to explicitly represent expectations

about stress patterns is something that differentiates probabilistic models from other approaches, making it possible to not only ask whether stress aids segmentation but also address how the role played by stress can be acquired. I extend on this in my experiments by directly examining how expectations about stress patterns develop over time.

Like Yang and Lignos, Doyle and Levy (2013) report that stress cues aid segmentation, although their reported gain of 1% in token f-score is even smaller than that reported by Lignos (2011). Also, a slightly surprising result of their work (which may even be viewed as supporting Yang and Gambell (2005)'s criticism of 'statistical' learners) is that a simple baseline which proposes to put a boundary at every possible position reaches a segmentation accuracy of 82% but Doyle and Levy's stress model only 68%.

The approach presented in this chapter is, in terms of also using a Bayesian model for word segmentation, similar to theirs but differs in several respects. First, I use Adaptor Grammars (Johnson et al., 2007b), a grammar-based formalism for specifying non-parametric hierarchical models. Previous work explored the usefulness of, for example, syllable-structure (Johnson, 2008b; Johnson and Goldwater, 2009; Börschinger et al., 2012, also previous chapter of this thesis) or morphology (Johnson, 2008b,a) in word segmentation. The closest work to this is Johnson and Demuth (2010) who investigate the usefulness of tones for Mandarin phonemic segmentation. Their way of adding tones to a model of word segmentation is very similar to my way of incorporating stress which I will explain in the next section.

5.3 MODELS

I give an intuitive description of the mathematical background of Adaptor Grammars in 5.3.1, referring the reader to Johnson et al. (2007b) for technical details and to chapter 2 for a brief review. The models I examine are derived from the collocational model of Johnson and Goldwater (2009) by varying three parameters, resulting in 6 models: two baselines that do not take advantage of stress cues and either do or do not use phonotactics, as described in Section 5.3.2; and four stress models that differ with respect to the use of phonotactics, and as to whether they embody the Unique Stress Constraint introduced by Yang (2004). I describe these models in section 5.3.3.

5.3.1 *Adaptor Grammars*

Briefly, an adaptor grammar (AG) is a probabilistic context-free grammar (PCFG) with a special set of *adapted* non-terminals which are treated differently from the non-terminals of a standard PCFG. I use underlining to distinguish adapted non-terminals ($\underline{\mathrm{X}}$) from non-adapted non-terminals (Y).

The distribution for each adapted non-terminal $\underline{\text{X}}$ is drawn from a Pitman-Yor Process which takes as its base-distribution the tree-distribution over trees rooted in $\underline{\text{X}}$ as defined by the PCFG. As an effect, each adapted non-terminal can be seen as having associated with it a cache of previously-generated subtrees that can be reused without having to be regenerated using the individual PCFG rules.[2]

This allows AGs to learn reusable sub-trees such as words, sequences of words, or smaller units such as Onsets and Codas. Thus, while ordinary PCFGs have a finite number of parameters (one probability for each rule), Adaptor Grammars in addition have a parameter for every possible complete tree rooted in any of its adapted non-terminals, leading to a potentially infinite number of such parameters. The Pitman-Yor Process induces a rich-get-richer dynamics, biasing the model towards identifying a small set of units that can be reused as often as possible. In the case of word segmentation, the model will try to identify as compact a lexicon as possible to segment the unsegmented input.

5.3.2 *Baseline models*

The starting point is the state-of-the-art AG model for word segmentation, Johnson and Goldwater (2009)'s colloc3-syll model, reproduced in Figure 5.1.[3] The model assumes that words are grouped into larger collocational units that themselves can be grouped into even larger collocational units. This accounts for the fact that in natural language, there are strong word-to-word dependencies that need to be accounted for if severe undersegmentations of the form "is in the" are to be avoided (Goldwater, 2007; Johnson and Goldwater, 2009; Börschinger et al., 2012, see also the previous chapter for extended discussion).

It also relies on an arguably universal form of syllable structure to constrain the space of possible words. Finally, this model can learn word-initial onsets and word-final codas. In a language like English, this ability provides additional cues to word-boundaries as certain onsets are much more likely to occur word-initially than medially (e.g. "bl" in "black"), and analogously for certain codas (e.g. "dth" in "width" or "ngth" in "strength").

I define an additional baseline model by replacing rules (5.8) and (5.9) by (5.20), and deleting rules (5.10) to (5.15). This removes the model's ability to use phonotactic cues to word-boundaries.

$$\underline{\text{Word}} \rightarrow \text{Syll}\,(\,\text{Syll}\,)\,(\,\text{Syll}\,)\,(\,\text{Syll}\,) \tag{5.20}$$

I refer to the model in Figure 5.1 as the colloc3-phon model, and the model that results from substituting and removing rules as described

2 This idea relies on the Chinese Restaurant representation, see Chapter 2 for discussion.

3 I follow Johnson and Goldwater (2009) in limiting the length of possible words to four syllables to speed up runtime. In pilot experiments, this choice did not have a noticeable effect on segmentation performance.

$$\text{Collocations3} \rightarrow \underline{\text{Collocation3}}^{+} \quad (5.1)$$
$$\underline{\text{Collocation3}} \rightarrow \text{Collocations2} \quad (5.2)$$
$$\text{Collocations2} \rightarrow \underline{\text{Collocation2}}^{+} \quad (5.3)$$
$$\underline{\text{Collocation2}} \rightarrow \text{Collocations1} \quad (5.4)$$
$$\text{Collocations1} \rightarrow \underline{\text{Collocation}}^{+} \quad (5.5)$$
$$\underline{\text{Collocation}} \rightarrow \text{Words} \quad (5.6)$$
$$\text{Words} \rightarrow \underline{\text{Word}}^{+} \quad (5.7)$$
$$\underline{\text{Word}} \rightarrow \text{SyllIF} \quad (5.8)$$
$$\underline{\text{Word}} \rightarrow \text{SyllI}\ (\ \text{Syll}\ )\ (\ \text{Syll}\ )\ \text{SyllF} \quad (5.9)$$
$$\text{SyllIF} \rightarrow (\ \underline{\text{OnsetI}}\ )\ \text{RhymeF} \quad (5.10)$$
$$\text{SyllI} \rightarrow (\ \underline{\text{OnsetI}}\ )\ \text{Rhyme} \quad (5.11)$$
$$\text{SyllF} \rightarrow (\ \underline{\text{Onset}}\ )\ \text{RhymeF} \quad (5.12)$$
$$\underline{\text{CodaF}} \rightarrow \text{Consonant}^{+} \quad (5.13)$$
$$\text{RhymeF} \rightarrow \text{Vowel}\ (\ \underline{\text{CodaF}}\ ) \quad (5.14)$$
$$\underline{\text{OnsetI}} \rightarrow \text{Consonant}^{+} \quad (5.15)$$
$$\text{Syll} \rightarrow (\ \underline{\text{Onset}}\ )\ \text{Rhyme} \quad (5.16)$$
$$\text{Rhyme} \rightarrow \text{Vowel}\ (\ \underline{\text{Coda}}\ ) \quad (5.17)$$
$$\underline{\text{Onset}} \rightarrow \text{Consonant}^{+} \quad (5.18)$$
$$\underline{\text{Coda}} \rightarrow \text{Consonant}^{+} \quad (5.19)$$

Figure 5.1: Adaptor Grammar for the baseline model. I use regular-expression notation to abbreviate multiple rules. $X^{\{n\}}$ stands for up to n repetitions of X, brackets indicate optionality, and X^{+} stands for one or more repetitions of X. $\underline{X}$ indicates an adapted non-terminal. Rules that introduce terminals for the pre-terminals Vowel, Consonant are omitted. Refer to the main text for an explanation of the grammar.

as the colloc3-nophon model. One can also limit the model's ability to capture word-to-word dependencies by removing either rules (5.1) to (5.2), yielding a colloc2-model, to (5.4) to yield a colloc-model or all rules up to (5.6) to yield a unigram-moldel (Johnson and Goldwater, 2009, previous chapter).

These models, in particular the colloc-model, are more similar to the Bigram model used in Doyle and Levy (2013) and, as expected, we find their performance to gradually deteriorate as compared to the colloc3-model (as discussed at length in the previous chapter). For this reason, I focus on the colloc3-models.

5.3.3 *Stress-based models*

In order for stress cues to be helpful, the model must have some way of associating the position of stress with word-boundaries. Intuitively, the reason stress helps infants in segmenting English is that a stressed syllable is a reliable indicator of the beginning of a word (Jusczyk et al., 1993). Thus, if one already knew of this correlation one way to take advantage of stress cues would be to always hypothesize (or hypothesize with very high probability) a word boundary before every observed stressed syllable; indeed, this *metrical segmentation strategy* (Cutler, 1991) has been found to be employed by adult speakers of English.

In the context of language acquisition, however, knowledge about the proper role of stress cannot be assumed from the outset as, crucially, languages differ with respect to how stressed syllables relate to word boundaries. Nevertheless, one can expect that if there is a (reasonably) reliable relationship between the position of stressed syllables and beginnings (or endings) of words, a learner might exploit this relationship by somehow picking up on it.

An alternative idea about how stress might be exploited in word segmentation is the one argued for by Yang (2004) and Lignos (2012). There, the USC provides a hard constraint that *excludes* possible segmentations that violate the USC and, thus, cuts down the space that needs to be considered by a learner, arguably facilitating the segmentation problem.

In the Bayesian framework used here, both ideas can be captured directly (and independently) by modifying the base distribution or lexical generator that is responsible for generating Word s. I first describe how stress-preferences can be incorporated before I show how the USC can, optionally, also be added to the models.

Here, changing the lexical generator corresponds to modifying the rules expanding Word . A straight-forward way to modify it accordingly is to enumerate all possible sequences of stressed and unstressed syllables.[4] While there is a huge number of alternative rules that could be used and that encode subtly different biases, I found this approach to work well already.

In the data, stress cues are represented using a special terminal "$*$" that follows a stressed vowel, as illustrated in Figure 5.2. In the grammar, "$*$" is constrained to only surface following a Vowel , rendering a syllable in which it occurs stressed (SSyll). Syllables that do not contain a "$*$" are considered unstressed (USyll).

By performing inference for the probabilities with which Word expands into any of the possibly sequences of stressed and unstressed syllables, i.e. the rules abbreviated by schema (5.21), the models can, for example, learn that a bi-syllabic word that is stress-initial (a trochee) is more probable than one that puts stress on the second syllable (an

4 This is, in essence, also the strategy chosen by Doyle and Levy (2013).

$$\underline{\text{Word}} \rightarrow \{\,\text{SSyll} \mid \text{USyll}\,\}^{\{1,4\}} \tag{5.21}$$
$$\text{SSyll} \rightarrow (\,\underline{\text{Onset}}\,)\,\text{RhymeS} \tag{5.22}$$
$$\text{USyll} \rightarrow (\,\underline{\text{Onset}}\,)\,\text{RhymeU} \tag{5.23}$$
$$\text{RhymeS} \rightarrow \text{Vowel} * (\,\underline{\text{Coda}}\,) \tag{5.24}$$
$$\text{RhymeU} \rightarrow \text{Vowel}\,(\,\underline{\text{Coda}}\,) \tag{5.25}$$
$$\underline{\text{Onset}} \rightarrow \text{Consonant}^{+} \tag{5.26}$$
$$\underline{\text{Coda}} \rightarrow \text{Consonant}^{+} \tag{5.27}$$

Figure 5.2: Description of the colloc3-nophon-stress model. I use $\text{X}^{\{m,n\}}$ for "at least m and at most n repetitions of X" and $\{\text{X} \mid \text{Y}\}$ for "either X or Y". Stress is associated with a vowel by suffixing it with the special terminal symbol $*$, leading to a distinction between stressed (SSyll) and unstressed (USyll) syllables. A word can consist of any possible sequence of up to four syllables, as indicated by the regular-expression notation. By additionally adding initial and final variants of SSyll and USyll as in Figure 5.1, phonotactics can be combined with stress cues.

iamb). This would be represented by having $\text{P}(\,\underline{\text{Word}} \rightarrow \text{SSyll USyll}\,) > \text{P}(\,\underline{\text{Word}} \rightarrow \text{USyll SSyll}\,)$.

Thus, by assigning probabilities to not only words but also to stress-patterns, our model can (partly) capture the regularities exhibited its input and, crucially, use these preferences to identify words.

One can combine this lexical generator with the colloc3-nophon baseline, resulting in the colloc3-nophon-stress model. One can also add phonotactics to the lexical generator in Figure 5.2 by adding initial and final variants of SSyll and USyll, analogous to rules (5.8) to (5.15) in Figure 5.1. This yields the colloc3-phon-stress model.

Finally, one can add the Unique Stress Constraint (USC) (Yang, 2004) by excluding all variants of rule (5.21) that generate two or more stressed syllables. For example, the lexical generator for the colloc3-nophon-stress model will include the rule $\underline{\text{Word}} \rightarrow \text{SSyll SSyll}$; this pattern violates the USC as it generates a word with two stressed syllables, and thus this rule will be missing from a lexical generator that embodies the USC. I refer to the models that include the USC as colloc3-nophon-stress-usc and colloc3-phon-stress-usc models. A compact overview of the six different models is given in Table 5.1.

5.4 EXPERIMENTS

I evaluate the models on several corpora of child directed speech. I first describe the corpora used, then the experimental methodology employed and finally the experimental results. As the trend is comparable across all corpora, I only discuss in detail results obtained on the Alex

grammar	phon	stress	USC
colloc3-nophon			
colloc3-phon	•		
colloc3-nophon-stress		•	
colloc3-phon-stress	•	•	
colloc3-nophon-stress-usc		•	•
colloc3-phon-stress-usc	•	•	•

Table 5.1: The different models used in the experiments. "phon" indicates whether phonotactics are used, "stress" whether stress cues are used and "usc" whether the Unique Stress Constraint is assumed.

orthographic	the **do**-gie
no-stress	dh ah d ao g iy
stress	dh ah d ao * g iy

Table 5.2: Illustration of the input-representation I choose. I indicate primary stress according to the dictionary with bold-face in the orthography. The phonemic transcription uses ARPABET and is produced using an extended version of CMUDict. Primary stress is indicated by inserting the special symbol "*" after the vowel of a stressed syllable.

corpus. For completeness, however, Table 5.4 reports the "standard" evaluation of performing inference over all of the three corpora.

5.4.1 *Corpora and corpus creation*

Following Christiansen et al. (1998) and Doyle and Levy (2013), I use the Korman corpus (Korman, 1984) as one of the corpora. It comprises child-directed speech for very young infants, aged between 6 and 16 weeks and, like all other corpora used in this chapter, is available through the CHILDES database (MacWhinney, 2000). I derive a phonemicized version of the corpus using an extended version of CMUDict (Carnegie Mellon University, 2008)[5], as I was unable to obtain the stress-annotated version of this corpus used in previous experiments. The phonemicized version is produced by replacing each orthographic word in the transcript with the first pronunciation given by the dictionary. CMUDict also annotates lexical stress, and I use this information to add stress cues to the corpus. I only code primary lexical stresses in the input, ignoring secondary stresses in line with experimental work that indicates that human listeners are capable of reliably distinguishing primary and secondary stress (Mattys, 2000). Due to the very low frequency of words with 3 or more syllables in these corpora, this choice has very little effect on the number of stress cues available in the input.

5 http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/cmudict.0.7a

My version of the Korman corpus contains, in total, 11,413 utterances. Unlike Christiansen et al. (1998), Yang (2004), and Doyle and Levy (2013), I follow Lignos and Yang (2010) in making the more realistic assumption that the 94 mono-syllabic function words listed by Selkirk (1984) never surface with lexical stress. As function words account for roughly 50% of the tokens but only roughly 5% of the types in the corpora, this means that the type and token distribution of stress patterns differs dramatically in all the corpora I examine, as can be seen from Table 5.3.

I also added stress information to the Brent-Bernstein-Ratner corpus (Bernstein-Ratner, 1987; Brent, 1999), following the procedure just outlined. This corpus is a de facto standard for evaluating models of Bayesian word segmentation (Brent, 1999; Goldwater, 2007; Goldwater et al., 2009; Johnson and Goldwater, 2009), comprising in total 9790 utterances.

As a third corpus, I use the Alex portion of the Providence corpus (Demuth et al., 2006; Börschinger et al., 2012). A major benefit of the Providence corpus (also see previous chapter) is that the video-recordings from which the transcripts were produced are available through CHILDES alongside the transcripts. This will allow future work to rely on even more realistic stress cues that can be derived directly from the acoustic signal. I believe choosing a corpus that makes richer information available will be important for future work on stress (and other acoustic) cues.

Another major benefit of the Alex corpus is that it provides longitudinal data for a single infant, rather than being a concatenation of transcripts collected from multiple children, such as the Korman and the Brent-Bernstein-Ratner corpus. In total, the Alex corpus comprises 17,948 utterances.

