How the statistical revolution changes (computational) linguistics

Mark Johnson

EACL workshop
March 2009

Thanks to Eugene Charniak and the BLLIP group, and Antske Fokkens, John Maxwell, Paola Merlo, Yusuke Miyao and Mark Steedman.

No warranty …

Half the money I spend on advertising is wasted.
The problem is: I don't know which half.

John Wanamaker

What is *computational* linguistics?

- Engineering goals — building useful systems
  - currently relies more heavily on statistics and machine learning than linguistic theory
  - not all scientific knowledge has engineering applications
- Scientific goals — understanding *computational* aspects of linguistic processes
  - language comprehension, production and acquisition (Chomsky's "use of knowledge of language")

Outline

Feature-based grammars and parsing

- Circa 1980 GPSG showed how to decompose long-distance "movement" phenomena into sequences of strictly local dependencies (feature-passing)

⇒ Explosion of mathematical and computational work in "unification-based" grammars that continues today

- Parsing with "unification-based" grammars
  - (computational) linguist devises grammar formalism (e.g., typed feature structure constraints)
  - linguist writes grammar in grammar formalism
  - computational linguist devises parser (inference engine) for grammar formalism

Statistical treebank parsing

- In the late 1980s statistical techniques (e.g., HMMs) became the dominant approach to speech recognition
- Parsing with statistical treebank parsers
  - linguist annotates a corpus with parses
  - computational linguist devises class of statistical models of parses (e.g., features of model)
  - computational linguist devises estimation procedure that maps parsed corpus to statistical model (e.g., feature weights)
  - computational linguist devises parser for statistical model
- *Why has statistical treebank parsing come to dominate computational linguistics?*

Both approaches involve linguistic knowledge

- Linguists write the grammars used in manual grammar-based approaches
- Linguists write the annotation guidelines and direct the annotation effort in statistical treebank parsing
  - how do corpus and annotation choices affect our statistical models?
- Corpus-based approach forces annotators to focus on common constructions (rather than rare constructions that may be scientifically more interesting)
- Large grammars are difficult to construct and maintain (software engineering)
- Treebanks are inherently redundant
  $\Rightarrow$ errors seem less catastrophic

Probabilities and grammars

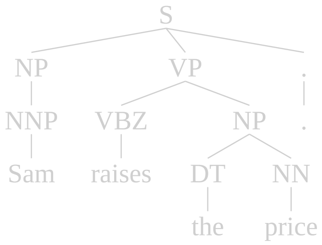
- Abney (1996) showed how to define MaxEnt models over virtually any linguistic representations
  - No requirement that features are independent ("context-free")
  - ⇒ No principled reason why we can't replicate treebank parsing techniques with grammar-based representations
- Are our probabilistic models appropriate?
  - is it reasonable to model P(Sentence)?
  - perhaps we should estimate (and learn from) P(Sentence|Meaning) instead?
  - ⇒ requires *explicit meaning representation*
  - part of attraction of statistical machine translation P(English|French), i.e., French as "language of thought"

Why parse?

- Engineering goal – we hope a general-purpose parser will be useful for other engineering tasks
  - ⇒ broad coverage
  - ⇒ robust
  - ⇒ good *average case* performance
- There are robust, broad-coverage grammar-based parsers
  - HPSG (Miyao and Tsujii, Cholakov et al)
  - LFG (Maxwell and Kaplan, van Genabith and Way)
  - CCG (Hockenmaier, Clark and Curran)
- Complex linguistic representations, but their phrase-structure parsing accuracy is not better than statistical treebank parsers'
- Not that many applications for parsing
  - often used as language model (keep probabilities, throw away parses)

Capturing vs. covering linguistic generalizations

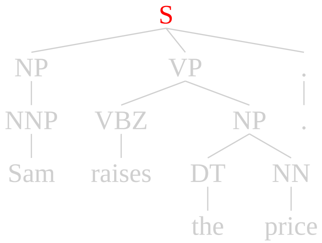
- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies



- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

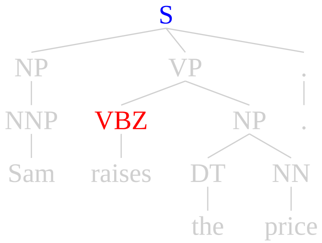
Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact



