Parsing Speech Corpora

Mark Johnson

joint work with Eugene Charniak and Matt Lease
(prosody with Mari Ostendorf and Jeremy Kahn at U Washington)

Brown University

Supported by NSF grants LIS 9720368 and IIS0095940
and DARPA GALE contract HR0011-06-2-0001

2006 CUNY conference

Outline

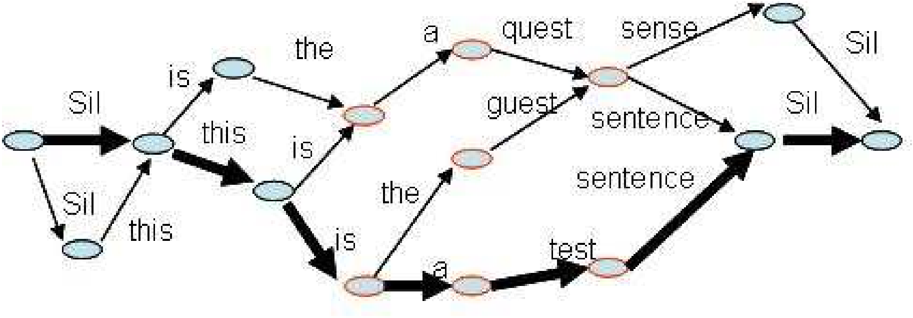
Why is parsing speech difficult?

- Speech is *rarely segmented into words, phrases or even sentences*
- Word identity is not as clear as in text
- Speech often contains *disfluencies*
- *Conversational speech* poses additional problems
  - overlapping turns
  - turns don't correspond to phrases or sentences
  - much higher disfluency rate
- but *prosodic cues* provide additional information

Hirschberg (2002)

Acoustic ambiguity and word lattices

…recognize speech …
…wreck a nice beach …



“Noisy channel” model of speech recognition

Language model P(*Words*)
(trigram model, statistical parser)

Source signal
$\ldots$ *recognize speech* $\ldots$

Channel model P(*Acoustics*|*Words*)

Acoustic features
$[\ r\ e\ k\ o\ g\ n\ ay\ z\ s\ p\ i\ ch\ ]$

- *Bayes rule* permits us to invert the channel

$$\mathrm{P}(\mathit{Words}|\mathit{Acoustics}) \propto \underbrace{\mathrm{P}(\mathit{Acoustics}|\mathit{Words})}_{\text{Acoustic model}} \underbrace{\mathrm{P}(\mathit{Words})}_{\text{Language model}}$$

Jelinek (1997) “Statistical methods for speech recognition”

n-gram language models

- A *language model* estimates the probability of strings of words in a language
  - used to distinguish likely from unlikely paths in the lattice
- *n*-gram language model predicts each word based on the $n-1$ preceding words
  - most commonly $n = 3$ (trigrams) or $n = 4$ (quadgrams)

$$\begin{aligned} & \mathrm{P}(\textit{this is a test sentence}) \\ & \quad \approx \quad \mathrm{P}(\textit{this})\,\mathrm{P}(\textit{is}|\textit{this})\,\mathrm{P}(\textit{a}|\textit{is})\,\mathrm{P}(\textit{test}|\textit{a})\,\mathrm{P}(\textit{sentence}|\textit{test}) \end{aligned}$$

- These conditional probabilities can be *estimated from raw text*
  - speech recognizer language models often estimated from billions of words of text
- computationally *simple and efficient*
- surprisingly effective at distinguishing English from word salad

Outline

Generative statistical parsers

- Probabilistic model associates trees and probabilities to *all possible sequences of words*
- Tree predicted node by node using *function-argument dependencies*
- A statistical *parser* returns the most probable tree for *Words*

$$\widehat{Tree} \;=\; \underset{Tree}{\operatorname{argmax}} \mathrm{P}(Tree \mid Words)$$

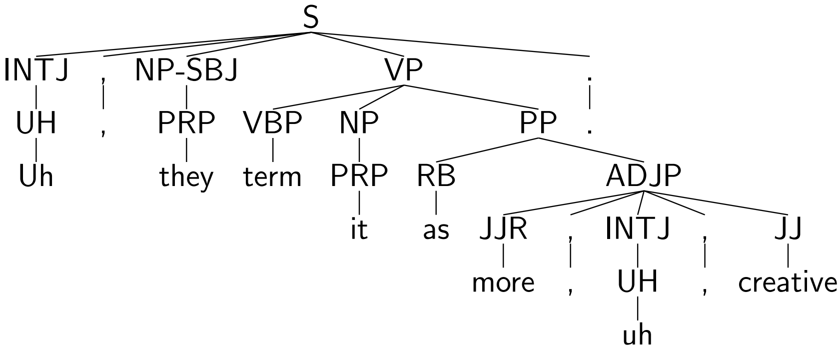
- A parser *language model* returns the probability of *Words*

$$\mathrm{P}(Words) \;=\; \sum_{Tree} \mathrm{P}(Tree, Words)$$

- Parser language models can work directly from lattices
- Parser language models can do better than n-gram models trained on the same data

Charniak (2001), Chelba and Jelinek (1998), Collins (2003), Hall and Johnson (2003), Roark (2001)

