

# Parsing and Language Modeling

Mary Harper, Zhongqiang Huang, and Denis Filimonov

10/11/2007



## Outline

- Overview
- Parsing Models for Mandarin Chinese
- Language Models
- Future Work

## Overview

- The overarching goal of our Gale research has been to construct Mandarin language processing tools to support:
  - speech recognition through structured language modeling,
  - annotation tasks (e.g., named entity detection, sentence segmentation/punctuation),
  - machine translation.
- Our current efforts have focused on building capability in Mandarin part-of-speech (POS) tagging and statistical parsing, with emerging research on LMs.

## Mandarin Parsers

- Charniak Parser
- Berkeley Latent Annotation PCFG parser

# Charniak Parser

$$p(\pi, s) = \prod_{c \in \pi} p(h(c)|t(c), hp(c), tp(c)) \cdot p(r(c)|h(c), t(c), tp(c))$$

- $c$  is a constituent in the parse tree  $\Pi$  of sentence  $s$
- $t(c)$  is the constituent type of  $c$  (e.g., if  $c$  is a prepositional phrase,  $t(c) = PP$ )
- $h(c)$  is the head word of  $c$
- $p(c)$  is the parent constituent of  $c$
- $hp(c)$  is the head word of the parent constituent of  $c$
- $tp(c)$  is the type of the parent constituent of  $c$

10/11/2007

U Maryland /Purdue

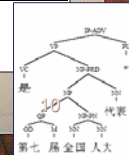
5

# Mandarin Parsing

- Charniak's Parser
  - Relatively low performance due to simple port of English parser

10/11/2007

U Maryland /Purdue



## Tree Transformations: Unary Rules

- The CTB requires phrasal projection for all lexical categories. As a result, 41% of the local tree tokens are unary rules (Levy & Manning, ACL'03).
- **Hypothesis:** Non-preterminal unary rules are often not posited by statistical parsers.
- Using Tsurgeon, we created scripts to modify the training and test trees to determine the impact of unary rules on parsers tasked with learning the grammar in the Chinese Penn Treebank. Available at: <http://nlp.stanford.edu/software/tsurgeon.shtml>

Ma

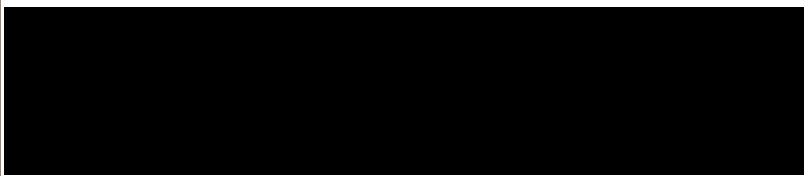
## Example of Unary Rule Removal, Tsurgeon Style

```
(ROOT
  (SBARQ
    (SQ (NP (NNS Cats))
      (VP (VBP do)
        (VP (WHNP what)
          (VB eat))))))
  )
) → (ROOT
  (SQ (NP (NNS Cats))
    (VP (VBP do)
      (VP (WHNP what)
        (VB eat))))))
```

```
@SBARQ=sbarq <: SQ
excise sbarq sbarq
```

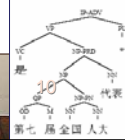
# Mandarin Parsing

- Charniak's Parser
  - Relatively low performance due to simple port of English parser

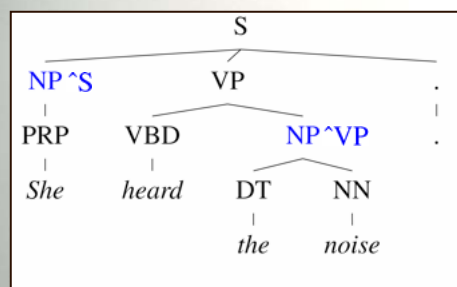


10/11/2007

U Maryland /Purdue

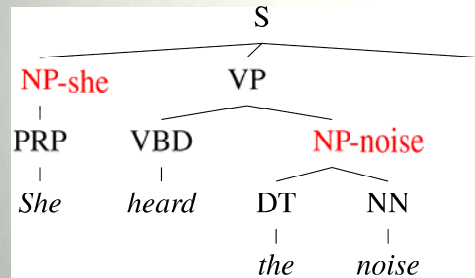


# The Game of Designing a Grammar



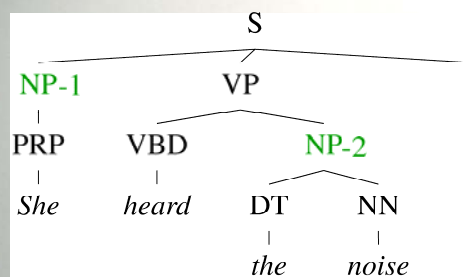
- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]