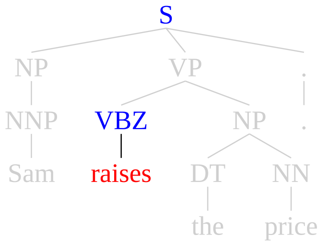
Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact



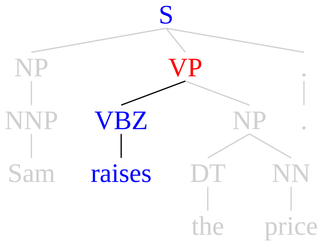
Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An “engineering” parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact



Capturing vs. covering linguistic generalizations

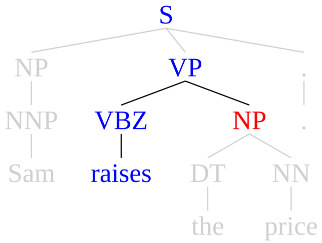
- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies



- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

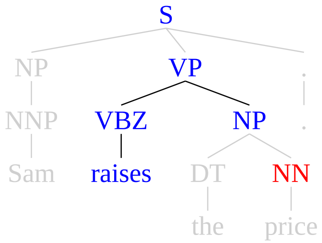
Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact



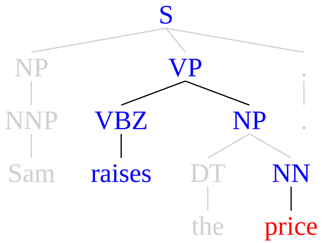
Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact



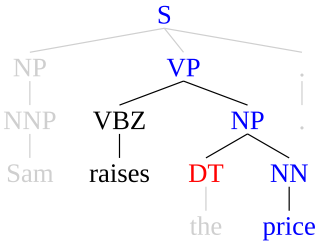
Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact



Capturing vs. covering linguistic generalizations

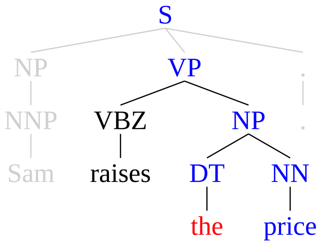
- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies



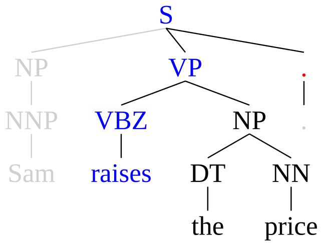
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact



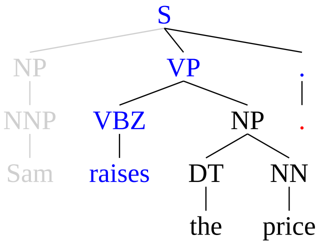
Capturing vs. covering linguistic generalizations



- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

Capturing vs. covering linguistic generalizations

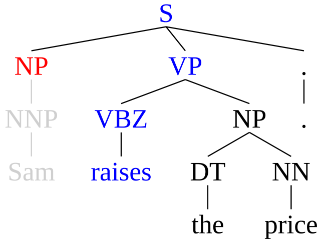
- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies



- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

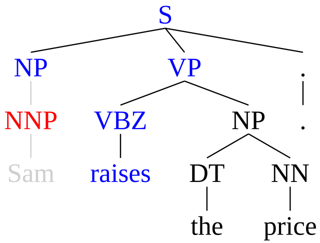
Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies
- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
    - basic linguistic insights tend to have greatest impact



Capturing vs. covering linguistic generalizations

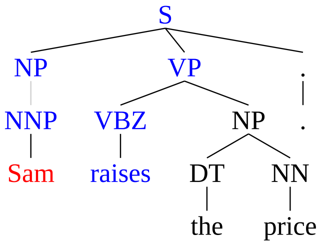
- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies



- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

Capturing vs. covering linguistic generalizations

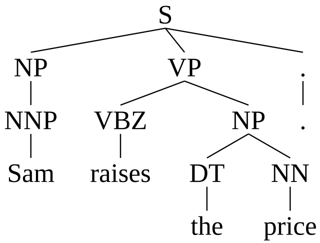
- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies



- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

Capturing vs. covering linguistic generalizations

- *Capturing a generalization*: grammar accurately describes phenomenon at appropriate level, e.g., subject-verb agreement via PERSON and NUMBER features
- *Covering a generalization*: model covers common cases of a generalization, perhaps indirectly. E.g., head-to-head POS dependencies