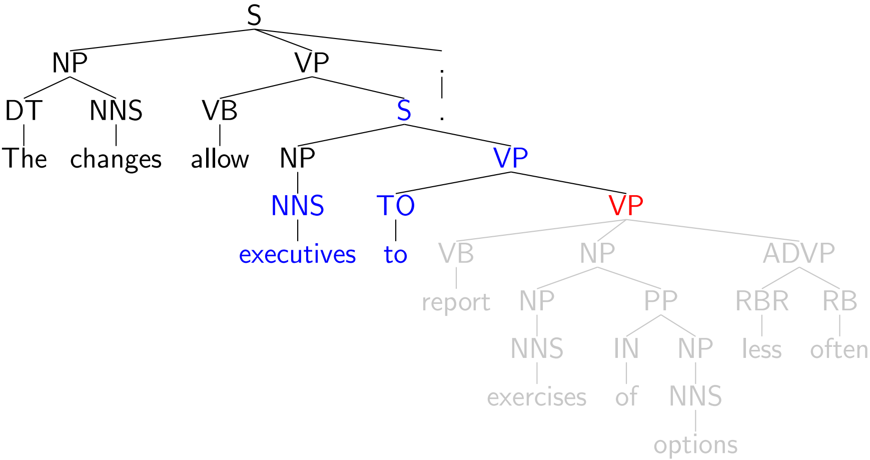
Treebank training data for statistical parsers



- The Switchboard corpus contains 1.2 million words of telephone conversational speech with syntactic and disfluency annotation

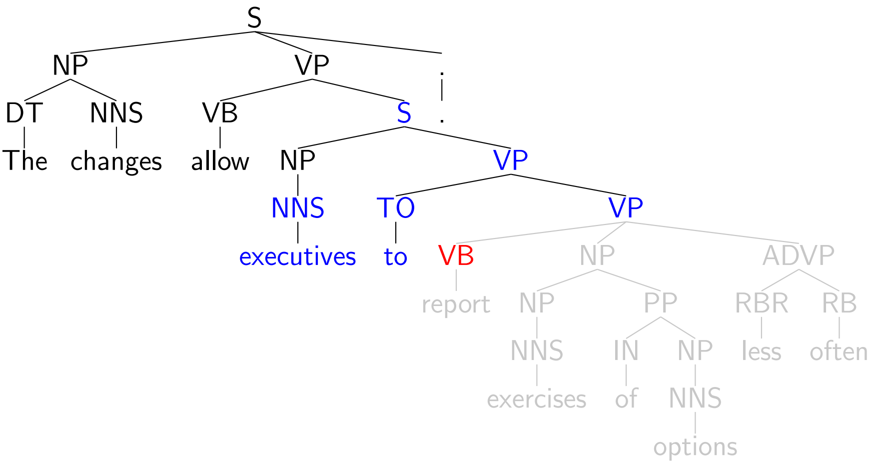
Marcus, Santorini and Marcinkiewicz (1993)

Generative language model (Charniak 2001)



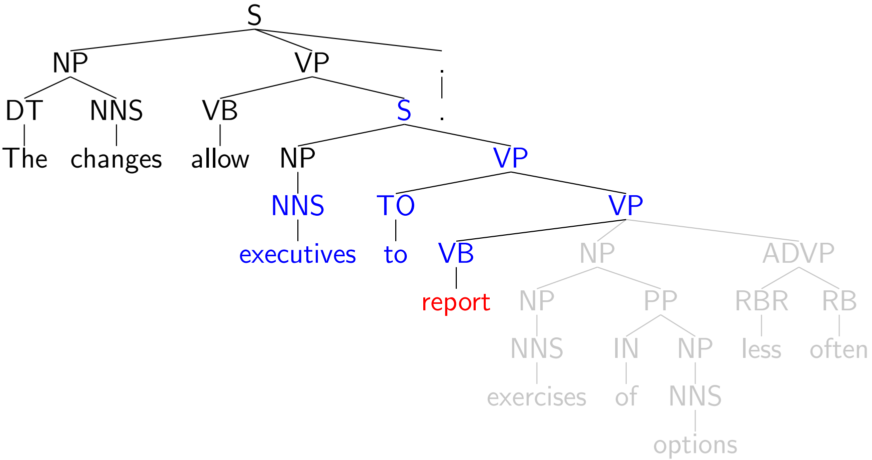
- Predicted node is shown in red
- Conditioning nodes are shown in blue

Generative language model (Charniak 2001)



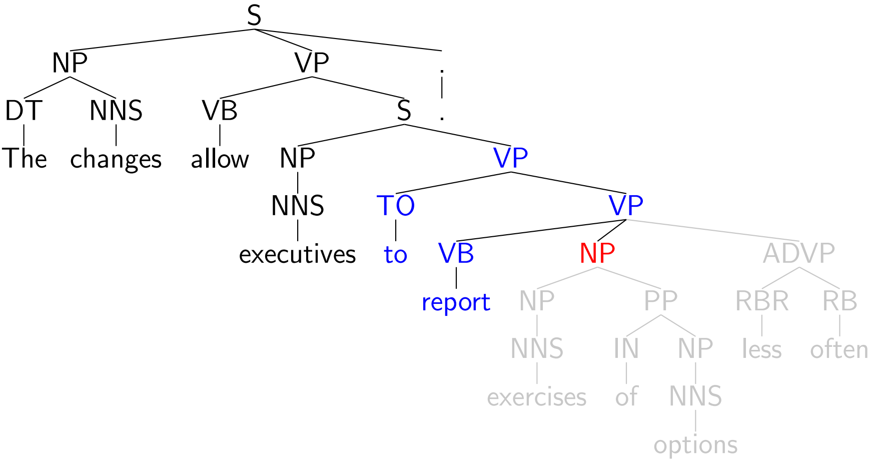
- Predicted node is shown in red
- Conditioning nodes are shown in blue