To make the results roughly comparable in terms of overall input size, I only use the first 10,000 utterances for both the Korman and the Alex corpus as the Brent-Bernstein-Ratner corpus only comprises 9,790 utterances. Note that despite the differences in age of the infants and overall make-up of the corpora, the distribution of stress patterns across the corpora is roughly the same, as shown by Table 5.3 for the first 10,000 utterances of each of the corpora. This suggests that the distribution of stress patterns both at a token and type level is a robust property of English child-directed speech.

5.4.2 *Syllabified versus phonemic input*

A major point of divergence from previous work such as Yang (2004), Lignos (2012) and Doyle and Levy (2013) is my use of phonemic rather than pre-syllabified input. I already discussed this choice in Chapter 3 but briefly repeat what I consider a convincing argument against the use of pre-syllabified input.

Pattern	brent		korman		alex	
	Token	Type	Token	Type	Token	Type
W^+	.48	.07	.47	.08	.44	.05
SW^*	.49	.86	.49	.86	.52	.87
WSW^*	.03	.07	.03	.06	.04	.07
Other	.00	.00	.00	.00	.00	.00

Table 5.3: Relative frequencies for stress patterns for the corpora used in the experiments. X^* stands for 0 or more, X^+ for one or more repetitions of X, and S for a stressed and W for an unstressed syllable. Note the stark asymmetry between type and token frequencies for unstressed words. Up to two-decimal places, patterns other than the ones given have relative frequency 0.00 (frequencies might not sum to 1 as an artefact of rounding to 2 decimal places). Note that these relative frequencies are calculated from the gold standard.

Onset-maximization is an efficient means of syllabifying words but, as can be seen easily, runs into problems when syllabifying an unsegmented corpus into syllables. In particular, a sequence such as /l ʊ k æt/ ("lookat") will, relying simply on onset-maximization, be syllabified as /l ʊ - k æt/ because /k/ is a valid onset in English. While it is true that, in fluent speech, this kind of re-syllabification where the coda of one word (here, the /k/ of "look") may be analyzed as the onset of the following word, running a segmentation model on input in which this sequence would be represented by two atomic syllables /lʊ/ and /k æt/ will prevent the model from segmenting this stretch correctly.

Preparing the syllabification of the corpus on the basis of a transcript that includes the word boundaries, however, corresponds to the dubious assumption that an infant may, indeed, have applied onset-maximization as if it knew the words. Consequently, I decide to use phonemic input but apply a model which can infer a latent syllabification *jointly with* syllabifying the corpus.

5.4.3 *Experimental questions*

The aim of the experiments is to understand the contribution of stress cues to the Bayesian word segmentation models described in Section 5.3. To get an idea of how input size interacts with this, I look at prefixes of the corpora with increasing sizes (100, 200, 500, 1000, 2000, 5000, and 10,000 utterances), essentially the methodology I also used in Chapter 3. Yet, the increments used here are much smaller and allow us to get at possible differences between models that, on corpora of 1000 or more utterances, may perform indistinguishably.

The standard evaluation of segmentation models, going back to Brent (1999), involves having them segment their input in an unsupervised

manner and evaluating performance on how well they segmented that input. I measure segmentation performance using the standard metric of token f-score (Brent, 1999) which is the harmonic mean of token precision and recall. Token f-score provides an overall impression of how accurate individual word tokens were identified. To illustrate, if the gold segmentation is "the dog", the segmentation "th e dog" has a token precision of $\frac{1}{3}$ (one out of three predicted words is correct); a token recall of $\frac{1}{2}$ (one of the two gold words was correctly identified); and a token f-score of 0.4.

Following the idea of chapter 3, to make segmentation scores of models which processed different amounts of inputs comparable, I additionally evaluate the models on a test set for each corpus. In addition, use of a separate test set has previously been suggested as a means of testing how well the knowledge a learner acquired generalizes to novel utterances (Pearl et al., 2010), allowing us to more directly ask in what sense learning about stress helps segmentation of novel utterances.

I create the test-sets by taking the final 1000 utterances for each corpus. These 1000 utterances will be segmented by the model after it has performed inference on its input, without making any further changes to the lexicon that the model has induced. In other words, the model will have to segment each of the test utterances using only the lexicon (and any additional knowledge about co-occurrences, phonotactics, and stress) it has acquired from the training portion of the corpus during inference.

Finally, I want to understand what kind of stress pattern preferences the models acquire. Recall that these models explicitly represent their knowledge about stress in the form of the probabilities it assigns to the different expansions of the adapted non-terminal Word in (5.21). I modified the adaptor grammar inference software to also produce samples of the rule probabilities in addition to sample segmentation and, for every sample segmentation, collect the associated rule probabilities. In this way, one can directly examine what preferences the model acquired and also how they develop as it processes more input.

5.4.4 *Inference*

For inference, I closely follow the methodology used in the previous chapter and Johnson and Goldwater (2009). Using my modified version of the adaptor grammar sampler that produces samples for the rule probabilities, I run 4 independent Markov Chains for 1,000 iterations. The first 800 iterations are used as burn-in and I collect 20 samples with a lag of 10 iterations between each sample from each chain, for a total of 80 sample segmentation per model for the input and 80 sample segmentations for the test set.

For evaluation, I determine the maximum marginal a posteriori segmentation – for each individual utterance, I determine the segmentation

p	s	usc	alex		korman		brent	
			train	test	train	test	train	test
			.81	.81	.85	.83	.82	.82
•			.85	.84	.86	.84	.86	.86
	•		.86	.87	.87	.86	.86	.87
•	•		**.88**	**.88**	**.88**	.87	**.87**	.87
	•	•	.87	**.88**	.87	**.88**	.86	.87
•	•	•	**.88**	**.88**	**.88**	.87	**.87**	**.88**

Table 5.4: Token f-scores on both train and test portions for all three corpora when inference is performed over the full corpus. Note that the benefit of stress is clearer when evaluating on the test set, and that overall, performance of the different models is comparable across all three corpora. Models are coded according to the key in Table 5.1.

which, across all samples, occurred the most frequently. This produces a single segmented input and test corpus for each model which I evaluate.

Also, rather than manually determining the hyper parameters for the models I put independent vague $\mathsf{Gamma}(0.01, 0.01)$ priors on the concentration parameters for each adapted non-terminal and independent uniform $\mathsf{Beta}(1, 1)$ priors on the discount parameter for each adapted non-terminal; the Adaptor Grammar sampler then also performs inference for the hyper parameters.[6]

5.4.5 *Experimental conditions*

Each of the six models is evaluated on inputs of increasing size, starting at 100 and ending at 10,000 utterances. This allows us to investigate both how performance and "knowledge" of the learner varies as a function of input size, similar to my experiments in Chapter 3 which showed that input size can have dramatic effects on model performance.

For completeness and comparison to prior work, however, I also report the "standard" evaluation, i.e. performance of the models in Table 5.4, i.e. the token f-score attained when inference was performed over the entire input. This I do for all of the three corpora on which I performed the experiments. As the experimental results are very similar across all corpora, I will focus the more detailed discussion on the Alex corpus.

For this, Figure 5.3 depicts how token f-score on the test set (depicted on the y-axis) changes as a function of the input size (depicted on the x-axis).

6 Note that I am using a different parametrization of the Gamma distribution than Johnson and Goldwater (2009), see Table 2.1 on page 22. Thus, the $\mathsf{Gamma}(0.01, 0.01)$ prior I use is equivalent to the $\mathsf{Gamma}(0.01, 100)$ prior Johnson and Goldwater (2009) suggest.

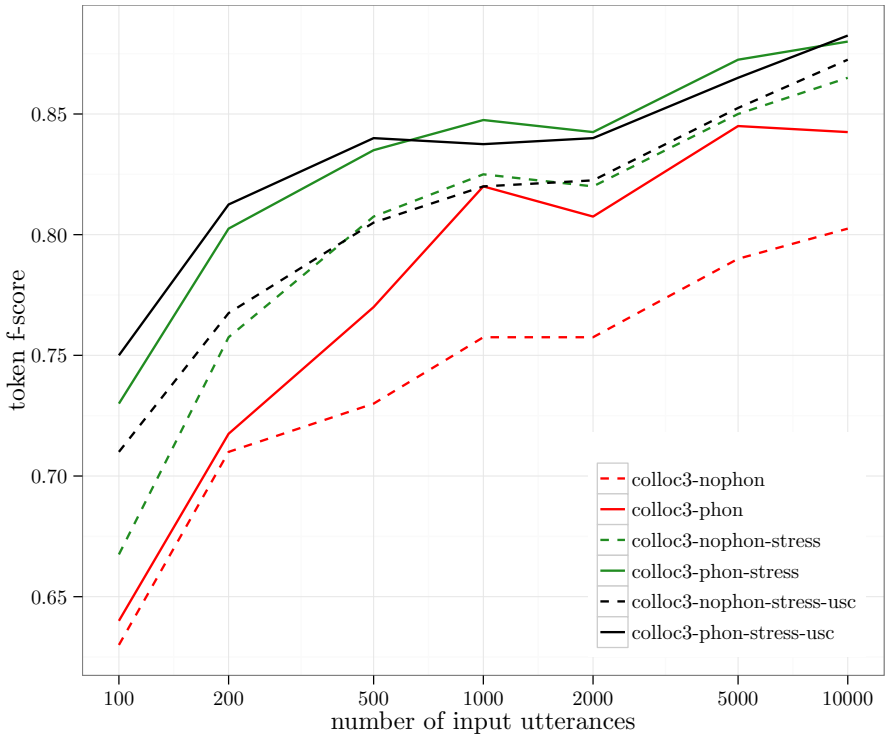


Figure 5.3: Segmentation performance of the different models, across different input sizes and as evaluated on the test-set for the Alex corpus. The no-stress baselines are given in red, the stress-models without the Unique Stress Constraint (USC) in green and the ones including the USC in black. Solid lines indicate models that use, dashed lines models that do not use phonotactics. Refer to the text for discussion.

5.5 DISCUSSION

We find a clear improvement for the stress-models over both the colloc3-nophon and the colloc3-phon models. As can be seen in Table 5.4, the overall trend is the same for all three corpora, both when evaluating on the input and the separate test-set.[7] Adding both stress or phonotactics by itself improves over the respective base-lines, although combining the two when performing inference over the full corpora yields no noticeably gains. Also note how the relative gain for stress is roughly 1% higher when evaluating on the test-set; this might have to do with Jusczyk (1997)'s observation that the advantage of stress "might be more evident for relatively unexpected or unfamiliarized strings".

7 I performed Wilcox rank sum tests on the individual scores of the 4 independent chains for each model on the full training data sets and found that the stress-models were always significantly more accurate ($p < 0.05$) than the baseline models except when evaluating on the training data for the Korman and Brent corpora.

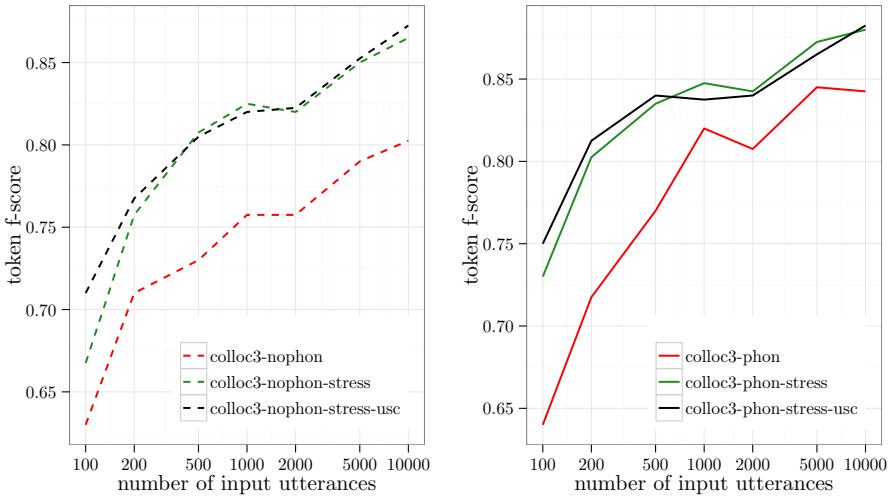


Figure 5.4: Token f-score on the test-set as a function of the input size, with models using phonotactics on the right and those without on the left. This presents the same information as Figure 5.3 but makes it easier to visualize the gain simply due to stress.

Moving beyond evaluating on a single huge test-set, however, we see from Figure 5.3 further interesting differences between the colloc3-nophon and the colloc3-phon models that only become evident when considering different input sizes.

5.5.1 *Stress cues without phonotactics*

For ease of visualization, Figure 5.4 plots the same information as Figure 5.3 but uses a separate plot for the models without and with phonotactics.

For the colloc3-nophon models, we observe a relatively stable improvement by adding stress cues of 6-7%, irrespective of input size and whether or not the Unique Stress Constraint (USC) is assumed. The sole exception to this occurs when the learner only gets to see 100 utterances: in this case, the colloc-nophon-stress model only shows a 3% improvement, whereas the colloc3-nophon-stress-usc model obtains a boost of roughly 8%.

Noticeable consistent differences between the colloc3-nophon-stress and colloc3-nophon-stress-usc model, however, all but disappear starting from around 500 utterances. This is somewhat surprising, considering that it is the USC that was argued by Yang (2004) to be key for taking advantage of stress.[8]

8 On data in which function words are marked for stress (as in Yang (2004) and Doyle and Levy (2013)), the USC yields extremely high scores across all models, simply because roughly every second word is a function word. Given that this assumption is extremely unnatural, I do not take this as an argument for the USC.

I take this behavior to indicate that even with as little evidence as 200 to 500 utterances, a Bayesian ideal learner can leverage stress cues in such a way that adding the USC does not add anything. In fact, from the discussion in Section 5.5.3 it will become clear that the model does, in a sense, infer the USC constraint from the input.

5.5.2 *Stress cues and phonotactics*

Overall, the models including phonotactic cues perform better than those that do not rely on phonotactics. However, the overall gain contributed by stress to the colloc3-phon baseline is smaller, although this seems to also depend on the size of the input.

Thus, while phonotactics by itself appears to be a powerful cue, yielding a noticeable 4-5% improvement over the colloc3-nophon baseline, the learner seems to require at least around 500 utterances before the colloc3-phon model becomes clearly more accurate than the colloc3-nophon model. There is virtually no improvement from adding phonotactics for 100 and 200 utterances, suggesting that a certain amount of input is required for phonotactic cues to become useful.

In contrast, even for only 100 utterances stress cues by themselves provide a 3% improvement to the colloc3-nophon model, as is very clear from Figure 5.4. This shows that these kinds of cues can be taken advantage of earlier, at least by a Bayesian ideal learner.

While the number of utterances processed by a Bayesian ideal learner is not directly related to developmental stages (also see the critical discussion at the end of Chapter 3), this observation is consistent with the psycholinguists' claim that phonotactics are used by infants for word segmentation after they have begun to use stress for segmentation (Jusczyk et al., 1999a).

The 4% difference between the colloc3-phon-stress / colloc3-phon-stress-usc models to the colloc3-phon baseline is smaller than the 7% difference between the colloc3-nophon and colloc3-nophon-stress models. This shows that there is a redundancy between phonotactic and stress cues in large amounts of data, as their joint contribution to the colloc3-nophon baseline of roughly 7% is less than the sum of their individual contributions at 10,000 utterances, of 4% (for phonotactics) and 6% (for stress).

This redundancy does, however, seem to depend to a large extent on the amount of input. In particular, at 100 utterances the addition of stress cues leads to an 8 – 10% improvement, depending on whether or not the USC is assumed, whereas for the colloc3-nophon model we only observed a 3 – 8% improvement. This is particularly striking when we consider that by themselves, the phonotactic cues only contribute a 1% improvement to the colloc3-nophon baseline when trained on the 100 utterance corpus, indicating a synergistic interaction (rather than redundancy) between phonotactics and stress for small inputs.