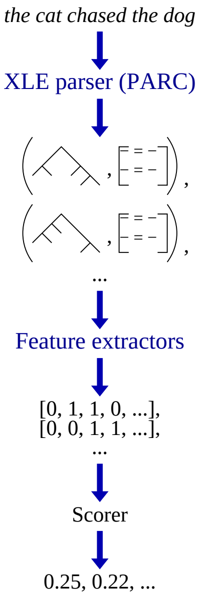


- An "engineering" parser only needs to cover generalizations
- But feature design requires *linguistic insight*
  - basic linguistic insights tend to have greatest impact

Outline

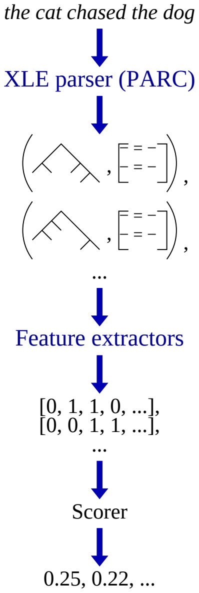
Stochastic Lexical Functional Grammar

- Parc LFG parser produces a set of parses (*c*- and *f*-structure pairs) for input sentence
- A set of "feature extractors" count how often various structures appear in each parse
  - Local trees (i.e., phrase-structure rules)
  - Local *f*-structures (i.e., argument structure frames)
  - Long-distance dependency paths
  - Misc. features: e.g., Coordination features
- Scorer computes "soft max" of weighted linear combination of feature values



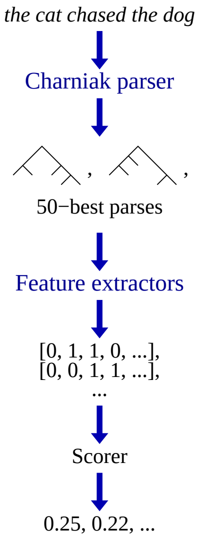
Features in MaxEnt parsing models

- A feature can be *any real-valued function* of the parses
- As far as the statistical model is concerned, *the feature vectors are all that matters*
  - details of linguistic representations are *irrelevant* so long as they are rich enough to compute feature vectors
- The most valuable features were defined on c-structures alone
  - c-structures are finer-grained than f-structures
  - All my f-structure features can be *recoded as* c-*structure features*
- Are there important generalizations that f-structure features cover better?



Discriminative rescoring treebank parser

- Use 50-best parses from Charniak parser
- Log of Charniak's parse probability is one of model's features
  - ⇒ Charniak's features are imported into model
  - ⇒ concentrate on *non-local features* not included in Charniak's model
- Trained from more data than SLFG system was ⇒ more accurate
  - these days it is common to train grammar-based models from PTB translated into grammar-based representations
  - but translated PTB doesn't contain more information than original PTB did



Outline

Conditional estimation for parse rescoring

- Easy to over-fit training data with large number of features

$\Rightarrow$ Regularize by adding a penalty term to log likelihood
  - L1 penalty term $\Rightarrow$ sparse feature weight vector
  - L2 penalty term (Gaussian prior) seems best

- 50-best parses $\mathcal{T}(s)$ may not include true parse $t^\star(s)$

$\Rightarrow$ Train rescorer to prefer parse in $\mathcal{T}_c(s)$ closest to $t^\star(s)$

- Often several parses from 50-best list are equally close to true parse

$\Rightarrow$ EM-inspired (non-convex) loss function

- Direct numerical optimization with L-BFGS (modified for L1 regularizer) produces best results

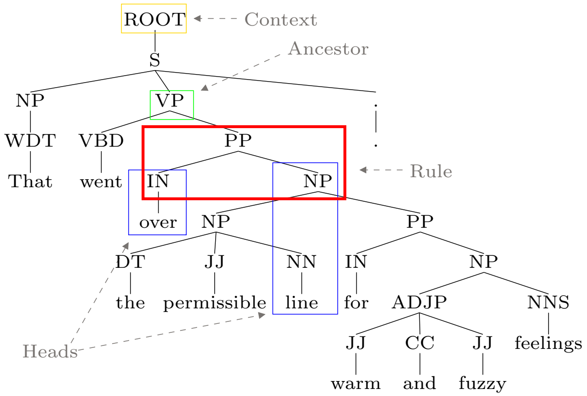
Riezler, King, Kaplan, Crouch, Maxwell and Johnson (2002), Goodman (2004), Andrew and Gao (2007)