Generative language model (Charniak 2001)



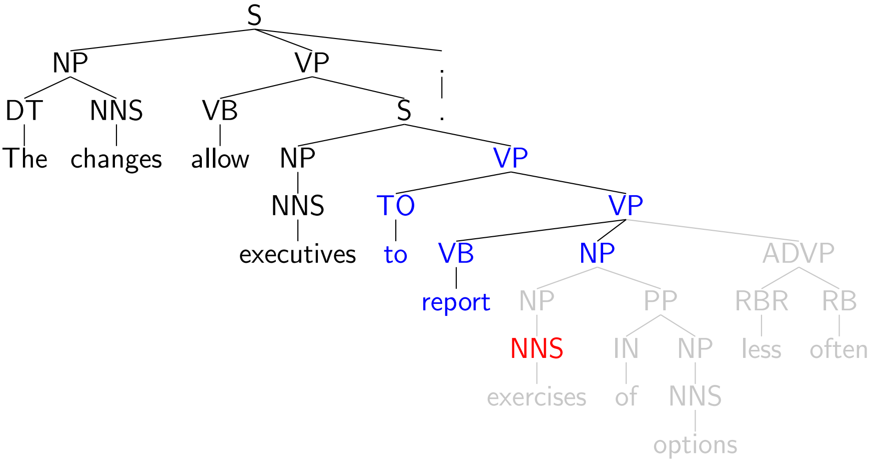
- Predicted node is shown in red
- Conditioning nodes are shown in blue

Generative language model (Charniak 2001)



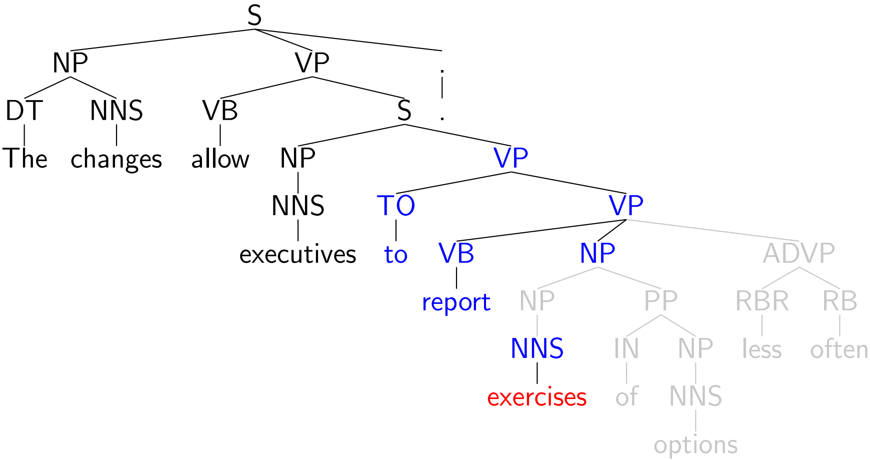
- Predicted node is shown in red
- Conditioning nodes are shown in blue

Generative language model (Charniak 2001)



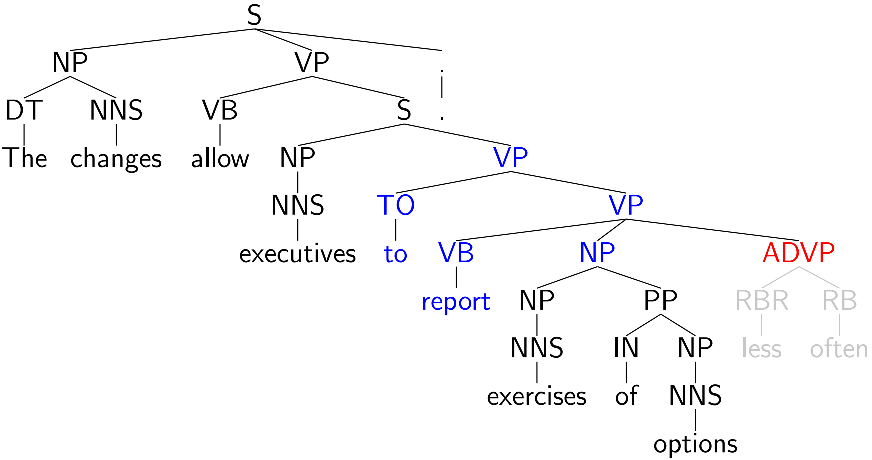
- Predicted node is shown in red
- Conditioning nodes are shown in blue

Generative language model (Charniak 2001)



- Predicted node is shown in red
- Conditioning nodes are shown in blue

Generative language model (Charniak 2001)



- Predicted node is shown in red
- Conditioning nodes are shown in blue

Outline

Why is speech difficult?