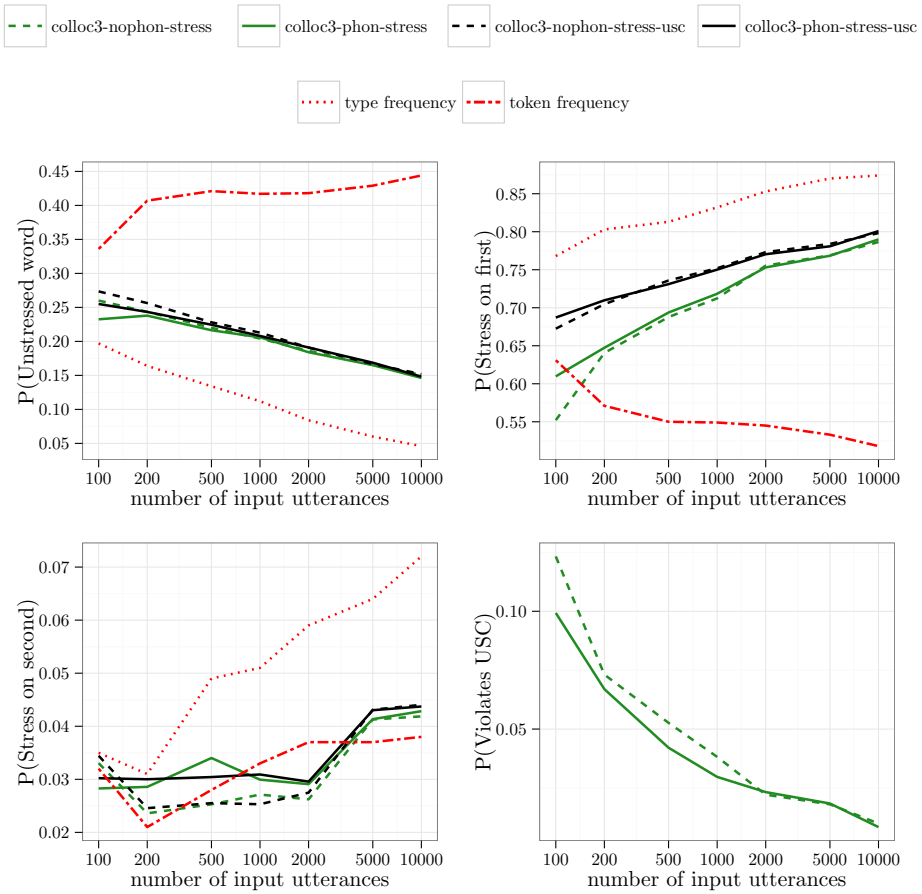


Figure 5.5: Evolution of the knowledge the learner acquires on the Alex corpus. The red dotted line indicates the empirical type distribution of a specific pattern, and the double-dashed line the empirical token distribution. Top-Left: Stress-initial pattern, Top-Right: Unstressed Words, Bottom-Left: Stress-second pattern, Bottom-Right: Patterns that violate the USC.

This effect disappears starting from around 1000 utterances; for inputs of size 1000 and larger, the net-gain of stress drops from roughly 10% to 3–4% improvement over and above what is contributed by phonotactics. That is, while we did not notice any relationship between input size and impact of stress cues for the colloc3-nophon model, we do see such an interaction for the combination of phonotactics and stress cues which, taken together, lead to a larger relative gain in performance on smaller inputs than on large ones.

5.5.3 *Acquisition of stress patterns*

In addition to acquiring a lexicon, the Bayesian learner acquires knowledge about the possible stress patterns of English words. The fact that this knowledge is explicitly represented through the PCFG rules and their probabilities that define the lexical generator allows us to study the

generalisations about stress the model actually acquires. While Doyle and Levy (2013) suggest carrying out such an analysis, they restrict themselves to estimating the fraction of stress patterns in the segmented output. As shown in Table 5.3, however, the type and token distributions of stress patterns can differ substantially. I therefore investigate the stress preferences acquired by the learner by examining the probabilities assigned to the different expansions of rule (5.21), aggregating the probabilities of the individual rules into patterns. For example, the rules $\underline{\text{Word}} \rightarrow \text{SSyll}\,(\,\text{USyll}\,)^{\{0,3\}}$ correspond to the pattern "Stress on the first syllable", whereas the rules $\underline{\text{Word}} \rightarrow \text{USyll}^{\{1,4\}}$ correspond to the pattern "Unstressed word". By computing the respective probabilities, one gets the overall probability assigned by a learner to the pattern.

Figure 5.5 provides this information for several different rule patterns. Additionally, these plots include the empirical type (red dotted) and token proportions (red double-dashed) for the input corpus. Note how for the two major patterns, all models successfully track the type, rather than the token frequency, correctly developing a preference for stress-initial over unstressed words, despite the comparable token frequency of these two patterns. This is compatible with a recent proposal by Thiessen and Saffran (2007), who argue that infants infer the stress pattern over their lexicon.

For Bayesian models such as the ones examined in this chapter or Goldwater et al. (2009)'s, there is no need to pre-specify that the distribution ought to be learned over types rather than tokens, as the models automatically interpolate between type and token statistics according to the properties of their input (Goldwater et al., 2006). In the adaptor grammar model, the fact that the left-hand side of the rules responsible for the stress-pattern (rule 5.21) ensures that the probability of a stress pattern will be estimated not directly from the number of tokens with which each pattern occurs. Instead, it will be estimated from the number of tables with this stress pattern in the Chinese Restaurant franchise representation where it is common to have very few tables with the identical label. This is discussed in more detail in section 2.5.2.1 on page 62.

In addition, a Bayesian framework provides a simple answer to the question of how a learner might identify the role of stress in its language without already having acquired at least some words. By combining different kinds of cues, e.g. distributional, phonotactic and prosodic, in a principled manner a Bayesian learner can jointly segment its input and learn the appropriate role of each cue, without having to pre-specify specific preferences that might differ across languages.

The iambic rule pattern that puts stress on the second syllable is much more infrequent on a token level. All models track this low token frequency, underestimating the type frequency of this pattern by a fair amount. This suggests that learning this pattern correctly requires con-

siderably more input than for the other patterns. Indeed, the iambic pattern is known to pose problems for infants when they start using stress as an effective cue. It is only from roughly 10 months of age that infants successfully segment iambic words (Jusczyk et al., 1999b). Not surprisingly, the USC doesn't aid in learning about this pattern because it is completely silent on where stress might fall (and does not noticeably improve segmentation performance to begin with).

Finally, one can also investigate whether the models that lack the USC nevertheless learn that words contain at most one lexically stressed syllable. The bottom-right graph in Figure 5.5 plots the probability assigned by the models to patterns that violate the USC. This includes, for example, the rules Word → SyllS SyllS and Word → SyllS SyllU SyllS. Note how the probabilities assigned to these rules approaches zero, indicating that the learner becomes more certain that there are no words that contain more than one syllable with lexical stress. As I argued above, this suggests that a Bayesian learner can acquire the USC from a modest amount of data — it will properly infer that the unnatural patterns are simply not supported by the input

To summarize, by examining the internal state of the Bayesian learners one can characterize how their knowledge about the stress preferences of their languages develops, rather than merely measuring how well they perform word segmentation. We find that the iambic pattern that has been observed to pose problems for infant learners also is harder for the Bayesian learner to acquire, arguably due to its extremely low token-frequency.

5.6 CONCLUSION AND FUTURE WORK

I have presented adaptor grammar models of word segmentation that are able to take advantage of stress cues and are able to learn from phonemic input. I find that phonotactics and stress interact in interesting ways, and that stress cues makes a stable contribution to existing word segmentation models, improving their performance by 4-6% token f-score.

I also find that the USC introduced by Yang (2004) need not be prebuilt into a model but can be acquired by a Bayesian learner from the data. Similarly, I directly investigate the stress preferences acquired by the models and find that for stress-initial and unstressed words, they track type rather than token frequencies. The rare stress-second pattern seems to require more input to be properly acquired, which is compatible with infant development data.

There are several directions in which future work can extend on my findings. An important goal to be addressed in the future is to evaluate segmentation models on typologically different languages and to study the relative usefulness of different cues cross-lingually. For example, lan-

guages such as French lack lexical stress; it would be interesting to know whether in such a case, phonotactic (or other) cues are more important.

Relatedly, there always is a risk that artificially created data masks the complexity exhibited by real speech. For example, the pronunciation variation that words are subject to in real speech is likely to affect how individual occurrences of a word are stressed as well. An example for this problem is given in the next chapter, and future work should use data directly derived from the acoustic signal to account for contextual effects, rather than using dictionary look-up or other heuristics. In using the Alex corpus, for which good quality audio is available, I have taken a first step in this direction, and indeed preliminary results building on this chapter are reported on in Pate et al. (shed).

6

A JOINT MODEL OF WORD SEGMENTATION AND PHONOLOGICAL VARIATION

Word-final /t/-deletion refers to a common phenomenon in spoken English where words such as /wɛst/ "west" are pronounced as [wɛs] "wes" in certain contexts. Phonological variation like this is common in naturally occurring speech. Current computational models of unsupervised word segmentation usually assume idealized input that is devoid of these kinds of variation. I extend a non-parametric model of word segmentation by adding phonological rules that map from underlying forms to surface forms to produce a mathematically well-defined joint model as a first step towards handling variation and segmentation in a single model. I analyze how my model handles /t/-deletion on a large corpus of transcribed speech, and show that the joint model can perform word segmentation and recover underlying /t/s. I find that Bigram dependencies are important for performing well on real data and for learning appropriate deletion probabilities for different contexts.

6.1 INTRODUCTION

Computational models of word segmentation try to solve one of the first problems language learners have to face: breaking an unsegmented stream of sound segments into individual words. Currently, most such models assume that the input consists of sequences of phonemes with no pronunciation variation across different occurrences of the same word type. In this chapter I describe an extension of the Bayesian models of Goldwater et al. (2009) that incorporates phonological rules to "explain away" surface variation. As a concrete example, I focus on word-final /t/-deletion in English, although our approach is not limited to this case. I choose /t/-deletion because it is a very common and well-studied phenomenon (see (Coetzee, 2004, chapter 5) for a review) and segmental deletion is an interesting test-case for our architecture. Recent work has found that /t/-deletion (among other things) is indeed common in child-directed speech (CDS) and, importantly, that its distribution is similar to that in adult-directed speech (ADS) Dilley et al. (to appear). This justifies our using ADS to evaluate our model, as discussed below.

Our experiments are consistent with long-standing and recent findings in linguistics, in particular that /t/-deletion heavily depends on the immediate context and that models ignoring context work poorly on real data. I also examine how well the models identify the probability of /t/-deletion in different contexts. I find that models that capture bigram dependencies between underlying forms provide considerably

more accurate estimates of those probabilities than corresponding unigram or "bag of words" models of underlying forms.

In section 6.2 I discuss related work on handling variation in computational models and on /t/-deletion. Section 6.3 describes my computational model and section 6.4 discusses its performance for recovering deleted /t/s. I look at both a situation where word boundaries are pre-specified and only inference for underlying forms has to be performed; and the problem of jointly finding the word boundaries and recovering deleted underlying /t/s. Section 6.5 discusses my findings, and section 6.6 concludes with directions for further research.

6.2 BACKGROUND AND RELATED WORK

6.2.1 */t/-deletion*

/t/-deletion has received a lot of attention within linguistics, and I point the interested reader to (Coetzee, 2004, Chapter 5) for a thorough review. Briefly, the phenomenon is as follows: word-final instances of /t/ may undergo deletion in natural speech, such that /wɛst/ "west" is actually pronounced as [wɛs] "wes".[1] While the frequency of this phenomenon varies across social and dialectal groups, within groups it has been found to be robust, and the probability of deletion depends on its phonological context: a /t/ is more likely to be dropped when followed by a consonant than a vowel or a pause, and it is more likely to be dropped when following a consonant than a vowel as well. Two recent publications are of direct relevance to this chapter.

Dilley et al. (to appear) study word-final variation in stop consonants in CDS, the kind of input one ideally would like to evaluate the models on. They find that "infants largely experience statistical distributions of non-canonical consonantal pronunciation variants [including deletion] that mirror those experienced by adults." This both directly establishes the need for computational models to handle this dimension of variation, and justifies my choice of using ADS for evaluation, as mentioned above.

6.2.2 *Prior modeling work*

The work of Elsner et al. (2012) is most closely related to my goal of building a model that handles variation. They propose a pipe-line architecture involving two separate generative models, one for word-segmentation and one for phonological variation. They model the mapping to surface forms using a probabilistic finite-state transducer which is able to represent a large class of phonological rules such as context-dependent deletion or insertion of particular segments.

1 Following the convention in phonology, I give underlying forms within "/. . . /" and surface forms within "[. . .]".

Their approach first applies the Bigram model of Goldwater et al. (2009) (see Figure 2.10 on 48) to a corpus which exhibits pronunciation variation. From this initial segmentation, the parameters of the transducer and a clustering of the types in the initial segmentation into clusters of 'underlying forms' is determined using Viterbi EM (Spitkovsky et al., 2010). Using this clustering, a second version of the corpus is generated by replacing every token with the underlying form of its corresponding cluster, reducing the variation. Then, they apply the word segmentation model again to the modified corpus. On an artificially created version of the Brent-Bernstein-Ratner corpus (Brent, 1999), they demonstrate that their pipeline approach leads to more accurate segmentation.

Elsner et al. (2013) extends this work by performing segmentation and clustering jointly, using essentially the same architecture. This results in a segmentation model that can handle virtually arbitrary pronunciation variation. However, as they point out, joint inference under this model is infeasible and they resort to several heuristics to perform approximate inference. In this chapter, I illustrate an alternative research strategy, starting with a single well-studied example of phonological variation. This permits me to develop a joint generative model for both word segmentation and variation which can form the basis for specific explorations – here, how a deletion phenomenon impacts segmentation.

An earlier approach that is close to the technical idea underlying my approach is Naradowsky and Goldwater (2009). Their work was motivated by the observation that Goldwater (2007)'s model for English stem-suffix morphology cannot correctly analyze forms such as *baking* where the stem *bake* loses its final *e*. This is because Goldwater's model can only concatenate stems and suffixes without changing them. Naradowsky and Goldwater (2009)'s model, in contrast, allows the output of the concatenation to undergo limited amounts of changes at the juncture by assuming that a *spelling rule* applies to the concatenation of stem and suffix. Here, the rule is "delete a stem-final *e* if the suffix begins with an *i*".

This is achieved by adding to the model a huge (but finite) number of possible rewrite rules. Crucially, each of these rules is 'reversible' in the sense that given an observed unsegmented word such as *baking* and an analysis of this into 'surface' stem and suffix such as *bak-ing*, each possible spelling rule determines exactly one underlying form. Thus, the spelling rule "delete a stem-final *e* if the suffix begins with an *i*" determines as the underlying stem *bake* whereas the rule "add a *k* to the stem if the suffix begins with an *i*" determines as underlying stem *ba*. The probabilities of these rules are learned jointly with the stem-suffix analyses and, indeed, this model performs better than the original morphology model as it handles case such as *baking* correctly. As will become clear below, my own model exploits the same idea – adding a

rule that can account for differences between the actual observations and the posited observation. However, as I consider segmenting entire utterances into words rather than the considerably simpler task of segmenting words into exactly one stem and one (possibly empty) suffix, only a single such rule will be considered in this chapter.

Coetzee and Kawahara (2013) provide a computational study of (among other things) /t/-deletion within the framework of Harmonic Grammar. They do not aim for a joint model that also handles word segmentation, however, and rather than training their model on an actual corpus, they evaluate on constructed lists of examples, mimicking frequencies of real data. Overall, my findings agree with theirs, in particular that capturing the probability of deletion in different contexts does not automatically result in good performance for recovering individual deleted /t/s. I will come back to this point in the discussion at the end of the chapter.

6.3 THE COMPUTATIONAL MODEL

My models build on the Unigram and the Bigram model introduced in Goldwater et al. (2009) and reviewed in more detail in Chapter 2 (see Figures 2.3 on page 35 and 2.10 on page 48). Figure 6.1 shows the graphical model for the joint Bigram model (the Unigram case is trivially recovered by generating the $U_{i,j}$s directly from G rather than from $H_{U_{i,j-1}}$). Figure 6.2 gives the mathematical description of the graphical model and Table 6.1 provides a key to the variables of my model.

The model generates a latent sequence of *underlying word-tokens* $U_1, \ldots, U_n$. Each word token is itself a non-empty sequence of segments or phonemes, and each U_j corresponds to an underlying word form, prior to the application of any phonological rule. This generative process is repeated for each utterance i, leading to multiple utterances of the form $U_{i,1}, \ldots, U_{i,n_i}$ where n_i is the number of words in the i^{th} utterance, and $U_{i,j}$ is the j^{th} word in the i^{th} utterance. Each utterance is padded by an observed utterance boundary symbol \$ to the left and to the right, hence $U_{i,0} = U_{i,n_i+1} = \$$.[2] Each $U_{i,j+1}$ is generated conditionally on its predecessor $U_{i,j}$ from $H_{U_{i,j}}$, as shown in the first row of the lower plate in Figure 6.1. Each H_w is a distribution over the possible words that can follow a token of w and G is a global distribution over possible words, used as back-off for all H_w. Just as in Goldwater et al. (2009), H is drawn from a Dirichlet Process (DP) with base distribution P_{lex} and concentration parameter α_0, and the word type specific distributions H_w are drawn from a $DP(L, \alpha_1)$, resulting in a hierarchical DP model (Teh et al., 2006).

The base distribution P_{lex} functions as a lexical generator, defining a prior distribution over possible words. In principle, P_{lex} can incorporate arbitrary prior knowledge about possible words, for example syllable

2 Each utterance terminates as soon as a \$ is generated, thus determining the number of words n_i in the i^{th} utterance. See Goldwater et al. (2009) for discussion.