## The Game of Designing a Grammar



- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]
  - Head lexicalization [Collins '99, Charniak '00]

## The Game of Designing a Grammar

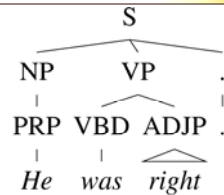


- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]
  - Head lexicalization [Collins '99, Charniak '00]
  - Automatic clustering

## Previous Work: Manual Annotation [Klein & Manning '03]

- Manually split categories

- NP: subject vs object
- DT: determiners vs demonstratives
- IN: sentential vs prepositional



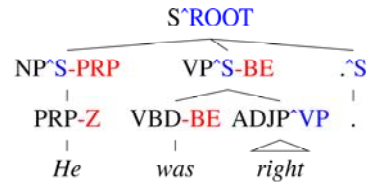
- Advantages:

- Fairly compact grammar
- Linguistic motivations

↓ 72.6 → 86.3

- Disadvantages:

- Performance leveled out
- Manually annotated



## Berkeley Parser: How Does Their Approach Work?

- **State Splitting**

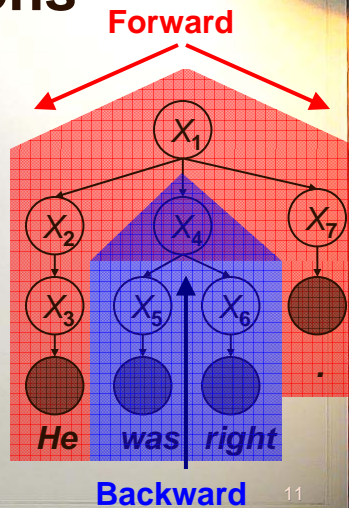
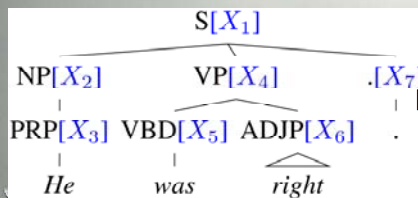
- Want to split complex categories more and simple categories less
- Idea: split everything, and then roll back the splits that are least useful

- **Adaptive State Merging**

- **Parameter Smoothing**

# Learning Latent Annotations

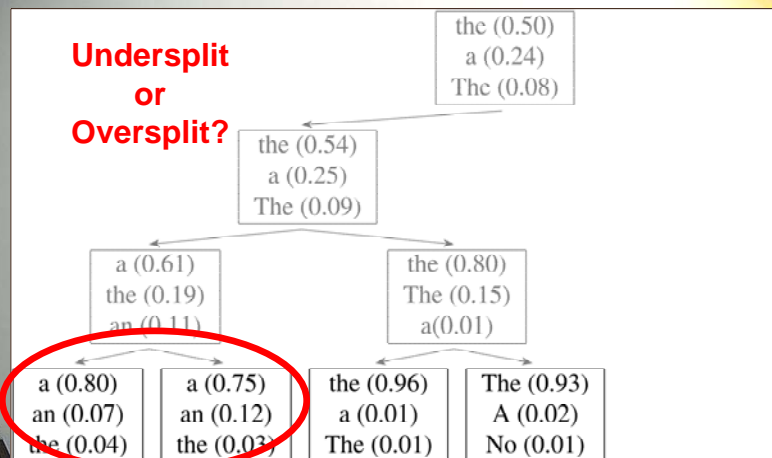
- Brackets are known
- Base categories are known
- Only induce subcategories



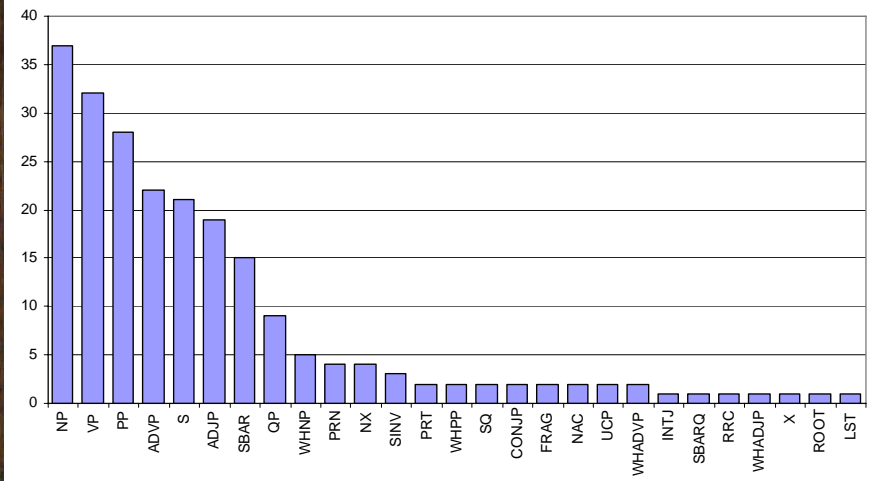
Just like Forward-Backward for HMMs.