Statistical parser language models

Discriminative reranking

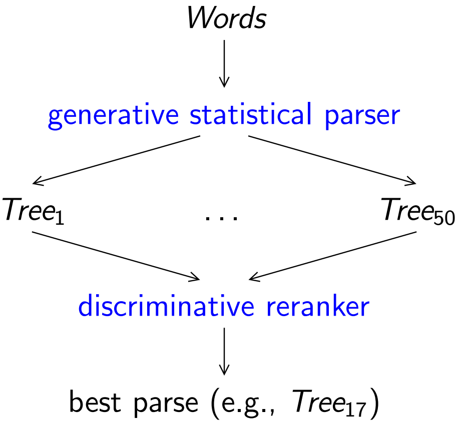
Parsing, punctuation and prosody

Detecting and correcting speech repairs

Discriminative reranking for speech

Conclusion

Discriminative reranking parsers



- Generative parser produces 50 most likely trees per sentence
- Discriminative reranker selects best tree using *much wider range of features than generative parser*
- *cannot be used for language modeling*

Charniak and Johnson (2005), Collins (2000), Johnson et al (1999)

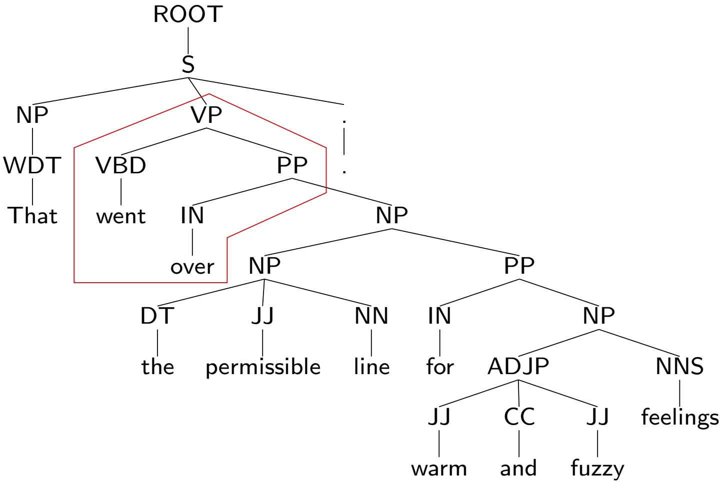
Features for discriminative reranking

- Discriminative rerankers use machine-learning techniques to select best parse tree from set of candidate parses
- Features can be *any real-valued function of parse trees* (generative parsers use function-argument dependencies)
- Our discriminative reranker has two kinds of features:
  - The tree's probability estimated by generative parser
  - The number of times particular configurations appear in the parse
- Rerankers can have hundreds of thousands of features
- Improves parsing significantly
  - best generative parsers' accuracy $= 0.90$
  - discriminative reranker accuracy > 0.92 (20% error reduction)

Collins and Koo (2005), Johnson (2005)

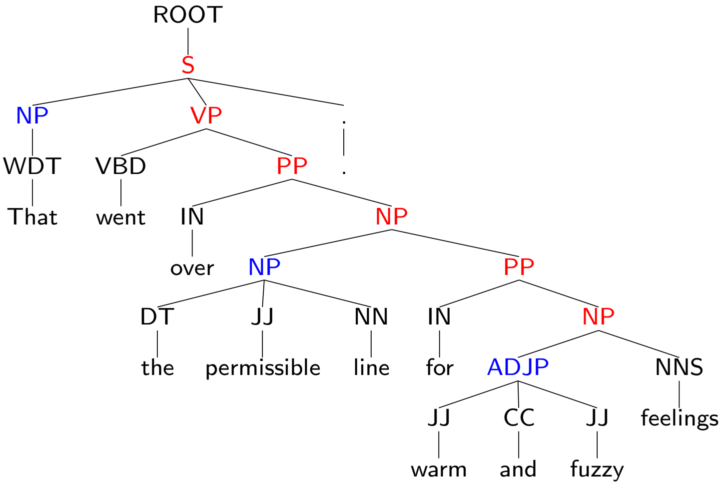
Tree n-gram

- 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, TAG local trees)
- lexicalized and non-lexicalized variants



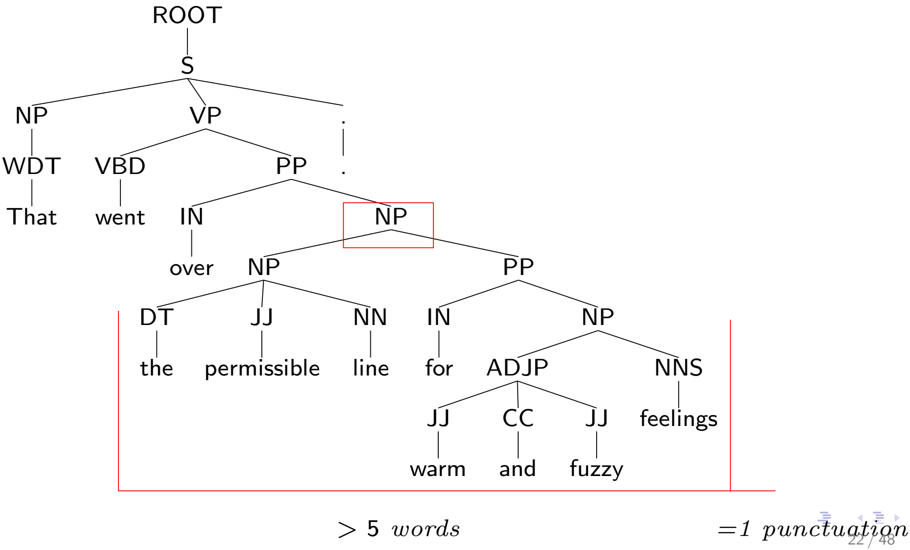
Rightmost branch feature

- The RightBranch feature indicates whether a node lies on the rightmost branch
- Reflects the tendancy toward right branching in English