Features for rescoring parses

- Parse rescorer's features can be *any computable function* of parse
- Choice of features is the most important and least understood aspect of the parser
  - feature design has a much greater impact on performance than the learning algorithm
- Features can be based on a linguistic theory (or more than one)
- …but need not be
  - "shot-gun" or "hail Mary" features very useful
- Feature selection: a feature's values on $T^{\star}(s)$ and $\mathcal{T}(s) \setminus T^{\star}(s)$ must differ on at least 5 sentences $s \in D$
- The *Charniak parser's log probability* combines all of the generative parser's conditional distributions into a single rescorer feature $\Rightarrow$ rescoring should never hurt

Lexicalized and parent-annotated rules

Lexicalization associates each constituent with its head
Ancestor annotation provides a little "vertical context"
Context annotation indicates constructions that only occur in main clause (c.f., Emonds)



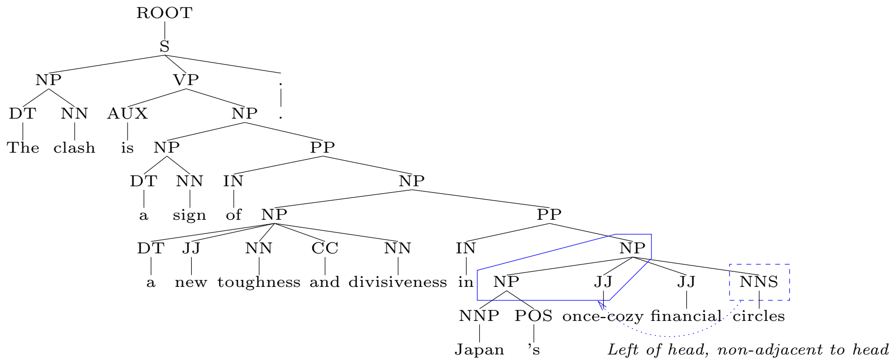
n-gram rule features generalize rules

Collects *adjacent constituents* in a local tree

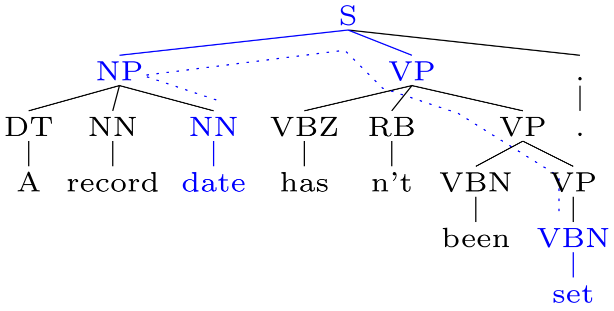
Also includes *relationship to head* (e.g., adjacent? left or right?)

Parameterized by *ancestor-annotation*, *lexicalization* and *head-type*

There are 5,143 unlexicalized rule bigram features and 43,480 lexicalized rule bigram features



Bihead dependency features

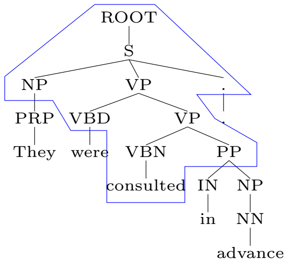


(S (NP (NN date)) (VP (VBN set)))

- Bihead dependency features approximate linguistic function-argument dependencies
- Computed for lexical (≈ semantic) and functional (≈ syntactic) heads
- One feature for each head-to-head dependency found in training corpus (70,000 features in all)

Head trees record all dependencies

- Head trees consist of a (lexical) head, all of its projections and (optionally) all of the siblings of these nodes
- correspond roughly to TAG elementary trees
- parameterized by *head type*, *number of sister nodes* and *lexicalization*