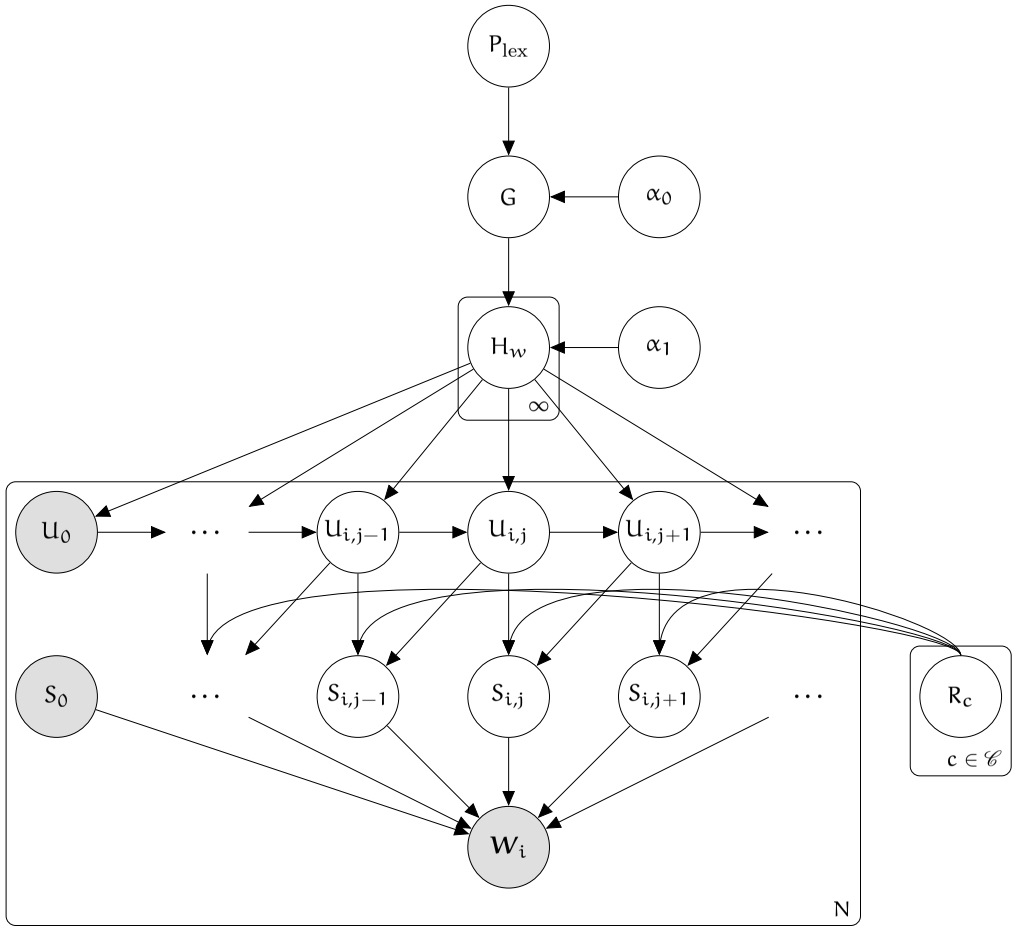


Figure 6.1: Graphical model for the joint model of word segmentation and phonological variation. The generative process mimics the intuitively plausible idea of generating underlying forms from some kind of syntactic model (here, a Bigram language model) and then mapping the underlying form to an observed surface-form through the application of a phonological rule component, here represented by the collection of rule probabilities R_c. Mathematical description in Figure 6.2.

$$
\begin{aligned}
\alpha_0 &\sim \mathsf{Gamma}(0.001, 0.001) \\
\alpha_1 &\sim \mathsf{Gamma}(0.001, 0.001) \\
G \mid \alpha_0 &\sim \mathsf{DP}(\alpha_0, P_{lex}) \\
H_w \mid G, \alpha_1 &\sim \mathsf{DP}(\alpha_1, G) \\
R_c &\sim \mathsf{Beta}(1, 1) \\
U_{i,0} &= \$ \\
S_{i,0} &= \$ \\
U_{i,j+1} \mid U_{i,j}, H_{U_{i,j}} &\sim H_{U_{i,j}} \\
S_{i,j} \mid U_{i,j}, U_{i,j+1}, \mathbf{R} &\sim P_R(\cdot \mid U_{i,j}, U_{i,j+1}) \\
W_i \mid S_{i,1}, \ldots, S_{i,n_i} &= \textsc{Concat}(S_{i,0}, \ldots, S_{i,n_i})
\end{aligned}
$$

Figure 6.2: Definition of Bigram model with phonological variation. The mapping distribution P_R is explained in the text below. Concat stands for concatenation without word boundaries and n_i refers to the number of words in utterance i.

Variable	Explanation
P_{lex}	base distribution over possible words
G	back-off distribution over words
H_w	distribution over words following w
$U_{i,j}$	underlying form at position j in utterance i, a word
$S_{i,j}$	surface realization of $U_{i,j}$, a word
R_c	/t/-deletion probability in context c
W_i	observed segments for i^{th} utterance

Table 6.1: Key for the variables in Figure 6.1 and Figure 6.2.

structure (cf. Johnson (2008b)). Following the experimental findings of Chapter 3, I use a simpler possible word constraint that only rules out sequences that lack a vowel (see Figure 2.5 on page 39 for more discussion). While this is clearly a simplification it is a plausible assumption for English data.

6.3.1 *Modeling variation*

Instead of generating the observed sequence of segments W directly by concatenating the underlying forms as in Goldwater et al. (2009), I map each $U_{i,j}$ to a corresponding surface-form $S_{i,j}$ by a probabilistic rule component P_R.

This rule component is a conditional distribution $P_R(S \mid U)$ over surface forms S given a particular underlying form U. The range of S is determined by the phonological processes that are available to the model. Here, the phonological phonological processes only include a rule for deleting word-final /t/s but in principle, P_R can be used to encode a wide variety of phonological rules.

Thus, in the /t/-deletion model for a given underlying form u the possible surface realizations depend on whether or not u ends in a /t/. If this is the case, S ranges over both u and $\textsc{DelF}(u)$ where $\textsc{DelF}(u) = u_{1:|u|-1}$; in other words, $\textsc{DelF}(u)$ is the result of deleting the final segment of u.

If u does not end in a /t/, S only ranges over u. This reflects that the model only assumes a single process of variation and that underlying forms that cannot exhibit this variation are deterministically mapped to their underlying form.

I consider three kinds of contexts on which a rule's probability of applying depends:

1. a *uniform* context that applies to every word-final position
2. a *right* context that also considers the following segment

3. a *left-right* context that additionally takes the preceeding segment into account

For each possible context $c \in \mathscr{C}$ there is a random variable R_c which stands for the probability of the rule applying in this context. I refer to concrete values of this probability with ρ_c.

Writing contexts in the notation familiar from generative phonology Chomsky and Halle (1968), the model can be seen as implementing the following *probabilistic* rules under the different assumptions:[3]

uniform	/t/	→	∅	/	___]$_{\text{word}}$
right	/t/	→	∅	/	___]$_{\text{word}}$ β
left-right	/t/	→	∅	/	α ___]$_{\text{word}}$ β

β ranges over V(owel), C(onsonant) and $ (utterance-boundary), and α over V and C. I define a function CONT that maps a pair of adjacent underlying forms $U_{i,j}, U_{i,j+1}$ to the context of the final segment of $U_{i,j}$.

For example, CONT(/wɛst/,/əv/) returns "C ___]$_{\text{word}}$ V" in the *left-right* setting, or simply "___]$_{\text{word}}$" in the *uniform* setting. CONT returns a special NOT context if $U_{i,j}$ doesn't end in a /t/. I stipulate that $\rho_{\text{NOT}} = 0.0$. Then one can define P_R as follows:

$$P_R(\text{DELFINAL}(u) \mid u, r)) = \rho_{\text{CONT}(u,r)} \tag{6.1}$$

$$P_R(u \mid u, r) = 1 - \rho_{\text{CONT}(u,r)} \tag{6.2}$$

Depending on the context setting used, the model includes one (*uniform*), three (*right*) or six (*left-right*) /t/-deletion probabilities ρ_c. I place a uniform $\text{Beta}(1,1)$ prior on each of those so as to learn their values in the LEARN-ρ experiments below.

Finally, the observed unsegmented utterances W_i are generated by concatenating all $S_{i,j}$ using the function CONCAT which simply concatenates all underlying words without boundaries. The entire input is made up of several unsegmented utterances – as usual, I assume that utterance boundaries are observed and need not be inferred.

6.3.2 *Modeling intuition*

I briefly comment on the central intuition of this model, i.e. why it can infer underlying from surface forms. Bayesian word segmentation models try to compactly represent the observed data in terms of a small set of units (word types) and a short analysis (a small number of word tokens). Phonological rules such as /t/-deletion can potentially "explain away" an observed surface type such as [wɛs] in terms of the underlying type /wɛst/ which is independently needed for surface tokens of [wɛst].

3 For *right* there are three and for *left-right* six different rules, one for every instantiation of the context-template.

Thus, the /t/→ ∅ rule makes possible a smaller lexicon for a given number of surface tokens.

For this to work, of course, there needs to be sufficient evidence for the underlying type in the observed data, and the "probability cost" incurred by each application of a phonological rule must not outweigh the savings made through the smaller lexicon.

Obviously, human learners have access to additional cues, such as the meaning of words, knowledge of phonological similarity between segments and so forth. One of the advantages of an explicitly defined generative model such as ours is that it is straight-forward to gradually extend it by adding more cues, as I point out in the discussion.

6.3.3 *Inference*

Just as for the Goldwater et al. (2009) segmentation models, exact inference is infeasible for the joint model. I extend the collapsed Gibbs breakpoint-sampler of Goldwater et al. (2009), reviewed in detail in chapter 2, to perform inference for the extended models. For details such as how to calculate the Bigram probabilities in Figure 6.3, see either chapter 2 or the original paper. Here I focus on the required changes to the sampler so as to perform inference under the richer model. I consider the case of a single surface string W, so I drop the i-index in the following discussion.

Knowing W, the problem is to recover the underlying forms $U_1, \ldots, U_n$ and the surface forms $S_1, \ldots, S_n$ for unknown n. A major insight in Goldwater's work is that rather than sampling over the latent word variables that define the segmentation directly (the number of which we don't even know), one can instead perform Gibbs sampling over a set of boundary variables $B_1, \ldots, B_{|W|-1}$ that jointly determine the values for the variables of interest where $|W|$ is the length of the surface string W. For the original segmentation model, this is discussed in more detail in Chapter 2, see in particular Figure 2.7 on page 43.

For the /t/-deletion model, each $B_j \in \{0, 1, t\}$, where $B_j = 0$ indicates absence of a word boundary, $B_j = 1$ indicates presence of a boundary and $B_j = t$ indicates presence of a boundary with a preceding underlying /t/. The relation between the B_j and the $S_1, \ldots, S_n$ and $U_1, \ldots, U_n$ is illustrated in Figure 6.4. The required sampling equations are given in Figure 6.3.

6.4 EXPERIMENTS

6.4.1 *The data*

I am interested in how well the model handles /t/-deletion in real data. Ideally, we'd evaluate it on CDS but as of now, I know of no available large enough corpus of accurately hand-transcribed CDS.

$$
\begin{aligned}
P(b_j = 0 \mid h^{-j}) &\propto P(w_{12,u} \mid w_{l,u}, h^{-j}) \times P_r(w_{12,s} \mid w_{12,u}, w_{r,u}) \\
&\quad \times P(w_{r,u} \mid w_{12,u}, h^{-j} \cup \langle w_{l,u}, w_{12,u} \rangle) \\
P(b_j = t \mid h^{-j}) &\propto P(w_{1,t} \mid w_{l,u}, h^{-j}) \times P_r(w_{1,s} \mid w_{1,t}, w_{2,u}) \\
&\quad \times P(w_{2,u} \mid w_{1,t}, h^{-j} \cup \langle w_{l,u}, w_{1,t} \rangle) \times P_r(w_{2,s} \mid w_{2,u}, w_{r,u}) \\
&\quad \times P(w_{r,u} \mid w_{2,u}, h^{-j} \cup \langle w_{l,u}, w_{1,t} \rangle \cup \langle w_{1,t}, w_{2,u} \rangle) \\
P(b_j = 1 \mid h^{-j}) &\propto P(w_{1,s} \mid w_{l,u}, h^{-j}) \times P_r(w_{1,s} \mid w_{1,s}, w_{2,u}) \\
&\quad \times P(w_{2,u} \mid w_{1,s}, h^{-j} \cup \langle w_{l,u}, w_{1,s} \rangle) \times P_r(w_{2,s} \mid w_{2,u}, w_{r,u}) \\
&\quad \times P(w_{r,u} \mid w_{2,u}, h^{-j} \cup \langle w_{l,u}, w_{1,s} \rangle \cup \langle w_{1,s}, w_{2,u} \rangle)
\end{aligned}
$$

Figure 6.3: Sampling equations for the Gibbs sampler, see figure 6.4 for illustration. $b_j = 0$ corresponds to no boundary at this position, $b_j = t$ to a boundary with a preceeding underlying /t/ and $b_j = 1$ to a boundary with no additional underlying /t/. I use h^{-j} for the statistics determined by all but the j^{th} position, including the specific seating arrangement (see Chapter 2 for details). I use $h^{-j} \cup \langle r, l \rangle$ for the result of updating the previous statistics with an additional count of the bigram $\langle r, l \rangle$. $P(w \mid l, h)$ refers to the bigram probability of $\langle l, w \rangle$ given h. See equations 2.26 on page 49 for the details of calculating these bigram probabilities. For more details about the sampler, see the discussion in chapter 2. The distribution that calculates the underlying-to-surface mapping probabilities P_R is defined in the text.

Instead, I used the Buckeye Corpus (Pitt et al., 2007) for my experiments, a large ADS corpus of interviews with English speakers that have been transcribed with relatively fine phonetic detail, with /t/-deletion among the things manually annotated. Pointing to the recent work by Dilley et al. (to appear) I want to emphasize that the statistical distribution of /t/-deletion has been found to be similar for ADS and CDS, at least for read speech.

I automatically derived a corpus of 285,792 word tokens across 48,795 utterances from the Buckeye Corpus by collecting utterances across all interviews and heuristically splitting utterances at speaker-turn changes and indicated silences.

The Buckeye corpus lists for each word token a manually transcribed pronunciation in context as well as its canonical pronunciation as given in a pronouncing dictionary. As input to my model, I use the canonical pronunciation unless the pronunciation in context indicates that the final /t/ has been deleted in which case I also delete the final /t/ of the canonical pronunciation Figure 6.5 shows an example from the Buckeye Corpus, indicating how the original data, a fully idealized version and the derived input that takes into account /t/-deletions looks like.

Overall, /t/-deletion is a quite frequent phenomenon with roughly 29% of all underlying /t/s being dropped. The probabilities become

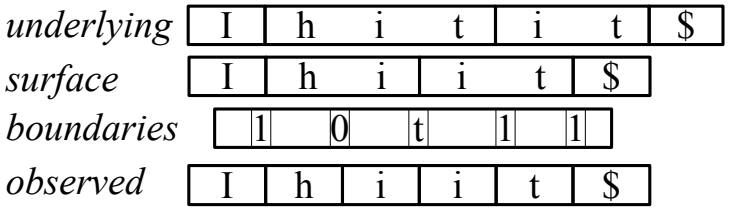


Figure 6.4: The relation between the observed sequence of segments (bottom), the boundary variables $\mathtt{b}_1, \ldots, \mathtt{b}_{|\mathcal{W}|-1}$ the Gibbs sampler operates over (in squares), the latent sequence of surface forms and the latent sequence of underlying forms. When sampling a new value for $\mathtt{b}_3 = \mathtt{t}$, the different word-variables in figure 6.3 are: $w_{12,u}$=$w_{12,s}$=*hiit*, $w_{1,t}$=*hit* and $w_{1,s}$=*hi*, $w_{2,u}$=$w_{2,s}$=*it*, $w_{l,u}$=*I*, $w_{r,u}$=*$*. Note that one needs a boundary variable at the end of the utterance as there might be an *underlying* /t/ at this position as well. The final boundary variable is set to $\mathtt{1}$, not $\mathtt{t}$, because the /t/ in *it* is observed.

orthographic	I don't intend to
transcript	/aɪ ɾ oʊ n ɪ n t ɛ n d ə/
idealized	/aɪ d oʊ n t ɪ n t ɛ n d t ʊ/
t-drop	/aɪ d oʊ n ɪ n t ɛ n d t ʊ/

Figure 6.5: An example fragment from the Buckeye-corpus in orthographic form, the fine transcript available in the Buckeye corpus, a fully idealized pronunciation with canonical dictionary pronunciations and the version of the data with dropped /t/s.

more peaked when looking at finer context; see Table 6.3 for the empirical distribution of /t/-dropping for the six different contexts of the *left-right* setting.