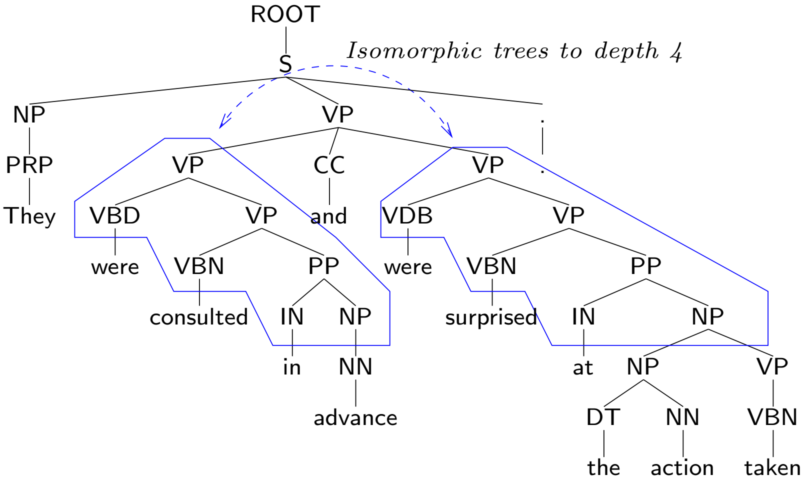
Constituent Heavyness and location

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



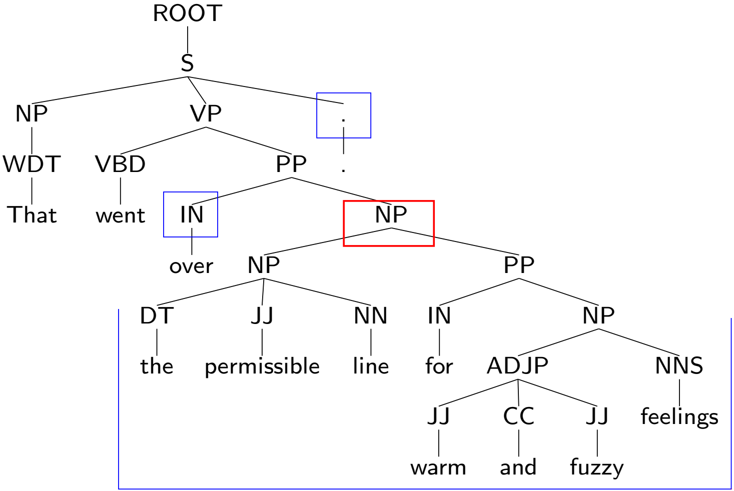
Coordination parallelism

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



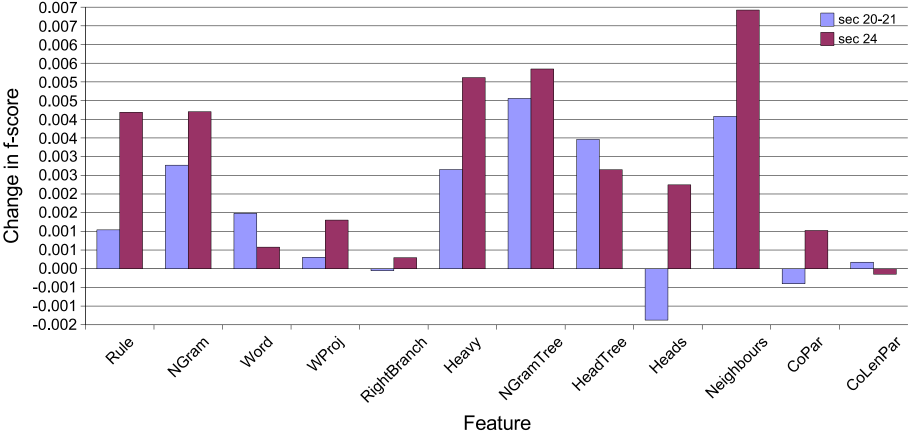
Neighbours

- A Neighbours feature indicates the node's category, its binned length and j left and k right POS tags for $j, k \leq 1$



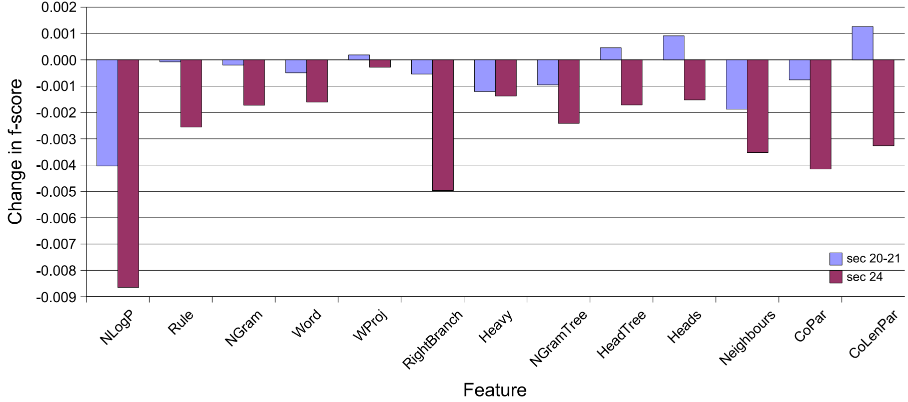
> 5 *words*

Accuracy improvement adding one feature class



- Parse accuracy measured using *f-score* on two development sections of WSJ treebank
- Generative parser's accuracy on sections 20–21 = 0.895 and on section 24 = 0.890