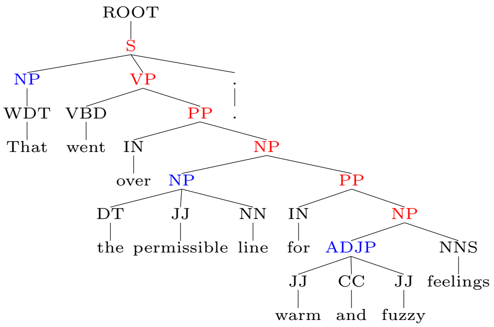


Rightmost branch bias

The RightBranch feature's value is the number of nodes on the right-most branch (ignoring punctuation) (c.f., Charniak 00)

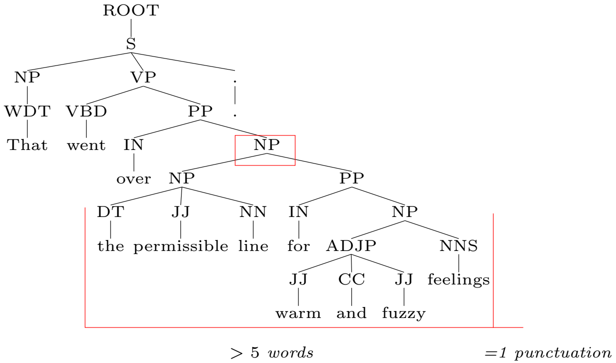
Reflects the tendancy toward right branching in English

Only 2 different features, but very useful in final model!



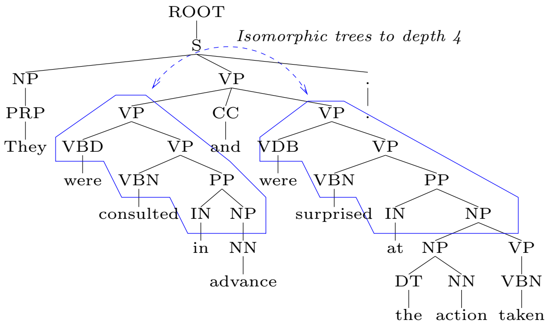
Constituent Heavyness and location

- Heavyness measures the constituent's category, its (binned) size and (binned) closeness to the end of the sentence



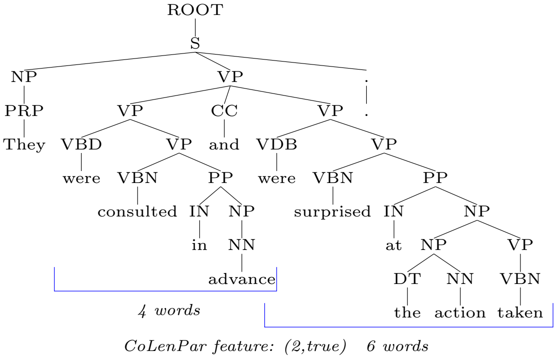
Coordination parallelism (1)

- A CoPar feature indicates the depth to which adjacent conjuncts are parallel



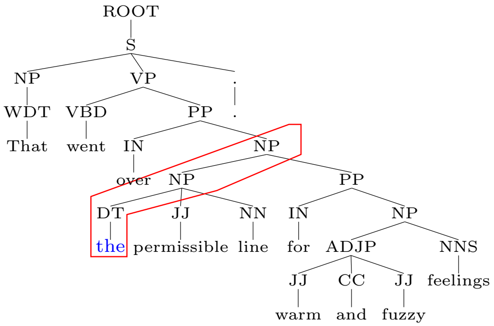
Coordination parallelism (2)

- The CoLenPar feature indicates the difference in length in adjacent conjuncts and whether this pair contains the last conjunct.

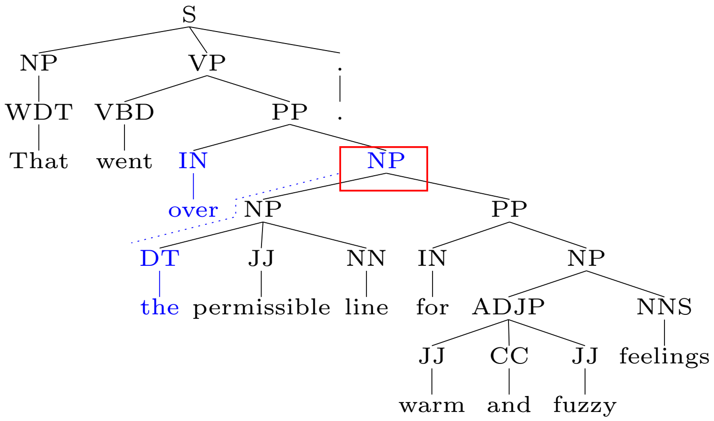


Word

- A Word feature is a word plus n of its parents (c.f., Klein and Manning's non-lexicalized PCFG)
- A WProj feature is a word plus all of its (maximal projection) parents, up to its governor's maximal projection