# Staged Split-Merge-Smooth

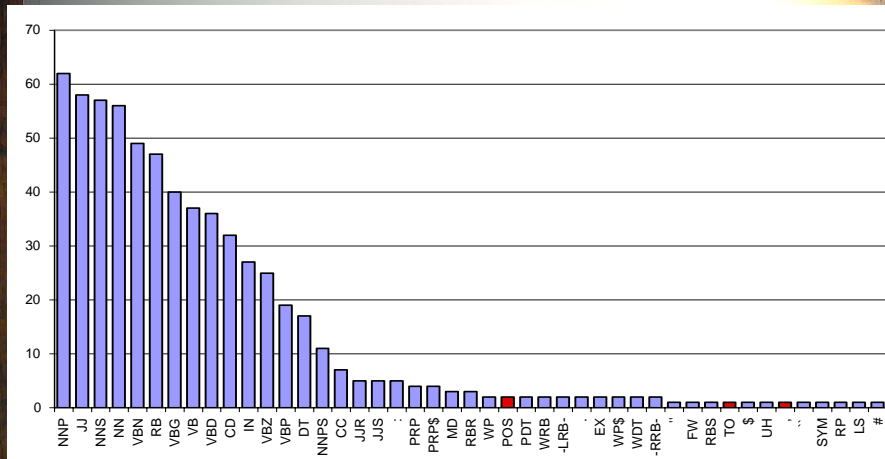
Undersplit  
or  
Oversplit?



## Number of Phrasal Subcategories



## Number of Lexical Subcategories

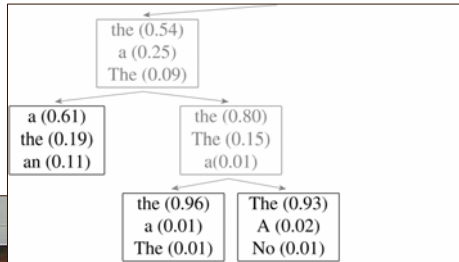


## State Merging

- Evaluate loss in likelihood from removing each split =

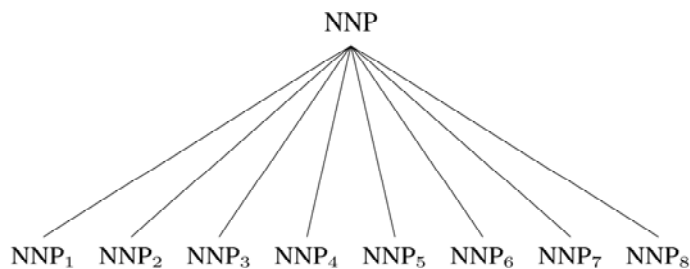
$$\frac{\text{Data likelihood with split reversed}}{\text{Data likelihood with split}}$$

- No loss in accuracy when 50% of the splits are reversed.