Accuracy decrease removing one feature class



- Accuracy with all features on sections 20–21 $= 0.9068$ and on section 24 $= 0.9028$
- Features are highly redundant and interact in complex ways

$\Rightarrow$ difficult to tell just which features are most important

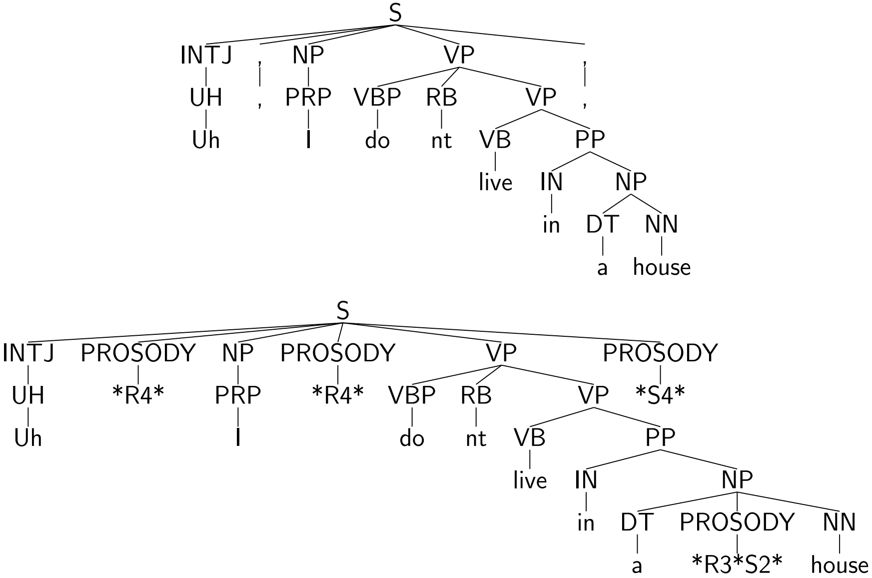
Outline

Parsing, punctuation and prosody – a first attempt

- Punctuation *significantly improves* parsing accuracy
  - no punctuation = 0.869, with punctuation = 0.882
- Prosody is strongly correlated with constituent boundaries
- Perhaps inserting prosodic information into tree mimicking punctuation will improve parsing?
- Prosodic features used (from Ferrer 2002 at SRI)
  - normalized pause duration
  - normalized last rhyme duration
  - log F0 deviation
  - F0 slope

Ferrer (2002), Hirschberg and Nakatani (1998)

"Prosody as pseudo-punctuation" example



“Prosody as pseudo-punctuation” results

- All of the different combinations of prosodic features we tried decreased parsing accuracy
  - accuracy with punctuation $= 0.882$
  - accuracy with no punctuation or prosody $= 0.869$
  - accuracy with prosody $= 0.848$–0.867 (depending on details)

$\Rightarrow$ Our prosodic features do not contain the same information that punctuation does

- Inserting extra pseudo-terminals may interfere with generative parser’s limited conditioning window
  - prosody pseudo-punctuation is crowding-out real lexical items?
- Might work better with real speech (rather than transcripts)

Gregory, Johnson and Charniak (2004)

Outline

Speech errors in (transcribed) speech

- Restarts and repairs

  Why didn't he, why didn't she stay at home?

  I want a flight *to Boston, uh,* to Denver on Friday

- Filled pauses

  I think it's, *uh*, refreshing to see the, *uh*, support ...

- Parentheticals

  But, *you know*, I was reading the other day ...

- "Ungrammatical" constructions

Bear, Dowding and Schriberg (1992), Charniak and Johnson (2001), Core and Schubert (1999), Heeman and Allen (1999), Nakatani and Hirschberg (1994), Stolcke and Schriberg (1996)

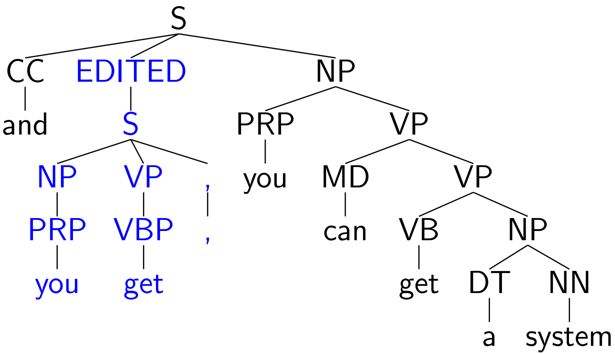
The structure of repairs

$$\ldots\text{and } \underbrace{\text{you get,}}_{\text{Reparandum}}\ \underbrace{\text{uh,}}_{\text{Interregnum}}\ \underbrace{\text{you can get}}_{\text{Correction}} \text{ a system } \ldots$$