Constituent "edge neighbour" features

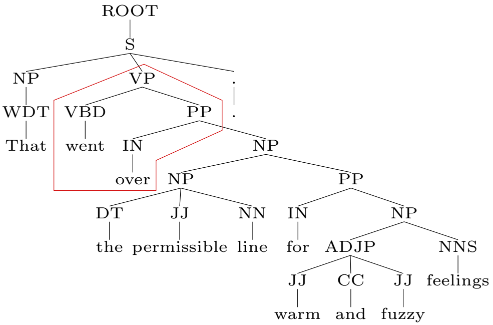


(IN over) (NP (DT the ...))

- Edge features are a kind of bigram context for constituents
- Would be difficult to incorporate into a generative parser

Tree n-gram

- A tree n-gram feature is a tree fragment that connect sequences of adjacent n words, for $n = 2, 3, 4$ (c.f. Bod's DOP models)
- lexicalized and non-lexicalized variants
- There are 62,487 tree n-gram features



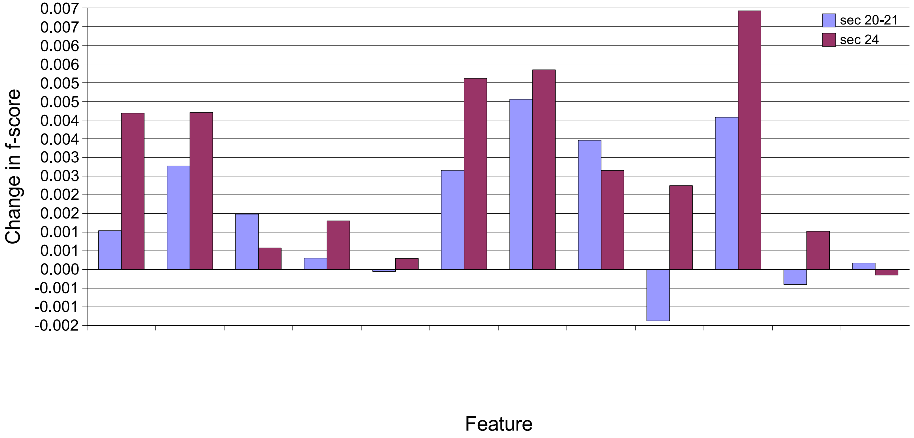
Experimental setup

- Feature tuning experiments done using Collins' split: sections 2-19 as train, 20-21 as dev and 24 as test
- $\mathcal{T}_c(s)$ computed using Charniak 50-best parser
- Features which vary on less than 5 sentences pruned
- Optimization performed using LMVM optimizer from Petsc/TAO optimization package or Averaged Perceptron
- Regularizer constant c adjusted to maximize f-score on dev

Evaluating features

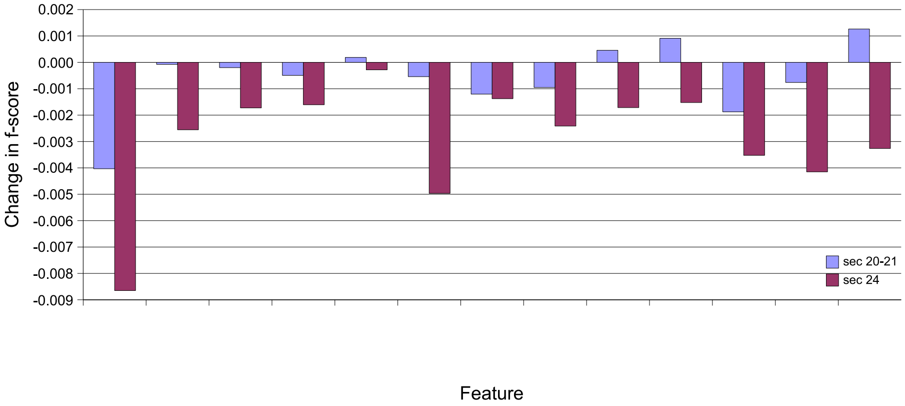
- The feature weights are not that indicative of how important a feature is
- The MaxEnt ranker with regularizer tuning takes approx 1 day to train
- The *averaged perceptron* algorithm takes approximately 2 minutes
  - used in experiments comparing different sets of features
  - Used to compare models with the following features: **NLogP Rule NGram Word WProj RightBranch Heavy NGramTree HeadTree Heads Neighbours CoPar CoLenPar**