6.4.2 *Recovering deleted /t/s, given word boundaries*

In this set of experiments I am interested in how well the model recovers /t/s when it is provided with the gold word boundaries. This allows us to investigate the strength of the statistical signal for the deletion rule without confounding it with the word segmentation performance, and to see how the different contextual settings *uniform*, *right* and *left-right* handle the data. Concretely, for the example in Figure 6.5 this means that one tells the model that there are boundaries between /aɪ/, /doʊn/, /ɪntɛnd/, /tu/ and /liv/ but one does not tell it whether or not these words end in an underlying /t/. Even in this simple example, there are 5 possible positions for the model to posit an underlying /t/. I evaluate the model in terms of f-score, the harmonic mean of recall (the fraction of underlying /t/s the model correctly recovered) and precision (the fraction of underlying /t/s the model predicted that were correct).

		uniform	*right*	*left-right*
Unigram	LEARN-ρ	56.52	39.28	23.59
	GOLD-ρ	62.08	60.80	66.15
Bigram	LEARN-ρ	60.85	62.98	77.76
	GOLD-ρ	69.06	69.98	73.45

Table 6.2: F-score of recovered /t/s with known word boundaries on real data for the three different context settings, averaged over two runs (all standard errors below 2%). Note how the Unigram model always suffers in the LEARN-ρ condition whereas the Bigram model's performance is actually best for LEARN-ρ in the *left-right* setting.

In these experiments, I ran a total of 2500 iterations with a burnin of 2000. I collect samples with a lag of 10 for the last 500 iterations and perform *maximum marginal decoding* over these samples Johnson and Goldwater (2009), as well as running two chains so as to get an idea of the variance.[4]

I am also interested in how well the model can infer the rule probabilities from the data, that is, whether it can learn values for the different ρ_c parameters. I compare two settings, one where inference for these parameters is performed assuming a uniform Beta prior on each ρ_c (LEARN-ρ) and one where the model is provided with the empirical probabilities for each ρ_c as estimated off the gold-data (GOLD-ρ), e.g., for the *uniform* condition 0.29. The results are shown in Table 6.2.

Best performance for both the Unigram and the Bigram model in the GOLD-ρ condition is achieved under the *left-right* setting, in line with the standard analyses of /t/-deletion as primarily being determined by the preceding and the following context. For the LEARN-ρ condition, the Bigram model still performs best in the *left-right* setting but the Unigram model's performance drops in all settings and is now worst in the *left-right* and best in the *uniform* setting.

In fact, comparing the inferred probabilities to the "ground truth" indicates that the Bigram model estimates the true probabilities more accurately than the Unigram model, as illustrated in Table 6.3 for the *left-right* setting. The Bigram model somewhat overestimates the probability for all post-consonantal contexts but the Unigram model severely underestimates the probability of /t/-deletion across all contexts.

4 As manually setting the hyper parameters for the DPs in the model proved to be complicated and may be objected to on principled grounds, I perform inference for them under a vague $\mathsf{Gamma}(0.01, 0.01)$ prior, as suggested by Teh et al. (2006) and Johnson and Goldwater (2009), using my own implementation of a slice sampler (Neal, 2003).

	C_C	C_V	C_$	V_C	V_V	V_$
empirical	0.62	0.42	0.36	0.23	0.15	0.07
Unigram	0.41	0.33	0.17	0.07	0.05	0.00
Bigram	0.70	0.58	0.43	0.17	0.13	0.06

Table 6.3: Inferred rule-probabilities for different contexts in the left-right setting from one of the runs. "C_C" stands for the context where the deleted /t/ is preceded and followed by a consonant, "V_$" stands for the context where it is preceded by a vowel and followed by the utterance boundary. Note how the Unigram model severely underestimates and the Bigram model slightly over-estimates the probabilities.

		uniform	right	left-right
Unigram	LEARN-ρ	94.35	23.55 (+)	63.06
	GOLD-ρ	94.45	94.20	91.83
Bigram	LEARN-ρ	92.72	91.64	88.48
	GOLD-ρ	92.88	92.33	89.32

Table 6.4: F-score of /t/-recovery with known word boundaries on artificial data, each condition tested on data that corresponds to the assumption, averaged over two runs (standard errors less than 2% except (+) = 3.68%)).

6.4.3 *Artificial data experiments*

To test my Gibbs sampling inference procedure, I ran it on artificial data generated according to the model itself. If the inference procedure fails to recover the underlying /t/s accurately in this setting, one should not expect it to work well on actual data. I generated the artificial data as follows.

I transformed the sequence of canonical pronunciations in the Buckeye corpus (which I take to be underlying forms here) by randomly deleting final /t/s using empirical probabilities as shown in Table 6.3 to generate a sequence of artificial surface forms that serve as input to the models. I did this for all three context settings, always estimating the deletion probability for each context from the gold-standard. The results of these experiments are given in table 6.4. Interestingly, performance on these artificial data is considerably better than on the real data. In particular the Bigram model is able to get consistently high F-scores for both the LEARN-ρ and the GOLD-ρ setting. For the Unigram model, we again observe the severe drop in the LEARN-ρ setting for the *right* and *left-right* settings although it does remarkably well in the *uniform* setting, and performs well across all settings in the GOLD-ρ condition. I take this to show that the inference algorithm is in fact working as expected.

	Unigram	Bigram
LEARN-ρ	33.58	55.64
GOLD-ρ	55.92	57.62

Table 6.5: /t/-recovery F-scores when performing joint word segmention in the *left-right* setting, averaged over two runs (standard errors less than 2%). See Table 6.6 for the corresponding segmentation F-scores.

6.4.4 *Segmentation experiments*

Finally, I am also interested to learn how well one can do word segmentation and underlying /t/-recovery jointly. Again, I look at both the LEARN-ρ and GOLD-ρ conditions but focus on the *left-right* setting as this worked best in the experiments above. For these experiments, I perform simulated annealing throughout the initial 2000 iterations, gradually cooling the temperature from 5 to 1, following the observation by Goldwater et al. (2009) that without annealing, the Bigram model gets stuck in sub-optimal parts of the solution space early on. During the annealing stage, I prevent the model from performing inference for underlying /t/s so that the annealing stage can be seen as an elaborate initialization scheme, and then have it perform joint inference for the remaining 500 iterations, evaluating on the last sample and averaging over two runs. As neither the Unigram nor the Bigram model performs "perfect" word segmentation, one expects to see a degradation in /t/-recovery performance and this is what one finds indeed. To give an impression of the impact of /t/-deletion, I also report numbers for running only the segmentation model on the Buckeye data with no deleted /t/s and on the data with deleted /t/s. The /t/-recovery scores are given in Table 6.5 and segmentation scores in Table 6.6. Again the Unigram model's /t/-recovery score degrades dramatically in the LEARN-ρ condition. Looking at the segmentation performance this isn't too surprising: the Unigram model's poorer token F-score, the standard measure of segmentation performance on a word token level, suggests that it misses many more boundaries than the Bigram model to begin with and, consequently, can't recover any potential underlying /t/s at these boundaries. Also note that in the GOLD-ρ condition, the joint Bigram model performs almost as well on data with /t/-deletions as the word segmentation model on data that includes no variation at all.

The generally worse performance of handling variation as measured by /t/-recovery F-score when performing joint segmentation is consistent with the finding of Elsner et al. (2012) who report considerable performance drops for their phonological learner when working with induced boundaries (note, however, that their model does not perform joint inference, rather the induced boundaries are given to their phonological learner as ground-truth).

	Unigram	Bigram
LEARN-ρ	54.53	72.55 (2.3%)
GOLD-ρ	54.51	73.18
NO-ρ	54.61	70.12
NO-VAR	54.12	73.99

Table 6.6: Word segmentation F-scores for the /t/-recovery F-scores in Table 6.5 averaged over two runs (standard errors less than 2% unless given). NO-ρ are scores for running just the word segmentation model with no /t/-deletion rule on the data that includes /t/-deletion, NO-VAR for running just the word segmentation model on the data with no /t/-deletions.

6.5 DISCUSSION

There are two interesting findings from my experiments. First of all, we find a much larger difference between the Unigram and the Bigram model in the LEARN-ρ condition than in the GOLD-ρ condition. I suggest that this is due to the Unigram model's lack of dependencies between underlying forms, depriving it of an important source of evidence. Bigram dependencies provide additional evidence for underlying /t/ that are deleted on the surface, and because the Bigram model identifies these underlying /t/ more accurately, it can also estimate the /t/ deletion probability more accurately.

For example, /t/ dropping in "don't you" yields surface forms "don you". Because the word bigram probability $\mathsf{P}(\text{you} \mid \text{don't})$ is high, the bigram model prefers to analyse surface "don" as underlying "don't". The Unigram model does not have access to word bigram information so the underlying forms it posits are less accurate (as shown in Table 2), and hence the estimate of the /t/-deletion probability is also less accurate.

When the probabilities of deletion are pre-specified the Unigram model performs better but still considerably worse than the Bigram model when the word boundaries are known, suggesting the importance of non-phonological contextual effects that the Bigram model but not the Unigram model can capture. This suggests that for example word predictability in context might be an important factor contributing to /t/-deletion.

The other striking finding is the considerable drop in performance between running on naturalistic and artificially created data. This suggests that the natural distribution of /t/-deletion is much more complex than can be captured by statistics over the phonological contexts I examined. Following Guy (1991), a finer-grained distinction for the preceding segments might address this problem.

Yet another suggestion comes from the recent work in Coetzee and Kawahara (2013) who claim that "[a] model that accounts perfectly for the overall rate of application of some variable process therefore does not necessarily account very well for the actual application of the process to individual words." They argue that in particular the extremely high deletion rates typical of high frequency items aren't accurately captured when the deletion probability is estimated across all types. A look at the error patterns of the model on a sample from the Bigram model in the LEARN-ρ setting on the naturalistic data suggests that this is in fact a problem. For example, the word "just" has an extremely high rate of deletion with $\frac{1746}{2442} = 0.71\%$. While many tokens of "jus" are "explained away" through predicting underlying /t/s, the (literally) extra-ordinary frequency of "jus"-tokens lets the model still posit it as an underlying form, although with a much dampened frequency (of the 1746 surface tokens, 1081 are analyzed as being realizations of an underlying "just").

The /t/-recovery performance drop when performing joint word segmentation isn't surprising as even the Bigram model doesn't deliver a very high-quality segmentation to begin with, leading to both sparsity (through missed word-boundaries) and potential noise (through misplaced word-boundaries). Using a more realistic generative process for the underlying forms, for example an adaptor grammar Johnson et al. (2007b), could address this shortcoming in future work without changing the overall architecture of the model although novel inference algorithms might be required.

6.6 CONCLUSION AND OUTLOOK

I presented a joint model for word segmentation and the learning of phonological rule probabilities from a corpus of transcribed speech. I find that a Bigram model reaches 77% /t/-recovery F-score when run with knowledge of true word-boundaries and when it can make use of both the preceeding and the following phonological context, and that unlike the Unigram model it is able to learn the probability of /t/-deletion in different contexts. When performing joint word segmentation on the Buckeye corpus, the Bigram model reaches around above 55% F-score for recovering deleted /t/s with a word segmentation F-score of around 72% which is 2% better than running a Bigram model that does not model /t/-deletion.

I also identified additional factors that might help handling /t/-deletion and similar phenomena, suggesting ways in which future work can extend the kind of model presented here. In particular, my findings highlight the importance of phonological contexts, word predictability and item- and frequency-specific probabilities in handling phonological variation.

Another obvious extension is towards models that handle more than just a single phenomenon and, ultimately, to models that also induce the phonological rules from the input.

Also, the two-level architecture I present is not limited to the mapping being defined in terms of rules rather than constraints in the spirit of Optimality Theory (Prince and Smolensky, 2004); I sketch how the model can be modified along those lines in the final chapter of the thesis as a suggestion for future work.

To conclude, I presented a model that provides a clean framework to test the usefulness of different factors for word segmentation and handling phonological variation in a controlled manner.

7 CONCLUSION

7.1 CONTRIBUTIONS

In this thesis, I have presented several studies on computational models of word segmentation. Here, I will briefly summarize the findings and contributions of the individual chapters.

BAYESIAN MODELING FRAMEWORK In chapter 1, I argue for a particular view of Bayesian modeling that emphasizes the idea of understanding what can be learned from particular kinds of inputs making particular assumptions, connecting this to poverty of stimulus arguments. In chapter 2, I introduce the mathematical background for non-parametric models of word segmentation that forms the basis for the experiments in the thesis, reviewing in close detail the Unigram and Bigram model of Goldwater et al. (2009). I also extend these models by adding hyper parameter inference and a base distribution that embodies a possible word constraint.

INCREMENTAL INFERENCE In chapter 3, I present a novel incremental particle filter algorithm for Goldwater et al. (2009)'s Unigram and Bigram model. I compare the algorithm to a batch Markov Chain Monte Carlo algorithm and identify several interesting differences between incremental and batch inference. I find that the need for a linguistically informed base distribution over possible words is much more pronounced in the incremental than in the batch setting; and that allowing the particle filter to perform 'rejuvenation' considerably improves its performance, allowing it to outperform the batch algorithm in terms of segmentation performance in particular settings.

I argue that the experimental findings ought to be interpreted not as providing evidence for particular mechanisms employed by human learners but as suggesting novel questions to ask about word segmentation models. In particular, the findings suggest that the kind of segmentations implied by a model can change as a function of the input size, a topic investigated in more detail in the next chapter.

SENSITIVITY TO INPUT SIZE Chapter 4 studies how input size and different modeling assumptions interact in word segmentation. I demonstrate that non-parametric word segmentation models exhibit a counter-intuitive overlearning property in which having access to more data worsens rather than improves segmentation performance and argue that this is due to linguistic dependencies that are not properly handled

by the model. This can be partly addressed by modeling additional aspects of language jointly with word segmentation – in particular, I show that using Johnson (2008b)'s idea of collocations prevents overlearning on a large corpus of child-directed speech.

I also find that for a collocation model to perform well, constraints on possible words are necessary. In particular, a collocation model that considers words to be arbitrary sequences of phonemes is prone to oversegment the input; as a result, it only attains 20% token f-score when performing inference over 1,000 utterances and peaks at 55% token f-score when performing inference over roughly 25,000 utterances. In contrast, if words are required to consist of syllables the same model consistently attains more than 80% token f-score, demonstrating both the effectiveness and necessity of constraining the form of possible words.

STRESS AND PHONOTACTIC CUES IN SEGMENTATION In chapter 5, I extend a previously proposed model of word segmentation Johnson and Goldwater (2009) in a way that allows it to take advantage of stress cues. In line with psycholinguistic evidence, I find that stress cues improve segmentation. In particular, I demonstrate that a segmentation model can correctly identify the preference for word-initial stress exhibited by the English language; and that, contrary to Yang (2004), no substantive constraint such as a Unique Stress Constraint needs to be built into a segmentation model but can be inferred from the input.

I also find that the ability to observe phonotactic cues to word boundaries interacts with the ability to use stress cues. The experimental results indicate phonotactic and stress cues are partly redundant for the segmentation models I study, highlighting the importance to be explicit about all assumptions built into a model so as to not under- or overestimate the relative importance of any individual cue.

MODELING PRONUNCIATION VARIATION Chapter 6 presents a way of adding pronunciation variation to the Unigram and Bigram model and discusses the specific phenomenon of word-final /t/-deletion. Experimental evaluation on a large corpus of naturalistic speech shows that the ability to model phonological context is essential to accurately model phonological variation; and that the ability to capture word dependencies is important to infer the rate at which a variable rule such as /t/-deletion applies.

The second important result is that the actual complexity of a phenomenon such as /t/-deletion can be vastly underestimated when evaluating models on 'artificial' data. Thus, whereas the model I present performs close to perfect on a corpus in which word-final /t/s were randomly deleted according to probabilities observed in naturalistic data, its performance drops when applied to a corpus in which the overall rate of deletions is identical but their occurrence is not 'random' but

was manually annotated. This highlights the importance of using naturalistic data in evaluating models.

7.2 DIRECTIONS FOR FUTURE WORK

The findings of this thesis suggest roughly two related directions for future work: moving towards more realistic models and moving towards more realistic and thorough evaluation. I will illustrate some possible future extensions.

7.2.1 *Towards more realistic models*

7.2.1.1 *Joint modeling*

The overlearning effect identified in chapter 4 indicates that even for a 'simple' language acquisition problem such as word segmentation, it is important to adequately capture the linguistic dependencies that exist between words. This points towards integrated models of language acquisition in which, for example, word segmentation is performed jointly with semantic and syntactic acquisition.

Recent examples of this kind of work by Synnave et al. (2014) and Johnson et al. (2014) explore semantic and syntactic extensions to the segmentation models examined in chapter 4.

ADDING SEMANTICS In Synnave et al. we demonstrate that in the same experimental scenario, a model that associates words with latent 'activities' such as eating or playing is less prone to undersegmentation and can successfully cluster words into semantically coherent groups. An obvious extension of this work is to incorporate the word-referent learning of Jones et al. (2010) which can be easily expressed as an adaptor grammar as well (Johnson et al., 2010). This would require, however, considerable manual annotation effort as salient objects for each utterance need to be identified from the video recordings of the transcripts.