- The Reparandum is *often not a syntactic phrase*
- The Interregnum is usually lexically and prosodically marked, but can be empty
- The Reparandum is often a *"rough copy"* of the Correction
  - Repairs are typically short
  - Correction can sometimes be completely different to Reparandum

Shriberg 1994 "Preliminaries to a Theory of Speech Disfluencies"

Treebank representation of repairs



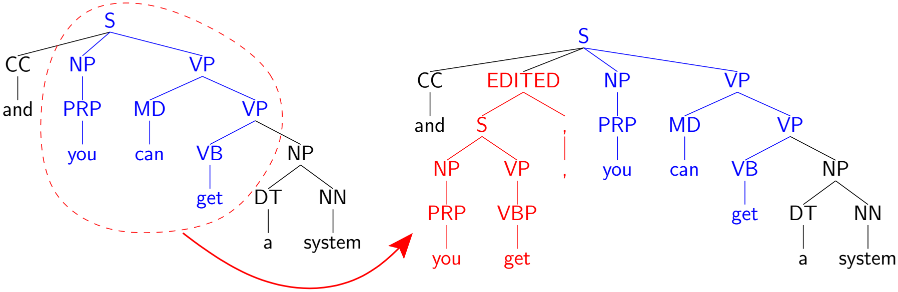
- The *Switchboard treebank* contains the parse trees for 1M words of spontaneous telephone conversations
- Each reparandum is indicated by an EDITED node (interregnum and repair are also annotated)
- But generative parsers are very poor at finding them!

The "true model" of repairs (?)

$$\ldots\text{and } \underbrace{\text{you get,}}_{\text{Reparandum}} \; \underbrace{\text{uh,}}_{\text{Interregnum}} \; \underbrace{\text{you can get}}_{\text{Correction}} \text{ a system } \ldots$$

- Speaker generates intended "conceptual representation"
- Speaker incrementally generates syntax and phonology,
  - recognizes that what is said doesn't mean what was intended,
  - "backs up", i.e., partially deconstructs syntax and phonology, and
  - starts incrementally generating syntax and phonology again
- but without a good model of "conceptual representation", this may be hard to formalize ...

Approximating the "true model" (1)



- Approximate semantic representation by *syntactic structure*
- Tree with reparandum and interregnum excised is what speaker intended to say
- Reparandum results from attempt to generate Correction structure
- Dependencies are *very different to those in "normal" language!*

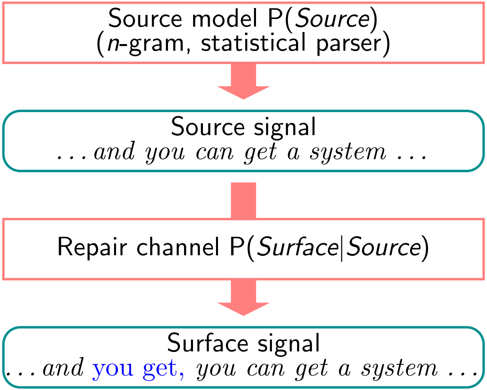
Approximating the "true model" (2)

I want a flight $\underbrace{\text{to Boston,}}_{\text{Reparandum}}$ $\underbrace{\text{uh, I mean,}}_{\text{Interregnum}}$ $\underbrace{\text{to Denver}}_{\text{Correction}}$ on Friday

- Use Correction string as approximation to intended meaning
- Reparandum string is "rough copy" of Correction string
  - involves *crossing* (rather than *nested*) dependencies
  - explains why standard (PCFG-based) generative parsers are bad at finding them
- String with reparandum and interregnum excised is well-formed
  - after correcting the error, what's left should have high probability
  - *use model of normal language to identify ill-formed input*

$\Rightarrow$ Use a *noisy channel model* to analyse repairs

A noisy channel model for speech repairs



- Noisy channel model combines language model and repair model
- *Bayes rule* describes how to invert the channel

$$\mathrm{P}(\mathit{Source}|\mathit{Surface}) \;\propto\; \mathrm{P}(\mathit{Surface}|\mathit{Source})\mathrm{P}(\mathit{Source})$$

The TAG channel model for repairs

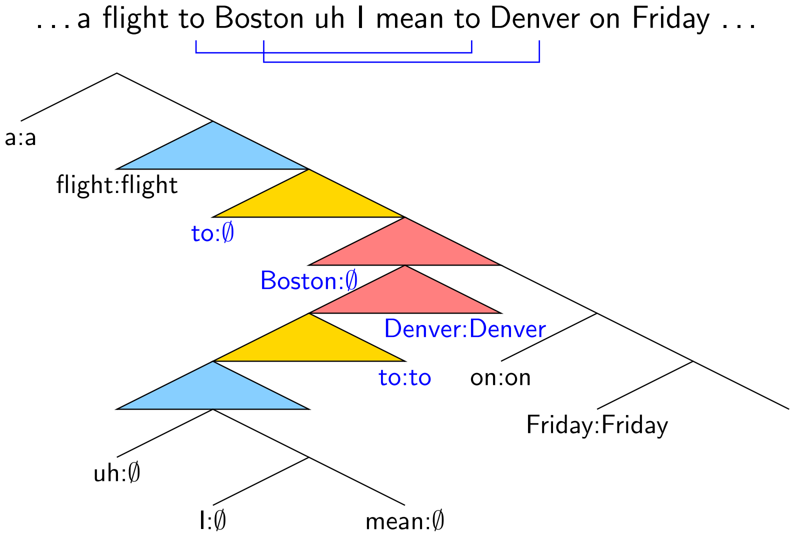
I want a flight $\underbrace{\text{to Boston,}}_{\text{Reparandum}}$ $\underbrace{\text{uh, I mean,}}_{\text{Interregnum}}$ $\underbrace{\text{to Denver}}_{\text{Correction}}$ on Friday