## Smoothing

- Heavy splitting can lead to overfitting  
Idea: pool statistics



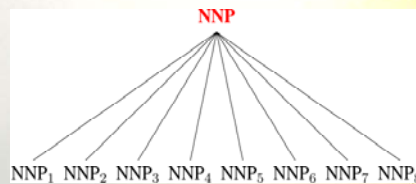
# Linear Smoothing

$$p_x = P(A_x \rightarrow BC)$$



$$p'_x = (1 - \alpha)p_x + \alpha\bar{p}$$

$$\text{where } \bar{p} = \frac{1}{n} \sum_x p_x$$

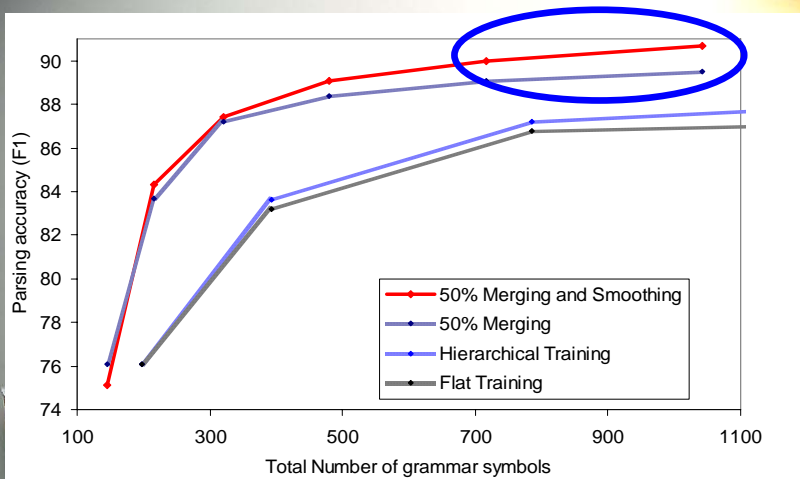


10/11/2007

U Maryland /Purdue

21

# Results in English



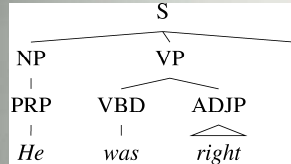
10/11/2007

U Maryland /Purdue

22

# Parse Selection

Parses:



Derivations:



Computing most likely unsplit tree is NP-hard:

- Settle for best derivation.
- Rerank n-best list.
- Use alternative objective function.

# Dynamic Programming

VARIATIONAL: 
$$q(A \rightarrow B C, i, k, j) = \frac{r(A \rightarrow B C, i, k, j)}{\sum_x P_{OUT}(A_x, i, j) P_{IN}(A_x, i, j)}$$

$$T_G = \operatorname{argmax}_T \prod_{e \in T} q(e)$$

[Matsuzaki et al. '05]  
Approximate posterior parse distribution

MAX-RULE-SUM: 
$$q(A \rightarrow B C, i, k, j) = \frac{r(A \rightarrow B C, i, k, j)}{P_{IN}(root, 0, n)}$$

$$T_G = \operatorname{argmax}_T \sum_{e \in T} q(e)$$

à la [Goodman '98]  
Maximize number of expected correct rules

MAX-RULE-PRODUCT: 
$$q(A \rightarrow B C, i, k, j) = \frac{r(A \rightarrow B C, i, k, j)}{P_{IN}(root, 0, n)}$$

$$T_G = \operatorname{argmax}_T \prod_{e \in T} q(e)$$

## WSJ Dynamic Programming Results

Objective	Precision	Recall	F1	Exact
BEST DERIVATION				
Viterbi Derivation	89.6	89.4	89.5	37.4
DYNAMIC PROGRAMMING				
Variational	90.7	90.9	90.8	<b>41.4</b>
Max-Rule-Sum	90.5	<b>91.3</b>	90.9	40.4
Max-Rule-Product	<b>91.2</b>	91.1	<b>91.2</b>	<b>41.4</b>

## Mandarin Parsing Results on CTB

Parsers		Labeled Recall	Labeled Precision	F-measure
Charniak's	Original parser	<b>79.71</b>	<b>81.93</b>	<b>80.81</b>
	Original parser, no unary rules	<b>81.68</b>	<b>82.01</b>	<b>81.85</b>
	Original parser	<b>80.92</b>	<b>84.03</b>	<b>82.44</b>
Latent Annotation PCFG				

## Can This Be Improved?

- OOV handling is primitive
- Single mechanism for improving parse quality: splits and merges based on EM
  - Unary rules
  - Parent Annotations
  - Head Annotations

10/11/2007

U Maryland /Purdue

27

## Character Based Unknown Word Handling

- Originally:
  - Most unknown words are treated the same, except for some special characters.
- Improvement:
  - Use all characters to estimate word probability (based on insights from our tagging experiments)

$$p(\text{UNK} | \text{T})$$

25,301  
二万五千三百零一

$$\sqrt[m]{\prod_{k:p(c_k|\text{T})\neq 0} p(c_k | \text{T})}$$
$$m = |\{k : p(c_k | \text{T}) \neq 0\}|$$

10/11/2007

U Maryland /Purdue

12

# Mandarin Parsing Results on CTB

Parsers		Labeled Recall	Labeled Precision	F-measure
Charniak's	Original parser	79.71	81.93	80.81
	Original parser, no unary rules	81.68	82.01	81.85
Latent Annotation PCFG	Original parser	80.92	84.03	82.44
	Better unknown word handling	81.08	84.57	82.79