ADDING SYNTAX Johnson et al. (2014) demonstrates how substantive knowledge about the syntactic distinction of function and content words aids segmentation. In many languages, function words like "the" and "a" tend to occur at the edges of content words and to be monosyllabic. These tendencies can be easily encoded in an adaptor grammar, and Johnson et al. show that a model that incorporates knowledge of these tendencies yields 92% token f-score on the Brent-Bernstein-Ratner corpus, outperforming previous state-of-the-art models by 4%.

In addition, their model correctly identifies content words such as "book", "want" or "doggy" as different from function words such as "a", "to" or "in". An obvious and, given the findings of chapter 4, promising

extension of their work is to study its sensitivity to input size. This will help understand whether basic knowledge of syntax can already address overlearning to some extent.

INTEGRATING SYNTAX AND SEMANTICS In addition to jointly modeling segmentation and semantics on the one hand and segmentation and syntax on the other hand, one can imagine jointly modeling all three. At the current stage, an integration of the models of Synnave et al. and Johnson et al. seems feasible and worthwhile, in particular with respect to the question whether modeling both (some aspects of) semantics and syntax is a more effective means of addressing overlearning than modeling just one of the two.

Of course, the concrete suggestions discussed here only scratch the surface of what acquisition of semantics and syntax really amount to. While I am looking forward to models that not only infer situational contexts and word referents but the actual meaning conveyed by an utterance jointly with word learning, I am doubtful that interesting models of this kind of complexity can be built at the moment. From a theoretical point of view because we lack detailed theories of how the kinds of meaning representations used by young infants actually look and, just as importantly from a modeling point of view, how the input required to build these kinds of representations could be encoded. From a practical point of view because, even if we had such theories, annotating even small amounts of data in such a fashion seems like a daunting task.

This is not to say that future work should limit itself to exploring 'mock' semantics and syntax; rather, I advocate a stepwise approach in which one gradually relaxes the simplifying assumptions inherent in current models in ways that promise to be feasible given current knowledge and technologies.

7.2.1.2 *Beyond exchangeability*

Another direction in which models can be made more realistic is by challenging some of the mathematical assumptions inherent in current proposals. In particular, recall that exchangeable models are completely insensitive to the order of observations. This is not true of human learning (see Langley, 1995, for a review). In addition, this is part of the reason for overlearning problem as order-insensitivity makes it possible to notice 'patterns' over observations that are arbitrarily far apart in the input.

The findings of the particle filter experiments show that one can derive order effects from exchangeable models using 'broken' inference. While this 'rational process' (Sanborn et al., 2006) approach to connect Bayesian models more directly to experimental results on human performance is popular, in chapter 3 I argue that we should instead consider

models which do not rely on a psychologically implausible assumption such as exchangeability.

Indeed, non-exchangeable models are a topic that has gained interest in statistical modeling. A recent review of a variety of prior distributions that can be used in the definition of such models is given by Foti and Williamson (2012). One particular proposal that lends itself to a straight-forward application to current models is the distance-dependent Chinese Restaurant Process (ddCRP) of Blei and Frazier (2011). As a concrete suggestion for future work, I sketch how it can be applied to the Unigram model of word segmentation.

UNIGRAM DDCRP MODEL The ddCRP is a generalization of the Chinese Restaurant Process which, as discussed in chapter 2, is at the heart of current models of word segmentation.[1]

At a high-level, the original Unigram model can be defined by sequentially sampling words using the predictive probability of a CRP (see equation 2.12 on page 31)

$$P(W_i = w \mid \boldsymbol{w}_{1:i-1}) \propto C(w, \boldsymbol{w}_{1:i-1}) + \alpha P_{\text{lex}}(w)$$

In this model, all previously generated words contribute to the predictive probability of the i^{th} words. In contrast, a ddCRP Unigram model is defined in terms of the following predictive distribution

$$P(W_i = w \mid \boldsymbol{w}_{1:i-1}) \propto \left(\sum_{j=1}^{i} \mathbb{1}[w_j = w] \, K(d_i, d_j) \right) + \alpha P_{\text{lex}}(w)$$

Here, d_i and d_j are timestamps of the respective words – for example, one can imagine each word to be timestamped with the utterance in which it occurs, with an actual time or simply with its position in the overall sequence of words. $K(x, y)$ is a kernel function which controls how strongly the j^{th} observation w_j influences the predictive probability of W_i. If K is constant, this just results in the original exchangeable Unigram model. However, if K is not constant this yields a non-exchangeable distribution over sequences of words where, for example, words that are "too far in the past" do not influence current segmentation choices. A simple choice for a kernel function is a fixed window size kernel of the form

$$K(d_i, d_j) = \begin{cases} 1 & \text{if } d_j \geqslant d_i - n \\ 0 & \text{else} \end{cases}$$

1 Strictly speaking, current models are defined in terms of the Dirichlet Process, and the Chinese Restaurant Process arises during inference – see section 2.3.6.1 on page 40 for discussion.

This has the effect that only the most recent n words[2] can directly influence how an utterance will be segmented. Intuitively, one can think of such a model as 'forgetting' observations that are too far in the past. Crucially, however, in so far as the segmentation choices for the utterances with the 'window' are affected by observations outside of the current window, these utterances still have an indirect effect on the segmentation; in particular, the segmentation choices of a ddCRP model with window size n on the final utterances in a large corpus do not have to be identical to those of an exchangeable CRP model which is only run on the final n utterances.

INFERENCE IN NON-EXCHANGEABLE MODELS A practical problem raised by non-exchangeable segmentation models is many inference algorithms rely on exchangeability for their efficiency. This is particularly clear for Gibbs sampling. For example, consider performing Gibbs sampling on a large corpus with thousands of utterances. In order to resample a boundary, the probabilities of making a small 'local change' need to be calculated which, in an exchangeable model, usually involves only the calculation of a small number of probabilities as one can think of the change as occurring at the end of the sequence of observations, leaving the probability of everything 'before it' unaffected (see Figure 2.8 on page 45).

In a non-exchangeable, however, the actual order of observations plays an important role. Therefore, the probabilities of everything following the boundary that is being resampled need to be recalculated – thus, if the second utterance in a $20,000$ utterance corpus is resampled, the change in probability across all of the remaining $19,998$ utterances needs to be considered. It may still be possible to derive efficient batch algorithms in this case, but at least obvious ideas such as using a Metropolis-Hastings sampler do not solve the problem introduced by non-exchangeability as calculating the acceptance probability raises the exact same problem.

However, an incremental particle filter like the one chapter 3 does not rely on exchangeability. Thus, the kind of sequential inference algorithm presented in this thesis is likely to prove useful for the study of non-exchangeable models for which batch algorithms may be impractical. It is worth pointing out, however, that in such a case, the use of rejuvenation also needs to be avoided or carefully restricted in so far as the correctness of rejuvenation also relies on exchangeability (see page 113).

2 Depending on what timestamps were chosen, this could also mean the most recent n utterances or the last hour / day / ....

7.2.1.3 *Handling phonological variation*

A third aspect of more realistic modeling assumptions concerns the handling of variation in the input. I only sketch some concrete proposals of how the work in chapter 6 may be extended.

LOCALLY NORMALIZED MAXIMUM ENTROPY MODELS One promising strategy is to use a locally normalized Maximum Entropy model (Berger et al., 1996) for the distribution P_R that maps underlying to surface forms.[3] From a practical point of view, this will allow for a less restricted use of contextual features in determining whether or not a phonological rule applies in any given context.

For example, while conditioning on whether or not the preceding segment is a consonant is already informative of whether or not /t/-deletion applied, knowing the place of articulation of that segment is likely to be even more informative: nasals such as "n" (in "want") tend to encourage /t/-deletion more than stops such as "p" (in "wept"). Handling a large number of possibly interacting features in a generative model like the one in chapter 6 is, however, impractical. For an excellent discussion of this point, see (Smith, 2011, p 83ff).

A conditional model, in contrast, makes it possible to use arbitrary features of the observations, sidestepping the kind of problems that arise in a generative model completely. Concretely, for the /t/-deletion in chapter 6 one can use a logistic regression model (Murphy, 2012, chapter 8) for the distribution P_R, yielding a simple form:

$$P_R(\text{TDROP} \mid u, r) = \frac{\exp(\overrightarrow{f(u,r)}\vec{w})}{1 + \exp(\overrightarrow{f(u,r)}\vec{w})}$$

$\overrightarrow{f(u,r)}$ is the *feature vector* which summarizes all relevant information for /t/-deletion provided by the context which is defined by the underlying form u and the following underlying form r. $\vec{w}$ is the weight-vector that indicates, for each feature, whether it makes /t/-deletion more probable (positive weight) or less probable (negative weight) to apply. The weight vector has to be modeled as a random variable and takes the role of the multiple ρ-probabilities in the original model. A natural prior distribution on it is a multi-dimensional Gaussian which corresponds to performing l_2 regularization (Murphy, 2012, p. 226).

Note that using this alternative distribution allows us to recover the original model by choosing six indicator features of the form "1 if the previous segment is a consonant and the following segment is a vowel, else 0", "1 if the previous segment is a vowel and the following segment is a vowel, else 0", ... More importantly, using this kind of distribution makes it very easy to incorporate many features that might be useful.

3 This idea of using locally normalized discriminative models as parts of larger generative model was popularized by Berg-Kirkpatrick et al. (2010).

For example, one can include a feature that indicates whether or not the preceding segment is a nasal, whether it is voiced, or whether the manner of articulation of the following segment is also a stop. All that needs to be changed is the function that maps underlying form and context to the feature vector.

Using such a distribution is also interesting from a theoretical perspective: Goldwater and Johnson (2003) shows that the features in a Maximum Entropy model can be related rather directly to the kinds of constraints posited in Optimality Theory (Prince and Smolensky, 2004).[4] This will make it possible to connect more directly with concrete proposals about phonological processes as constraints proposed by phonologists can be used to design useful features; and modeling results could be used to test whether or not particular constraints facilitate acquisition of particular phenomena.

INFERENCE FOR WEIGHTS Modifying the inference to accommodate this richer kind of distribution is straight-forward. In particular, conditional on any particular value of $\vec{w}$, the /t/-deletion probabilities for every underlying form are independent and the sampling equations in Figure 6.3 on page 195 can be used.

Similarly, it is straight-forward to re-estimate $\vec{w}$ given a current segmentation with indicated /t/-deletion decisions. Thus, note that this defines a set of labeled examples as every underlying form that ends in a /t/ is a relevant example and the currently inferred positions at which /t/-deletion occurred provide the labels. There are several possibilities of performing inference for the posterior distribution of $\vec{w}$ given a set of labeled examples (see Murphy, 2012, chapter 8.4). Arguably the easiest approach is to use Metropolis-Hastings sampling with a Gaussian proposal distribution that is centered around the MAP value for $\vec{w}$.

This requires identifying the MAP weight vector for a logistic regression which is, however, a standard problem in machine learning (see Murphy, 2012, chapter 8 for extensive review). In addition, I suspect that directly using the MAP or sampling values using a Metropolis-Hastings scheme will make little difference, just as sampling or optimizing the concentration parameters seems to make no difference in practice.

HANDLING MORE PHENOMENA Another important goal for future work is to design models that handle more variation phenomena. Here as well, using a logistic regression for P_{R} seems like a promising approach as it is easy to extend to a multi-class logistic regression (also known simply as a Maximum Entropy classifier). Formally, one adds a random variable O_i for every underlying form U_i which ranges over all

4 Strictly speaking, the connection is to Harmonic Grammar rather than Optimality Theory. For discussion, see Goldwater and Johnson (2003) and for a review of both Optimality Theory and Harmonic Grammar, see Smolensky and Legendre (2005).

possible phonological processes that can apply to U_i, including a special NoOp rule which indicates that no rule applied. The corresponding surface form S_i is then generated by 'applying' O_i to U_i. This mapping needs to be deterministic but otherwise, can be rather unconstrained. However, it needs to be kept in mind that during inference, the reverse mapping of S_i to all possible pairs $\langle U_i, O_i \rangle$ that yield S_i needs to be considered.

In this case, the sampling algorithm described in chapter 6 needs to be modified so as to not only make a ternary decision (no boundary, boundary with /t/-deletion, boundary without /t/-deletion) but to make a decision over all possible pairs of underlying forms and operations that are compatible with the observed surface form. For a very specific process such as /t/-deletion, the underlying form is uniquely determined. For more general processes such as deletion of any final segment, however, many possible underlying forms may need to be considered. To illustrate, a surface form such as [k æ] can be the result of applying DeleteFinal to any of /k æt/ ("cat"), /k æb/ ("cab"), /k æp/ ("cap"), . . . , . In theory, this is straight-forward but may raise practical problems if the set of underlying forms becomes very large.

LEARNING PHONOLOGICAL RULES This raises the issue which rules ought to be considered and, ultimately, whether the rules themselves could be learned rather than pre-specified.

Here, the alternative finite-state transducer architecture of Elsner et al. (2013) looks very attractive as a finite-state transducer can be viewed as a compact representation of a very large number of different phonological rules that rewrite underlying segments depending on their context. As Elsner et al. point out themselves, however, to be tractable the structure of their transducer needs to severely constrained. For example, they report that it is infeasible to condition re-write operations on the phonological context, allowing essentially only for context-free rewrite rules of the form /x/ → [y]. In addition, they exclude the ability to delete underlying segments (rules of the form /x/ → ϵ). The reason for this latter restriction is that unrestricted deletion yields an infinite number of possible underlying forms – a single observed segment could have arisen from an arbitrary number of underlying segments of which all but one were deleted. A consequence of this restriction is that their model currently induces /w a n/ ("wan") as the underlying form for /w a n t/ ("want") as it can only capture rules that insert segments rather than delete segments.

Arguably, this is merely a practical problem. A finite-state transducer architecture can in principle handle both context-sensitive rewriting and deletion. And there are likely to be ways to make inference in these models feasible without sacrificing the ability to recover plausible underlying forms. Yet, I find the practical problems that arise when one tries to model phonological variation in a highly unrestricted fashion

telling; and I think they suggest that strong constraints are needed for a feasible treatment of phenomena such as variation.

LEARNING OPTIMALITY THEORETIC CONSTRAINTS Another interesting recent proposal worth discussing as a possible extension is Doyle et al. (2014)'s non-parametric model that can induce Optimality Theory (Prince and Smolensky, 2004)-like constraints in an unsupervised fashion. This is particularly interesting because, as mentioned above, the features in a Maximum Entropy model can be viewed as corresponding to Optimality Theory constraints. However, in its current form Doyle et al.'s model requires knowledge of the underlying forms.

Theoretically, it is easy to extend their model to an unsupervised case by using an iterative strategy such as Expectation Maximization or Gibbs sampling in which underlying forms are proposed conditioning on a current set of constraints, and constraints are induced conditioning on the currently estimated underlying forms.

In practice, however, this again raises the question how the range of underlying forms can be suitably constrained – the nature of the induced constraints depends heavily on the form of the underlying forms, but what counts as a plausible underlying form depends heavily on what possible constraints are. Thus, even if it may be possible to induce the relevant constraints given knowledge of the underlying forms rather than assuming knowledge of the relevant constraints to be innate, some strategy of restricting the underlying forms in a plausible way during learning is required in a language acquisition scenario where only surface forms are observed.[5]

7.2.2 *More realistic evaluations*

7.2.2.1 *Cross-linguistic evaluation*

An important question is whether model performance on English translates to other languages. As it is plausible to assume that infants' ability to segment their speech stream is universal, failure to generalize to other languages constitutes a severe shortcoming of any proposed model.

This question has been discussed widely in the literature (Fleck, 2008; Fourtassi et al., 2013; Daland and Zuraw, 2013), and the general finding is that there is a huge gap between performance on English and other languages. The stress model investigated in this thesis raises two concrete questions in this respect.

First, does the particular way in which the current model takes advantage of stress generalize to languages which exhibit more complex stress-patterns? In particular, English's tendency of short words makes it unnecessary to consider relative stress-patterns such as 'ante penulti-

5 Note that in word segmentation, the problem is even harder in that even the surface forms – i.e. the actual words – are latent.

mate' (stressed on the third syllable counting from the end of the word) on child-directed speech as this pattern only makes a real difference over 'stressed on the first syllable' for words that are at least 4 syllables long. For languages with longer words, however, a more complex model of stress may be required to yield noticeable segmentation gains. Unfortunately, at the current stage there is lack of experimental evidence with respect to the role stress plays in infant segmentation across languages with different stress patterns (Höhle et al., 2009), raising the question of how results from computational models on other languages ought to be interpreted.

A second question concerns whether making use of stress cues provides a way of improving the performance of current models on languages other than English. For example, Fourtassi et al. (2013) argue that the reason Japanese is hard is a very high segmentation ambiguity – there is an extremely large number of possible segmentations of any given utterance even if one only considers true Japanese words. Arguably, to solve this problem a model needs to rely on additional cues which reduce this ambiguity which is inherent in the data if it is only considered as a sequence of phonemes. Stress or, in the case of Japanese, pitch accent seems like a natural candidate for this and it would be interesting to understand both if and how a segmentation model could use these cues to aid segmentation.