Adding one feature class



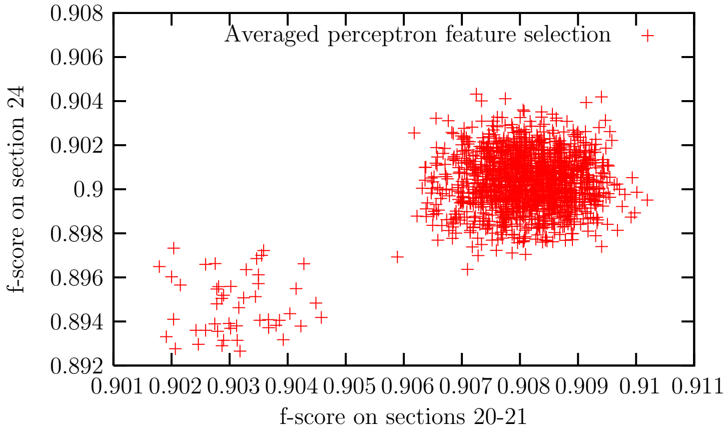
- Averaged perceptron baseline with only base parser log prob feature
  - section 20–21 f-score = 0.894913
  - section 24 f-score = 0.889901

Subtracting one feature class



- Averaged perceptron baseline with all features
  - section 20–21 f-score = 0.906806
  - section 24 f-score = 0.902782

Feature selection is hard!



Greedy feature selection using *averaged perceptron* optimizing f-score on sec 20–21

All models also evaluated on section 24

Comparing estimators

- Training on sections 2–19, regularizer tuned on 20–21, evaluate on 24

Estimator	# features	sec 20-21	sec 24
MaxEnt model, $p = 2$	670,688	0.9085	0.9037
MaxEnt model, $p = 1$	14,549	0.9078	0.9024
averaged perceptron	523,374	0.9068	0.9028
expected f-score	670,688	0.9084	0.9029

- None of the differences are significant
- Because the exponential model with $p = 2$ was the first model I tested new features on, they may be biased to work well with it.

Outline

Conventional grammars are closed-world

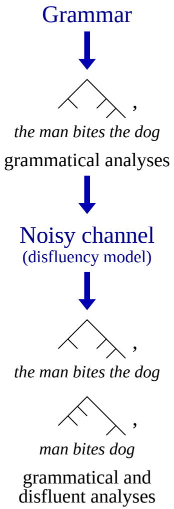
- The *closed world assumption* for grammars
  - Rules and lexical entries define set of grammatical structures
  - Everything not grammatical is ungrammatical
- "Parsing as deduction": parsing is the process of proving the grammaticality of a sentence
- But: goal is *understanding* what the speaker's trying to say; not determining whether the sentence is grammatical
- My ideal parser would be *open-world*
  - even ungrammatical sentences are interpretable
    E.g., *man bites dog* $\neq$ *dog bites man*
  - words and constructions we recognize provide information about sentence's meaning
  - unknown words or phrases do not cause interpretation to fail
    - parsing and acquisition are two aspects of same process
- *Statistical treebank parsers are open-world*
  - every possible tree receives positive probability

Open-world grammar-based parsing via *relaxation*

- Replace "hard" unification equality constraints with "soft" probabilistic features
  - e.g., for subject-verb agreement, introduce feature that fires when subject's agreement disagrees with verb's agreement
- Most techniques for making grammar-based parsers broad-coverage perform grammar or lexicon relaxation
  - Unknown word lexical entry guesser
  - Fragment parsing mechanisms
- Under this approach, grammaticality does not play a central role in parsing
- Statistical treebank parsers relax via *smoothing* and *Markovization*