- Channel model is a *probabilistic transducer* producing *source:output pairs*

  . . . a:a flight:flight ∅:to ∅:Boston ∅:uh ∅:I ∅:mean to:to Denver:Denver . . .
- *Reparandum* is "rough copy" of Correction
  - We need a probabilistic model of rough copies
  - FSMs and CFGs *can't generate copy dependencies* . . .
  - but *Tree Adjoining Grammars* can
  - the TAG does not describe familiar linguistic dependencies

Johnson and Charniak (2004)

Schematic TAG channel derivation



Evaluation of model's performance

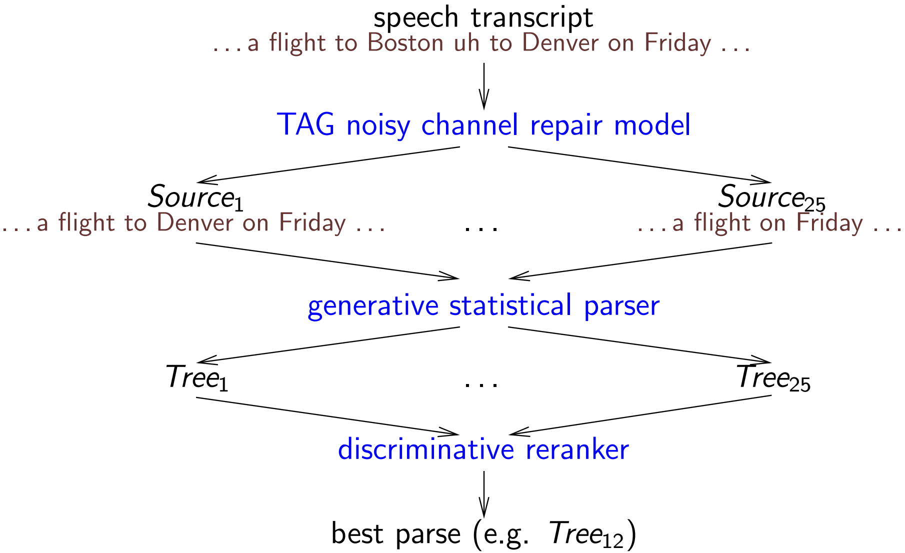
	Classifier	Bigram	Parser
Precision	0.974	0.781	0.810
Recall	0.600	0.737	0.778
F-score	**0.743**	**0.758**	**0.794**

- We can run the noisy channel with different language models
  - "Bigram" is the TAG channel model with a bigram language model
  - "Parser" is the TAG channel model with a generative parser language model
  - Classifier is a word-by-word classifier using machine-learning techniques
- Machine-learning classifier uses lots of local features $\Rightarrow$ more accurate on short repairs
- Noisy channel model is more accurate on longer repairs

Charniak and Johnson (2001)

Outline

Discriminative reranking for speech repairs



Prosody in discriminative reranking for repairs

- Input to discriminative reranker can contain
  - TAG channel model probabilities
  - generative parser probabilities
  - local features (e.g., the ones used in "machine learning" classifier)
  - location and syntactic context of each repair
  - *prosodic features* supplied by M. Ostendorf (normalized pause duration in reparandum and normalized pause duration elsewhere)

Features used	**Speech recognizer**	**Human transcript**
Local + Parser + TAG + Prosody	75.8%	52.8%
Local + Parser + TAG	76.4%	54.3%
Local + TAG + Prosody	76.7%	55.0%
Local + Parser + Prosody	81.0%	56.5%

Edited word detection error rate on RT04 data

Johnson, Charniak and Lease (2005)

Prosody in discrimative reranking for parsing

- Output of the repair detector → discriminative reranking parser
- Reranker incorporates *prosody* × *syntax* features
  - Cooccurence of binned "break probability" and right edge of phrasal category

	No repair detector	TAG repair detector	True repairs
Parser	0.844	0.850	0.869
Parser + Prosody	0.850	0.856	0.876
Parser + Syntax	0.859	0.864	0.884
All features	0.860	0.866	0.886

Parsing accuracy on Switchboard speech data with varying reranker features

Kahn, Lease, Charniak, Johnson and Ostendorf (2005)

Outline

Conclusion

- Speech presents a lot of problems (ambiguity, turns, disfluencies, etc.) and some opportunities (prosody) relative to text
- Generative parsing algorithms model “function argument” dependencies in language
- Discriminative rerankers can incorporate a much wider set of dependencies
- Even though prosody seems analagous to punctuation, treating prosody as punctuation doesn’t work
- Disfluencies involve “rough copy” rather than “function argument” dependencies

  $\Rightarrow$ TAG noisy-channel model and parser language model
- Discriminative rerankers can combine parser, TAG channel model and prosody to optimize repair detection and parse accuracy