Of course, any cross-linguistic explorations either presupposes the existence or needs to involve the creation of a relevant data set. While CHILDES (MacWhinney, 2000) makes available transcripts of child-directed speech across a wide variety of languages, creating data sets that can be used to evaluate models and the usefulness of cues such as stress is a challenge in and of itself.

7.2.2.2 *More detailed evaluations*

Whereas it is generally accepted that there is a need for wider cross-linguistic evaluation, a topic that has not received a lot (if any) attention so far is the methodological question how segmentation models ought to be evaluated. At the moment, models are mainly compared by their segmentation accuracy which is usually defined as the token f-score.

This raises the question to what gold standard proposed segmentations of a given corpus should be compared. The orthographic segmentation is currently considered as the only correct answer. This makes it easy to evaluate models on large corpora as creating the gold standard is trivial. It is, however, questionable that the actual segmentations of young infants coincide perfectly with those prescribed somewhat arbitrarily by a writing system.

This may be more obvious for languages which lack a clear notion of orthographic word. For example, Chew (1964) provides ample evidence that native speakers of Japanese are often unable to put meaningful word boundaries in their romanized transcriptions of Japanese

sentences. In addition to indicating that even competent speakers may not have clear intuitions as to where exactly word boundaries fall in every context, this also suggests a possible explanation for the surprisingly high segmentation ambiguity Fourtassi and Dupoux (2014) found: the orthographic segmentation of their corpus may simply not employ a consistent notion of word versus morpheme, leading to an overestimation of true words in this corpus because sequences of grammatical morphemes may have been transcribed as individual words in some contexts and sequences of multiple 'words' in others.

In conclusion, it is an empirical question into what word-like units humans segment their input, even for languages such as English. Thus, consider the question of whether treating the sequence /ðədog/ ("the-dog") as a single rather than two words really ought to be considered as a mistake. On the one hand, we have every reason to believe that speakers of English know that "the" is a word and that "dog" is another word. Yet, there is evidence that young infants do treat cases like this as instances of single words (Brown, 1973), and we are ultimately not interested in models for the sake of recovering the orthographic standard but to understand how infants do what they do. In so far as this might deviate from the orthographic standard, it is not clear whether a model that attains a higher token f-score with respect to it is really preferable over a model that attains a lower f-score.

Relatedly, if models posit additional units such as collocations an important question is whether these units are also posited by human learners – concretely, are the multi-word sequences posited by a collocation model similar to the kinds of collocations infants are known to acquire?

Consequently, we should be thinking about psychologically motivated evaluations that go beyond looking at a simple summary statistic such as segmentation accuracy with respect to some gold segmentation. This is not to say that our current way of evaluating models is completely meaningless – they are the best objective standard we currently have – but I think there is a clear need to work more closely with experimental psycholinguists in evaluating and comparing different models.

To this end, the kind of qualitative analysis done in Table 4.3 on page 156 where particular patterns are directly examined could be used to both derive predictions for possible language learning experiments and to evaluate models against results of such experiments suggests one possible analysis that could be used to provide more detailed analyses of models in the future that can then be compared to what we know about human performance on particular patterns.

Similarly, taking a closer look at the performance of particular types of words as in Figure 4.7 on page 155 or directly evaluating linguistic knowledge inferred by a model as in the stress pattern evaluation in Figure 5.5 on page 182 are strategies that should allow for a more

direct connection to psycholinguistic findings than merely comparing segmentation accuracies on large quantities of data.

7.2.2.3 *Posterior analysis*

To conclude, I want to illustrate a strength of the *Bayesian* approach that has so far not been used to its full extent and, in particular with respect to detailed evaluations, might prove very fruitful: its ability to directly quantify uncertainty. Thus, it is common in current work (including this thesis) to collapse posterior distributions down to a single number or, in the case of word segmentation, segmentation. At no point, posterior probabilities of particular choices are considered directly. For example, even though maximum marginal segmentations take, in a clear sense, into account the information present in the posterior distribution, at the end they produce single segmentation for each utterance.

The Bayesian framework allows one to also ask directly how certain one should be about segmenting an utterance in a particular way. For example, it may well be the case that given the input the model is next to 100% certain that one utterance ought to be segmented like this but, with respect to some other utterance, only 9% certain of the single most probable segmentation. This is valuable information in that low certainty can be viewed as the model's way of explicitly saying that it does not really know what to do with an observation – there really is not sufficient information to segment it.

There is evidence that human learners are sensitive to certainty in this sense as they tend to actively ignore observations which are so complex that, given their current knowledge, they cannot identify any particular analysis with reasonable certainty. This "Goldilocks-effect" (Kidd et al., 2012) can be derived directly from Bayesian models, and I will briefly sketch two ways in which this can be done in future work in the context of word segmentation.

POSTERIOR ENTROPY TO SELECT UTTERANCES Table 2.3 on page 55 illustrated the idea of enumerating an entire marginal posterior distribution for a particular segmentation. Manual inspection of these distributions can prove interesting in and of itself. The information conveyed by such a distribution can, however, also be used to decide whether or not a segmentation prediction is made at all. In particular, if the posterior distribution reflects that there are many competing segmentations of comparable probability it intuitively makes sense to abstain from committing to any individual segmentation.

Marginal posterior entropy is an easy-to-calculate metric that reflects just this. For an utterance u, one can define its posterior entropy as $H(u) = -\sum_s \hat{P}(s) \log \hat{P}(s)$ where the sum is over all sample segmentations for this utterance and $\hat{P}(s)$ is the Monte Carlo approximation to the posterior probability of this particular segmentation. To illustrate,

select	# utterances	token f-score	pred. types	% correct
all	1,093	.58	420	.47
$H(u) \leqslant 2.0$	818	.65	250	.56
$H(u) \leqslant 1.0$	505	.75	155	.59
$H(u) \leqslant 0.1$	216	.84	58	.74

Table 7.1: Evaluating the Bigram model with hyper parameter sampling and with unconstrained base distribution on the Alice corpus, taking into account posterior entropy. Note how token f-score on the utterances about which the model is certain is considerably higher than on the entire corpus.

the entropy of the posterior in Table 2.3 is approximately 1.12 which, at a high level, can be interpreted as there being two segmentations that account for most of the probability mass and with respect to which the learner is somewhat undecided.

To illustrate how this idea can be used in evaluating models, recall the bad performance of the Bigram model on the Alice corpus when hyper parameters are inferred (see Table 2.5 on page 57). Table 7.1 shows that if evaluation is limited to those utterances about which the model is very certain, both the token f-score (on a smaller sample) and the precision of the inferred lexicon are considerably higher, suggesting that the bad performance is due to there being a lot of uncertainty about most of the utterances. This is, actually, in line with the discussion in chapter 4 – more complex models, such as the Bigram model, require either strong constraints or large amounts of data to reliably segment their input.

In addition to offering a finer grained way of evaluating models, this strategy might also be ultimately incorporated directly into learning algorithms. For example, one can imagine a learner that processes its input sequentially in mini batches. After each mini batch, it can use posterior entropy to determine the utterances which are actually used to update its beliefs about words; and ignore segmentations about which it is very uncertain, mimicking to some extent the selective behavior exhibited by humans (Kidd et al., 2012).

BAYESIAN WORD SPOTTING The posterior distribution also allows one to identify parts within individual utterances about which the model is certain to different degrees. This makes it possible to perform *word spotting*, i.e. identifying only those parts in an utterance about which the model is certain.

One can view each sample segmentation as a binary vector that indicates presence (1) or absence (0) of a word-boundary. By averaging over all binary vectors derived from our samples, we get a marginal estimate of how strongly the learner beliefs there to be a boundary at any given

certainty	segmentation
1.0	xxx
0.9	xxx si xxx
0.8	xxx si ðəbʊk
0.5	yu wanttu si ðəbʊk

Table 7.2: Partial segmentations of "youwanttoseethebook" at different certainty levels. Note that at certainty 0.5, the segmentation is identical to the maximum marginal posterior segmentation of the entire utterance, but that for higher certainty levels entire parts of the utterance may remain unanalyzed.

position. As an example, the average boundary vector for the posterior in Table 2.3 is

$$\nu = y_{\langle 0} u_{0.51} w_0 a_0 n_0 t_0 t_0 u_{0.99} s_0 i_{1.0} ð_0 ə_{0.13} b_0 ʊ_{0\rangle} k$$

with the marginal boundary probabilities written as subscripts in between adjacent phonemes. While this vector does not define any particular segmentation, one can use it to identify parts of the utterance which the model is certain enough to segment and parts which it is not certain about.

To this end, one can define the notion of a 'γ-certain word': a γ-certain word is a word for which the model is certain to degree γ that it is present in an utterance. Crucially, all segmentations with non-zero posterior probability, that is, not only the single most probable segmentation, are taken into account for this.

Referring to the elements of ν using subscripts such that $\nu_0 = 0$ and $\nu_1 = 0.51$, a γ-certain' word is any sequence of phonemes between two boundary indices l and r, $l < r$, such that $\nu_l > \gamma$ and $\nu_r > \gamma$, and for all j, $l < j < r$, $\nu_j < 1 - \gamma$.

Table 7.2 illustrates this idea. For different values of γ, it gives the γ-certain words in the example utterance and represents the ignored parts using xxx. We see that in this case, the learner is really only certain about the presence of the word /si/ ("see"). In particular, it is rather uncertain how the initial sequence "you wantto" ought to be segmented, and only at certainty 0.5 it identifies the two words "you" and "wanttu".

This technique could be used to see whether, for example, particular words such as "mammy" or "daddy" are learned with much higher certainty than other words by different models. More generally, the idea of word spotting might make it possible to connect segmentation models with experimental results in which infants' knowledge of words at different stages is evaluated.

7.3 CONCLUSION

There are many interesting questions that remain to be addressed in understanding how human infants perform word segmentation and ultimately acquire language. I hope that this thesis serves its purpose by providing some answers to certain specific questions, raising novel questions that ought to be addressed in future work, and, in this final chapter, providing concrete suggestions as to how one may go about answering some of these questions.

BIBLIOGRAPHY

Alishahi, A. (2011). *Computational Modeling of Human Language Acquisition*. Morgan & Claypool.

Beal, M., Ghahramani, Z., and Rasmussen, C. (2002). The infinite Hidden Markov Model. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems*, volume 14, pages 577–584. The MIT Press.

Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J., and Klein, D. (2010). Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California. Association for Computational Linguistics.

Berger, A. L., Della Pietra, V. J., and Della Pietra, S. A. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Bernardo, J. M. (1996). The concept of exchangeability and its applications. *Far East Journal of Mathematical Sciences*, 4:111–122.

Bernstein-Ratner, N. (1987). The phonology of parent-child speech. In Nelson, K. and van Kleeck, A., editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blanchard, D., Heinz, J., and Golinkoff, R. (2010). Modeling the contribution of phonotactic cues to the problem of word segmentation. *Journal of Child Language*, 37(3):487–511.

Blei, D. M. and Frazier, P. I. (2011). Distance dependent Chinese restaurant processes. *Journal of Machine Learning Research*, 12:2461–2488.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Börschinger, B. (2012). Review of "Linguistic Structure Prediction" by Noah Smith. *Linguist List*.

Börschinger, B., Demuth, K., and Johnson, M. (2012). Studying the effect of input size for Bayesian word segmentation on the Providence corpus. In *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*, pages 325–340, Mumbai, India. Coling 2012 Organizing Committee.

Börschinger, B. and Johnson, M. (2011). A particle filter algorithm for Bayesian word segmentation. In *Proceedings of the 2011 Australasian Language Technology Workshop*, pages 10–18, Canberra, Australia. Australasian Language Technology Association.

Börschinger, B. and Johnson, M. (2012). Using rejuvenation to improve particle filtering for Bayesian word segmentation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 85–89, Jeju Island, Korea. Association for Computational Linguistics.

Börschinger, B. and Johnson, M. (2014). Exploring the role of stress in Bayesian word segmentation using Adaptor Grammars. *Transactions of the Association of Computational Linguistics*, 2:93–104.

Börschinger, B., Johnson, M., and Demuth, K. (2013). A joint model of word segmentation and phonological variation for English word-final /t/-deletion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1508–1516, Sofia, Bulgaria. Association for Computational Linguistics.

Börschinger, B., Jones, B. K., and Johnson, M. (2011). Reducing grounded learning tasks to grammatical inference. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1416–1425, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Boruta, L. and Jastrzebska, J. (2012). A phonemic corpus of polish child-directed speech. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 1017–1020, Istanbul, Turkey. European Language Resources Association (ELRA).

Brent, M. (1999). An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Brent, M. and Cartwright, T. (1996). Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

Brown, R. (1973). *A First Language: the Early Stages*. Harvard University Press, Cambridge, MA.

Buntine, W. and Hutter, M. (2010). A bayesian view of the poisson-dirichlet process. *arXiv preprint arXiv:1007.0296*.

Canini, K. R., Shi, L., and Griffiths, T. L. (2009). Online inference of topics with latent Dirichlet allocation. In van Dyk, D. and Welling, M., editors, *Proceeings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 65–72.

Carnegie Mellon University (2008). The cmu pronpronoun dictionary, v.0.7a.

Chater, N. and Oaksford, M., editors (2008). *The probabilistic mind: prospects for Bayesian cognitive science*. Oxford University Press.

Chew, John J., J. (1964). On word boundaries in japanese. *The Journal-Newsletter of the Association of Teachers of Japanese*, 2(3):pp. 6–12.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. The MIT Press, Cambridge, Massachusetts.

Chomsky, N. (1968). Quine's empirical assumptions. *Synthese*, 19(1/2):pp. 53–68.

Chomsky, N. (1980). *Rules and Representations*. Blackwell Publishers.

Chomsky, N. (1981). *Lectures on Government and Binding*. Foris, Dordrecht.

Chomsky, N. (1986). *Knowledge of Language: Its Nature, Origin and Use*. Praeger, New York.

Chomsky, N. (1995). *The Minimalist Program*. The MIT Press, Cambridge, Massachusetts.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Haper & Row, New York.

Christiansen, M. and Curtin, S. (1999). The power of statistical learning: No need for algebraic rules. In *Proceedings of the 21st Annual Conference of the Cognitive Science Society*, Mahwah, NJ.

Christiansen, M. H., Allen, J., and Seidenberg, M. S. (1998). Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes*, 13(2-3):221–268.

Christiansen, M. H. and Chater, N. (2009). The myth of language universals and the myth of universal grammar. *Behavioral and Brain Sciences*, 32(05):452–453.

Clark, E. V. (2009). *First Language Acquisition*. Cambridge University Press.

Coetzee, A. W. (2004). *What it Means to be a Loser: Non-Optimal Candidates in Optimality Theory*. PhD thesis, University of Massachusetts , Amherst.

Coetzee, A. W. and Kawahara, S. (2013). Frequency biases in phonological variation. *Natural Language and Linguisic Theory*, 31:47–89.

Cohen, S. (2011). *Computational Learning of Probabilistic Grammars in the Unsupervised Setting*. PhD thesis, Carnegie Mellon University.

Cohen, S. B., Blei, D. M., and Smith, N. A. (2010). Variational inference for adaptor grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572, Los Angeles, California. Association for Computational Linguistics.

Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13.

Curtin, S., Mintz, T. H., and Christiansen, M. H. (2005). Stress changes the representational landscape: Evidence from word segmentation. *Cognition*, 96(3):233–262.

Cutler, A. (1991). Exploiting prosodic probabilities in speech segmentation. In *Cognitive models of speech processing*, pages 104–121. MIT Press.

Cutler, A. (2005). Lexical stress. In Pisoni, D. B. and Remez, R. E., editors, *The Handbook of Speech Perception*, pages 264–289. Blackwell Publishing.

Cutler, A. and Carter, D. M. (1987). The predominance of strong initial syllables in the English vocabulary. *Computer Speech and Language*, 2(3):133–142.

Cutler, A., Mehler, J., Norris, D., and Segui, J. (1986). The syllable's differing role in the segmentation of French and English. *Journal of Memory and Language*, 25(4):385 – 400.

Daland, R. (2009). *Word segmentation, word recognition, and word learning: a computational model of first language acquisition*. PhD thesis, Northwestern University.

Daland, R. and Pierrehumbert, J. B. (2011). Learning diphone-based segmentation. *Cognitive Science*, 35(1):119–155.

Daland, R. and Zuraw, K. (2013). Does korean defeat phonotactic word segmentation? In *ACL (2)*, pages 873–877.

de Finetti, B. (1990). *Theory of Probability*, volume Vol. 1-2. John Wiley and Sons., reprint of the 1975 translation edition.

Demuth, K., Culbertson, J., and Alter, J. (2006). Word-minimality, epenthesis, and coda licensing in the acquisition of English. *Language and Speech*, 49:137–174.

Dilley, L., Millett, A., McAuley, J. D., and Bergeson, T. R. (to appear). Phonetic variation in consonants in infant-directed and adult-directed speech: The case of regressive place assimilation in word-final alveolar stops. *Journal of Child Language*.

Douc, R. and Cappé, O. (2005). Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69. IEEE.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208.