10/11/2007

U Maryland /Purdue

13

# Remove Unary Rules

(ROOT  
(SBARQ  
 (SQ (NP (NNS Cats))  
 (VP (VBP do)  
 (VP (WHNP what)  
 (VB eat))))))

→

(ROOT  
 (SQ (NP (NNS Cats))  
 (VP (VBP do)  
 (VP (WHNP what)  
 (VB eat))))))

@SBARQ=sbarq <: SQ  
 excise sbarq sbarq

March 21, 2006

GALE PI Meeting, March, 2006

30

# Mandarin Parsing Results on CTB

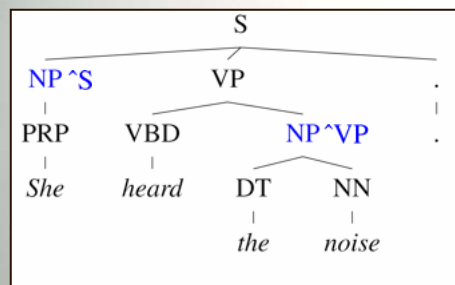
Parsers		Labeled Recall	Labeled Precision	F-measure
Charniak's	Original parser	79.71	81.93	80.81
	Original parser, no unary rules	81.68	82.01	81.85
Latent Annotation PCFG	Original parser	80.92	84.03	82.44
	Better unknown word handling	81.08	84.57	82.79
	Better unknown word handling, no unary rules	83.69	84.16	83.93

10/11/2007

U Maryland /Purdue

13

## Add Parent Annotations



- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]

## Mandarin Parsing Results on CTB

Parsers		Labeled Recall	Labeled Precision	F-measure
Charniak's	Original parser	<b>79.71</b>	<b>81.93</b>	<b>80.81</b>
	Original parser, no unary rules	<b>81.68</b>	<b>82.01</b>	<b>81.85</b>
Latent Annotation PCFG	Original parser	<b>80.92</b>	<b>84.03</b>	<b>82.44</b>
	Better unknown word handling	<b>81.08</b>	<b>84.57</b>	<b>82.79</b>
	Better unknown word handling, no unary rules	<b>83.69</b>	<b>84.16</b>	<b>83.93</b>
	Better unknown word handling, no unary rules, & parent	<b>84.09</b>	<b>84.72</b>	<b>84.41</b>

10/11/2007

U Maryland /Purdue

13

## Use More Data

Parsers		Labeled Recall	Labeled Precision	F-measure
Charniak's	Original parser	<b>79.71</b>	<b>81.93</b>	<b>80.81</b>
	Original parser, no unary rules	<b>81.68</b>	<b>82.01</b>	<b>81.85</b>
Latent Annotation PCFG	Original parser	<b>80.92</b>	<b>84.03</b>	<b>82.44</b>
	Better unknown word handling	<b>81.08</b>	<b>84.57</b>	<b>82.79</b>
	Better unknown word handling, no unary rules	<b>83.69</b>	<b>84.16</b>	<b>83.93</b>
	Better unknown word handling, no unary rules, & parent	<b>84.09</b>	<b>84.72</b>	<b>84.41</b>
	All on CTB 6.0	<b>85.86</b>	<b>87.14</b>	<b>86.49</b>

10/11/2007

U Maryland /Purdue

13

# Adding Heads

$$\text{LA-PCFG} \quad p(\pi, s) = \sum_{t_x} \prod_{c \in \pi} p(r(c) | t_x(c)) \quad \rightarrow \quad \text{Head-Driven LA-PCFG} \quad p(\pi, s) = \sum_{t_x} \prod_{c \in \pi} p(h(c) | t_x(c)) \cdot p(r(c) | t_x(c))$$

- **Head-Driven Latent Annotation PCFG**
  - Lexicalization: expand the work with latent annotated PCFG to incorporate head word information (c: constituent, t(·): constituent type, r(·): rule, h(·): head word, x: latent variable)
  - Using rescoring, we improve the F-score by ~1% using this new model (86.49% to 87.1%)
  - Greater improvement is expected when using dynamic programming approach

# Benefits

- High quality parsing is important for following applications (e.g., mapping to meaning, annotation of source language input to MT)
- We can also use this parser to help build LMs to improve ASR quality and to help rescore MT quality:
  - SuperARV LM
  - Latent Tag LM