Open-world grammar-based parsing via *disfluency model*

- Intuition: Ungrammatical sentences are understood by analogy with grammatical ones
- Possible formulation: noisy channel model where channel generates disfluencies
  - source model only generates *grammatical* sentences
- We've used this kind of model to detect and correct disfluencies in speech transcripts



Outline

Statistical models in linguistics

- Scientific side of computational linguistics studies the computational aspects of language
  - computation (the meaning-respecting manipulation of meaning-bearing symbols) is a process
  - ⇒ computational linguistics bears most directly on language comprehension, production and acquisition
- Recent explosion of interest in:
  - computational psycholinguistics (esp. sentence processing) (Bachrach, Hale, Keller, Levy, Roark, etc.)
  - computational models of language acquisition (esp. phonology and morphology)
    (Albright, Boersma, Frank, Goldsmith, Hayes, Pater, etc.)

Optimality theory

- Smolensky's *Optimality Theory* (OT) seems exquisitely designed to permit numerophobes perform optimization
- Detailed OT analyses of many aspects of phonology and morphology
- Optimality Theory is a limiting case of *Harmony Theory* which are equivalent to MaxEnt models
  - OT constraints $\equiv$ MaxEnt features

$\Rightarrow$ Easy for linguists to move from OT analyses to MaxEnt models

Chomsky ought to be a Bayesian!

$$\underbrace{\mathrm{P(Grammar \mid Data)}}_{\text{Posterior}} \propto \underbrace{\mathrm{P(Data \mid Grammar)}}_{\text{Likelihood}} \underbrace{\mathrm{P(Grammar)}}_{\text{Prior}}$$

- Likelihood measures how well grammar describes data
- Prior expresses knowledge of grammar before data is seen
  - can be very specific (e.g., Universal Grammar), or
  - can be generic (e.g., prefer shorter grammars)
  - can express *markedness preferences* that can be overridden by data
- Posterior is *distribution* over grammars
  - expresses uncertainty about which grammar is correct
- In principle we can *empirically evaluate* utility of specific linguistic universals in language acquisition

Bayesian updating integrates parsing and learning

$$\underbrace{\mathrm{P}(\text{Grammar} \mid \text{Sentence}_{1:n})}_{\text{Updated grammar}}$$
$$\propto \underbrace{\mathrm{P}(\text{Sentence}_n \mid \text{Grammar})}_{\approx \text{ parsing}} \underbrace{\mathrm{P}(\text{Grammar} \mid \text{Sentence}_{1:n-1})}_{\text{Previous grammar}}$$

- *Incremental Bayesian belief updating* updates posterior as each data item is encountered
- With e.g. PCFGs the update process requires computing expected rule counts, and adding these to rule count sums.
- *Sampling the sentence's likely parses* is a general way of computing these expectations

⇒ *Learning might be a low-cost by-product of parsing*

Non-parametric Bayesian inference

- Most standard learning methods are based on *optimizing a function of a finite vector of real-valued parameters*
- Learn rules or structures by iterating:
  - generate hopefully useful rules/structures
  - estimate their parameters via a parametric estimator
  - prune low-probabilty rules/structures
- Non-parametric Bayesian methods provide a theoretically sound framework for identifying relevant finite subset from infinite set of entities
  - example: learn finite set of phonemic forms from infinite set of potential phonemic forms
  - Monte Carlo sampling methods

Computational models of word learning

- During their "word spurt", children learn words in "one-shot"
- No obvious limit on number of possible words
  $\Rightarrow$ non-parametric Bayes
- Learn to segment words from *unsegmented broad phonemic transcription* (Brent)
  - Example: *y u w a n t t u s i D 6 b u k*
- Goldwater (2006) learns a bigram model of word sequences *without being told what the words are*
  - Learning inter-word dependencies improves word segmentation
  - Learning intra-word structure (syllable structure) also helps
- There are learning algorithms whose computationally most demanding component is sampling parses of input sequences

Outline

Conclusion

- Be clear about why you're building any computational model
  - usual engineering goals $\Rightarrow$ average case matters
  - open-world parsing (unknown words, fragment parsing)
  - don't have to *capture* a generalization in order to *cover* it
  - what does determining grammaticality have to do with comprehension?
- Statistical models of processing and acquisition are having a major impact
  - probabilistic parsing models are having an increasing impact in psycholinguistics and neurolinguistics
  - (non-parametric) Bayesian models provide new ways of viewing language acquisition