Doyle, G., Bicknell, K., and Levy, R. (2014). Nonparametric learning of phonological constraints in optimality theory. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1103, Baltimore, Maryland. Association for Computational Linguistics.

Doyle, G. and Levy, R. (2013). Combining multiple information types in Bayesian word segmentation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 117–126, Atlanta, Georgia. Association for Computational Linguistics.

Dunbar, E. (2013). *Statistical Knowledge and Learning in Phonology*. PhD thesis, University of Maryland, College Park.

Earman, J. (1992). *Bayes or Bust? A critical examination of Bayesian Confirmation Theory*. The MIT Press.

Elsner, M., Goldwater, S., and Eisenstein, J. (2012). Bootstrapping a unified model of lexical and phonetic acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 184–193, Jeju Island, Korea. Association for Computational Linguistics.

Elsner, M., Goldwater, S., Feldman, N., and Wood, F. (2013). A joint learning model of word segmentation, lexical acquisition, and phonetic variability. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 42–54, Seattle, Washington, USA. Association for Computational Linguistics.

Ferguson, T. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1:209–230.

Fleck, M. M. (2008). Lexicalized phonotactic word segmentation. In *Proceedings of ACL-08: HLT*, pages 130–138, Columbus, Ohio. Association for Computational Linguistics.

Foti, N. J. and Williamson, S. (2012). A survey of non-exchangeable priors for bayesian nonparametric models. *arXiv preprint arXiv:1211.4798*.

Fourtassi, A., Börschinger, B., Johnson, M., and Dupoux, E. (2013). Why is English so easy to segment? In *Proceedings of the 4th Workshop on Cognitive Modeling and Computational Linguistics*.

Fourtassi, A. and Dupoux, E. (2014). A rudimentary lexicon and semantics help bootstrap phoneme acquisition. In *Proceedings of the 18th Conference on Computational Natural Language Learning (CoNLL)*.

Frank, M. C., Goodman, N., and Tenenbaum, J. (2009). Using speakers' referential intentions to model early cross-situational word learning. *Psychological Science*, 20:579–585.

Frank, S. (2014). Learning the hyperparameters to learn morphology. In *Proceedings of the 5th Workshop on Cognitive Aspects of Computational Language Learning (CogACLL)*, pages 14–18, Gothenburg, Sweden. Association for Computational Linguistics.

Fromkin, V., editor (2001). *Linguistics: An Introduction to Linguistic Theory*. Blackwell, Oxford, UK.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014). *Bayesian Data Analysis*. Chapman and Hall/CRC, Boca Raton, FL, 3 edition.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Gervain, J. and Erra, R. G. (2012). The statistical signature of morphosyntax: A study of hungarian and italian infant-directed speech. *Cognition*, (0):–.

Gibson, E. and Wexler, K. (1994). Triggers. *Linguistic Inquiry*, 25(3):407–454.

Gillies, D. (2000). *Philosophical theories of Probability*. Routledge.

Goldwater, S. (2007). *Nonparametric Bayesian Models of Lexical Acquisition*. PhD thesis, Brown University.

Goldwater, S., Griffiths, T., and Johnson, M. (2006). Interpolating between types and tokens by estimating power-law generators. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*, pages 459–466, Cambridge, MA. MIT Press.

Goldwater, S., Griffiths, T. L., and Johnson, M. (2009). A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Goldwater, S., Griffiths, T. L., and Johnson, M. (2011). Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research*, 12:2335–2382.

Goldwater, S. and Johnson, M. (2003). Learning OT constraint rankings using a Maximum Entropy model. In Spenader, J., Eriksson, A., and Dahl, O., editors, *Proceedings of the Stockholm Workshop on Variation within Optimality Theory*, pages 111–120, Stockholm. Stockholm University.

Good, I. J. (1971). Twenty-seven principles of rationality. In *Foundations of Statistical Inference*. Holt, Rinehart and Winston.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET.

Griffiths, T. L., Kemp, C., and Tenenbaum, J. B. (2008). *The Cambridge Handbook of Computational Psychology*, chapter Bayesian Models of Cognition. In Sun (2008a).

Guasti, M. T. (2004). *Language acquisition: The growth of grammar*. MIT Press.

Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., and Nordlund, P.-J. (2002). Particle filters for positioning, navigation, and tracking. *Signal Processing, IEEE Transactions on*, 50(2):425–437.

Guy, G. R. (1991). Contextual conditioning in variable lexical phonology. *Language Variation and Change*, 3(2):223–39.

Hacking, I. (2009). *An Introduction to Probability and Inductive Logic*. Cambridge University Press.

Hájek, A. (2012). Interpretations of probability. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Winter 2012 edition.

Harley, T. A. (2008). *The psychology of language: from data to theory*. Psychology Press.

Hayes, B. (2009). *Introductory Phonology*. Wiley-Blackwell.

Höhle, B., Bijeljac-Babic, R., Herold, B., Weissenborn, J., and Nazzi, T. (2009). Language specific prosodic preferences during the first half year of life: Evidence from german and french infants. *Infant Behavior and Development*, 32(3):262–274.

Jansen, A., Dupoux, E., Goldwater, S., Johnson, M., Khudanpur, S., Church, K., Feldman, N., Hermansky, H., Metze, F., Rose, R., et al. (2013). A summary of the 2012 JHU workshop on zero resource speech technologies and models of early language acquisition. In *Proceedings of ICASSP*, volume 2013, pages 8111–8115.

Jaynes, E. (2003). *Probability Theory*. Cambridge University Press.

Johnson, M. (2007). Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with Unfold-Fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175, Prague, Czech Republic. Association for Computational Linguistics.

Johnson, M. (2008a). Unsupervised word segmentation for Sesotho using Adaptor Grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27, Columbus, Ohio. Association for Computational Linguistics.

Johnson, M. (2008b). Using Adaptor Grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, pages 398–406, Columbus, Ohio. Association for Computational Linguistics.

Johnson, M., Christophe, A., Dupoux, E., and Demuth, K. (2014). Modelling function words improves unsupervised word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 282–292, Baltimore, Maryland. Association for Computational Linguistics.

Johnson, M. and Demuth, K. (2010). Unsupervised phonemic Chinese word segmentation using Adaptor Grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 528–536, Beijing, China. Coling 2010 Organizing Committee.

Johnson, M., Demuth, K., Frank, M., and Jones, B. (2010). Synergies in learning words and their referents. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 1018–1026.

Johnson, M. and Goldwater, S. (2009). Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado. Association for Computational Linguistics.

Johnson, M., Griffiths, T., and Goldwater, S. (2007a). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York. Association for Computational Linguistics.

Johnson, M., Griffiths, T. L., and Goldwater, S. (2007b). Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.

Jones, B. K., Johnson, M., and Frank, M. C. (2010). Learning words and their meanings from unsegmented child-directed speech. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 501–509, Los Angeles, California. Association for Computational Linguistics.

Joyce, J. (2008). Bayes' theorem. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition.

Jusczyk, P. (1997). *The discovery of spoken language*. MIT Press, Cambridge, MA.

Jusczyk, P. W., Cutler, A., and Redanz, N. J. (1993). Infants' preference for the predominant stress patterns of English words. *Child Development*, 64(3):675–687.

Jusczyk, P. W., Hohne, E. A., and Bauman, A. (1999a). Infants' sensitivity to allophonic cues for word segmentation. *Perception and Psychophysics*, 61:1465–1476.

Jusczyk, P. W., Houston, D. M., and Newsome, M. (1999b). The beginnings of word segmentation in English-learning infants. *Cognitive Psychology*, 39(3-4):159–207.

Kidd, C., Piantadosi, S. T., and Aslin, R. N. (2012). The goldilocks effect: Human infants allocate attention to visual sequences that are neither too simple nor too complex. *PLoS ONE*, 7(5):e36399.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models*. The MIT Press.

Korman, M. (1984). Adaptive aspects of maternal vocalizations in differing contexts at ten weeks. *First Language*, 5:44–45.

Kripke, S. (1982). *Wittgenstein on Rules and Private Language*. Harvard University Press.

Kurumada, C., Meylan, S. C., and Frank, M. C. (2013). Zipfian frequency distributions facilitate word segmentation in context. *Cognition*, 127(3):439–453.

Ladefoged, P. (2012). *Vowels and consonants*. John Wiley & Sons.

Langley, P. (1995). Order effects in incremental learning. In Reimann, P. and Spada, H., editors, *Learning in humans and machines: Towards an interdisciplinary learning science*. Elsevier.

Liang, P., Jordan, M., and Klein, D. (2009). Probabilistic grammars and hierarchical Dirichlet processes. In *The Oxford Handbook of Applied Bayesian Analysis*. Oxford University Press.

Liang, P., Petrov, S., Jordan, M., and Klein, D. (2007). The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697.

Lignos, C. (2011). Modeling infant word segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 29–38. Association for Computational Linguistics.

Lignos, C. (2012). Infant word segmentation: An incremental, integrated model. In *Proceedings of the West Coast Conference on Formal Linguistics 30*.

Lignos, C. and Yang, C. (2010). Recession segmentation: simpler online word segmentation using limited resources. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 88–97. Association for Computational Linguistics.

Liu, J. S. (1994). The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966.

Lunn, D., Jackson, C., Best, N., Thomas, A., and Spiegelhalter, D. (2012). *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Chapman and Hall/CRC.

MacKay, D. J. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

MacWhinney, B. (2000). The CHILDES project: Tools for analyzing talk: Volume I: Transcription format and programs, volume II: The database. *Computational Linguistics*, 26(4):657–657.

Marr, D. (1982). *Vision*. W.H. Freeman and Company, New York.

Marthi, B., Pasula, H., Russell, S., and Peres, Y. (2002). Decayed mcmc iltering. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 319–326. Morgan Kaufmann Publishers Inc.

Mattys, S. L. (2000). The perception of primary and secondary stress in English. *Perception and Psychophysics*, 62(2):253–265.

Mattys, S. L. and Jusczyk, P. W. (2000). Phonotactic cues for segmentation of fluent speech by infants. *Cognition*, 78(2):91–121.

Mattys, S. L., Jusczyk, P. W., Luce, P. A., and Morgan, J. L. (1999). Phonotactic and prosodic effects on word segmentation in infants. *Cognitive psychology*, 38(4):465–494.

May, C., Clemmer, A., and Van Durme, B. (2014). Particle filter rejuvenation and latent dirichlet allocation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 446–451, Baltimore, Maryland. Association for Computational Linguistics.

Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, 44(247):335–341.

Meylan, S., Kurumada, C., Börschinger, B., Johnson, M., and Frank, M. C. (2012). Modeling online word segmentation performance in structured artificial languages. In *Proceedings of the 34th Annual Meeting of the Cognitive Science Society*.

Mochihashi, D., Yamada, T., and Ueda, N. (2009). Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore. Association for Computational Linguistics.

Murphy, K. and Russell, S. (2001). Rao-blackwellised particle filtering for dynamic bayesian networks. In Doucet, A., de Freitas, N., and Gordon, N., editors, *Sequential Monte Carlo in Practice*, pages 499–516. Springer.

Murphy, K. P. (2012). *Machine Learning: A Probabalistic Perspective*. The MIT Press.

Naradowsky, J. and Goldwater, S. (2009). Improving morphology induction by learning spelling rules. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1531–1536, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Neal, R. M. (2003). Slice sampling. *Annals of Statistics*, 31:705–767.

Newport, E. L. (1990). Maturational constraints on language learning. *Cognitive Science*, 14:11–28.

Niyogi, P. (2006). *The Computational Nature of Language Learning and Evolution*. MIT Press.

Norris, D., Mcqueen, J. M., Cutler, A., and Butterfield, S. (1997). The possible-word constraint in the segmentation of continuous speech. *Cognitive Psychology*, 34(3):191 – 243.

Pate, J., Johnson, M., and Demuth, K. (unpublished). Vowel quality provides the same information as stress for word segmentation.

Pearl, L. and Goldwater, S. (in press). Statistical learning, inductive bias, and bayesian inference in language acquisition. In Lidz, J., editor, *Oxford Handbook of Developmental Linguistics*. Oxford University Press.

Pearl, L., Goldwater, S., and Steyvers, M. (2010). How ideal are we? Incorporating human limitations into Bayesian models of word segmentation. In *Proceedings of the 34th annual Boston University Conference on Child Language Development*, pages 315–326, Somerville, MA. Cascadilla Press.

Perfors, A., Tenenbaum, J. B., and Regier, T. (2011). The learnability of abstract syntactic principles. *Cognition*, 118(3):306–338.

Peters, A. (1983). *The Units of Language Acquisition*. Cambridge University Press.

Phillips, L. and Pearl, L. (2012). "Less is more" in Bayesian word segmentation: when cognitively plausible learners outperform the ideal. In Miyake, N., Peebles, D., and Cooper, R. P., editors, *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, Austin, Texas. Cognitive Science Society.

Phillips, L. and Pearl, L. (in press). The utility of cognitive plausibility in language acquisition modeling: Evidence from word segmentation. *Cognitive Science*.

Pinker, S. and Jackendoff, R. (2009). The reality of a universal language faculty. *Behavioral and Brain Sciences*, 32(05):465–466.

Pitt, M. A., Dilley, L., Johnson, K., Kiesling, S., Raymond, W., Hume, E., and Fosler-Lussier, E. (2007). Buckeye corpus of conversational speech.

Popper, K. R. (1959). *The logic of scientific discovery*.

Prince, A. and Smolensky, P. (2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell.

Quine, W. (1951). Two dogmas of empiricism. *The Philosophical Review*, 60:20–43.

Ramsey, F. (1931). Truth and probability. In *The Foundations of Mathematics and Other Logical Essays*, chapter VII, pages 156–198. Routledge and Kegan Paul.

Roy, B. C., Frank, M. C., and Roy, D. (2012). Relating activity contexts to early word learning in dense longitudinal data. In *Proceedings of the 34th Annual Cognitive Science Conference*.

Saffran, J., Aslin, R., and Newport, E. (1996). Statistical learning by 8-month-old infants. *Science*, 274:1926–1928.

Sakas, W. G. and Fodor, J. D. (2012). Disambiguating syntactic triggers. *Language Acquisition*, 19(2):83–143.

Sanborn, A. N., Griffiths, T. L., and Navarro, D. J. (2006). A more rational model of categorization. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*.

Sanborn, A. N., Griffiths, T. L., and Navarro, D. J. (2010). Rational approximation to rational models: Alternative algorithms for category learning. *Psychological Review*, 117(4):1144–1167.

Schrimpf, N. M. and Jarosz, G. (2014). Comparing models of phonotactics for word segmentation. *ACL 2014*, pages 19–28.

Selkirk, E. O. (1984). *Phonology and Syntax: The Relation Between Sound and Structure*. MIT Press.

Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650.

Smith, N. A. (2011). *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Smolensky, P. and Legendre, G. (2005). *The Harmonic Mind: From Neural Computation To Optimality-Theoretic Grammar*. The MIT Press.

Spitkovsky, V. I., Alshawi, H., Jurafsky, D., and Manning, C. D. (2010). Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17. Association for Computational Linguistics.

Stan Development Team (2014). Stan: A c++ library for probability and sampling, version 2.2.

Steinhardt, J. and Liang, P. (2014). Filtering with abstract particles. In *Proceedings of The 31st International Conference on Machine Learning*, pages 727–735.

Sun, R., editor (2008a). *The Cambridge Handbook of Computational Psychology*. Cambridge University Press.

Sun, R. (2008b). Introduction to computational cognitive modeling. In Sun (2008a), pages 3–19.

Swingley, D. (2005). Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology*, 50:86–132.

Synnave, G., Dautriche, I., Börschinger, B., Dupoux, E., and Jo, M. (2014). Word segmentation in context. In *Proceedings of COLING14*.

Talbott, W. (2013). Bayesian epistemology. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Fall 2013 edition.

Teh, Y. W. (2006a). A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, National University of Singapore.

Teh, Y. W. (2006b). A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992.

Teh, Y. W., Jordan, M., Beal, M., and Blei, D. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

Thiessen, E. D. and Saffran, J. R. (2003). When cues collide: use of stress and statistical cues to word boundaries by 7-to 9-month-old infants. *Developmental Psychology*, 39(4):706.

Thiessen, E. D. and Saffran, J. R. (2007). Learning to learn: Infants' acquisition of stress-based strategies for word segmentation. *Language Learning and Development*, 3(1):73–100.

Tomasello, M. (2003). *Constructing a Language: A Usage-Based Approach to Child Language Acquisition*. Cambridge, MA: Harvard University Press.

Venkataraman, A. (2001). A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):351–372.

Yang, C. (2004). Universal grammar, statistics or both? *Trends in Cognitive Sciences*, 8(10):451–456.

Yang, C. (2006). *The infinite gift: How children learn and unlearn the languages of the world*. Simon and Schuster.

Yang, C. and Gambell, T. (2005). Word segmentation: Quick but not dirty.

Zhao, Z., Du, L., Börschinger, B., and Johnson, M. (unpublished). A parallel sparse sampling algorithm for the topical collocation model.