# Example SuperARV

C	Lexical Category: Verb		
	Features: {VerbType = past, Voice = active, Inverted = yes, Gapp = none, Mood = whquestion, Agr = all}		
F	Role=G,	Label = VP, pos(x) > mod(x)	(MC=pronoun)
	Role=Need 1,	Label = S, pos(x) < mod(x)	(MC=pronoun)
	Role=Need 2,	Label = S, pos(x) < mod(x)	(MC=verb)
G+N	Role=Need 3,	Label = S, pos(x) = mod(x)	(MC=verb)
	Dependent Position Relation Constraints:		
DC	mod[G] < pos(x) = mod[Need3]		
	< mod[Need1] < mod[Need2]		

MC

10/11/2007

U Maryland /Purdue

37

## Parse for "What did you learn"

1	2	3	4
what	did	you	learn
pronoun case=common behavior=nominal type=interrogative agr=3s	<b>verb</b> subcat=base vtype=past voice=active inverted=yes type=none gapp=yes mood=whquestion agr=all	pronoun case=common behavior=nominal type=personal agr=2s	verb subcat=obj vtype=infinitive voice=active inverted=no gapp=yes mood=whquestion agr=none
G=NP-4	<b>G=VP-1</b> Need1=S-3 Need2=S-4 Need3=S-2	G=NP-2	G=VP-2 Need1=S-4 Need2=S-1 Need3=S-4

ARV (unary constraint) for *VP-1* assigned to *G* for *did*:

cat=verb, subcat=base, vtype=past, voice=active, inverted=yes, type=none, gapp=yes, mood=whquestion, agr=all

rid1=G, label1=vp, (> (pos x)(mod x)), (mod\_cat = pronoun,...)

## Generating CDG Parses For Training LMs

- To create LM training, we must have SuperARV tags for the training data:
  - Automatically derive SuperARV annotations from existing treebanks
  - Parse data using a high quality parser and then derive SuperARV annotations from the parses.
- Build a class language model using SuperARVs as classes to jointly predict a sequence of words and their SuperARVs:

$$P(W_i, T_i), W_i = w_1 \dots w_i, T_i = t_1 \dots t_i$$

$$\Pr(W_N T_N) = \prod_{i=1}^N \Pr(w_i t_i | W_{i-1} T_{i-1}) = \prod_{i=1}^N \Pr(t_i | W_{i-1} T_{i-1}) \cdot \Pr(w_i | W_{i-1} T_i)$$

## ASR LM RT'02 Results

Condition	WER (%) (absolute reduction)	
	Baseline LM	SuperARV LM
2-gram recognition	36.8	36.1 (-0.7%)
4-gram recognition	32.1	30.8 (-1.3%)
Final result (combined system)	25.8	24.2 (-1.6%)

## Mandarin SuperARV LM?

- To port SuperARV LMs to a Mandarin, we need an approach to derive SuperARVs from parse trees for the language as well as language-specific resources (e.g., lexical resources for subcategorization).
- Also, the English model uses lexical feature information that is unavailable in Mandarin
- **Speculation:** the latent tags learned by the parser may be almost as effective as a SuperARV. They are learned automatically from the treebank and can be produced during decoding.

## Preliminary: Factored Latent Variable Language Model

- English factored language models using words, POS, and latent POS tags were evaluated using the entire vocabulary set on the WSJ standard training and test splits:
  - Word trigram: ppl= 246.648
  - Standard POS trigram: ppl= 244.229
  - **Latent POS trigram: ppl= 197.16**
- Mandarin factored language models using words, POS, and latent POS tags were evaluated using the entire vocabulary set on CTB standard training and test splits:
  - Word trigram: ppl= 403.778
  - Standard POS trigram: ppl= 405.487
  - **Latent POS trigram: ppl= 294.198**

## Readings and Useful Links

- Link to the Berkeley parser page (contains links to code and papers): <http://nlp.cs.berkeley.edu/Main.html#Parsing>
- Link to a useful LM resource (man page link points to useful readings): <http://www.speech.sri.com/projects/srilm/>
- Some useful papers:
  - <http://www.eecs.berkeley.edu/~petrov/data/naacl07.pdf>
  - <http://www.eecs.berkeley.edu/~petrov/data/naacl07.pdf>
  - <http://www.eecs.berkeley.edu/~petrov/data/aaai07.pdf>
  - <http://www.cs.berkeley.edu/~klein/papers/unlexicalized-parsing.pdf>
  - <http://acl.ldc.upenn.edu/P/P05/P05-1010.pdf>
  - <ftp://ftp.ecn.purdue.edu/harper/papers/EMNLP163.pdf>