Parsing and Language Modeling

Mary Harper, Zhongqiang Huang, and Denis Filimonov

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, \( t(c) = \text{PP} \))
- \( h(c) \) 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 \( c \)

Mandarin Parsing

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

Example of Unary Rule Removal, Tsurgeon Style

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

@SBARQ=sbarq <: SQ
excise sbarq sbarq

(Root
  (SQ (NP (NNS Cats))
   (VP (VBP do)
    (VP (WHNP what)
       (VB eat))))
Mandarin Parsing

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

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]

- 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

• 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?

He was right.
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

\[ p_x = P(A_x \rightarrow BC') \]

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

where \( \bar{p} = \frac{1}{n} \sum_x p_x \)

Results in English

![Graph showing parsing accuracy (F1) vs. total number of grammar symbols for different training methods: 50% Merging and Smoothing, 50% Merging, Hierarchical Training, and Flat Training. The graph demonstrates the progression of parsing accuracy as the number of grammar symbols increases.]
Parse Selection

Computing most likely unsplit tree is NP-hard:
- Settle for best derivation.
- Rerank n-best list.
- Use alternative objective function.

Dynamic Programming

**Variational:**

\[ q(A \rightarrow B C, i, k, j) = \frac{r(A \rightarrow B C, i, k, j)}{\sum_{x} P_{\text{in}}(A_{x}, i, j) P_{\text{out}}(A_{x}, i, j)} \]

\[ T_{G} = \max_{T} \prod_{c \in T} q(c) \]

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

**Max-Rule-Sum:**

\[ q(A \rightarrow B C, i, k, j) = \frac{r(A \rightarrow B C, i, k, j)}{P_{\text{in}}(\text{root}, 0, n)} \]

\[ T_{G} = \max_{T} \sum_{c \in T} q(c) \]

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

**Max-Rule-Product:**

\[ q(A \rightarrow B C, i, k, j) = \frac{r(A \rightarrow B C, i, k, j)}{P_{\text{in}}(\text{root}, 0, n)} \]

\[ T_{G} = \max_{T} \prod_{c \in T} q(c) \]
### WSJ Dynamic Programming Results

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

### Mandarin Parsing Results on CTB

<table>
<thead>
<tr>
<th>Parsers</th>
<th>Labeled Recall</th>
<th>Labeled Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak's</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser</td>
<td>79.71</td>
<td>81.93</td>
<td>80.81</td>
</tr>
<tr>
<td>Original parser, no unary rules</td>
<td><strong>81.68</strong></td>
<td><strong>82.01</strong></td>
<td><strong>81.85</strong></td>
</tr>
<tr>
<td>Latent Annotation PCFG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser</td>
<td><strong>80.92</strong></td>
<td><strong>84.03</strong></td>
<td><strong>82.44</strong></td>
</tr>
</tbody>
</table>
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

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)
Mandarin Parsing
Results on CTB

<table>
<thead>
<tr>
<th>Parsers</th>
<th>Labeled Recall</th>
<th>Labeled Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak’s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser</td>
<td>79.71</td>
<td>81.93</td>
<td>80.81</td>
</tr>
<tr>
<td>Original parser, no unary rules</td>
<td>81.68</td>
<td>82.01</td>
<td>81.85</td>
</tr>
<tr>
<td>Latent Annotation PCFG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser</td>
<td>80.92</td>
<td>84.03</td>
<td>82.44</td>
</tr>
<tr>
<td>Better unknown word handling</td>
<td>81.08</td>
<td>84.57</td>
<td>82.79</td>
</tr>
</tbody>
</table>

Remove Unary Rules

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

(Sbarq=sbarq <: SQ
  excise sbarq sbarq)
Mandarin Parsing Results on CTB

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

Add Parent Annotations

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]
## Mandarin Parsing Results on CTB

<table>
<thead>
<tr>
<th>Parsers</th>
<th>Labeled Recall</th>
<th>Labeled Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Charniak’s</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser</td>
<td>79.71</td>
<td>81.93</td>
<td>80.81</td>
</tr>
<tr>
<td>Original parser, no unary rules</td>
<td>81.68</td>
<td>82.01</td>
<td>81.85</td>
</tr>
<tr>
<td><strong>Latent Annotation PCFG</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser (w/ new word handling)</td>
<td>80.92</td>
<td>84.03</td>
<td>82.44</td>
</tr>
<tr>
<td>Better unknown word handling, no unary rules</td>
<td>81.08</td>
<td>84.57</td>
<td>82.79</td>
</tr>
<tr>
<td>Better unknown word handling, new word handling, no unary rules</td>
<td>83.69</td>
<td>84.16</td>
<td>83.93</td>
</tr>
<tr>
<td>Better unknown word handling, new word handling, no unary rules, &amp; parent</td>
<td>84.09</td>
<td>84.72</td>
<td>84.41</td>
</tr>
</tbody>
</table>

## Use More Data

<table>
<thead>
<tr>
<th>Parsers</th>
<th>Labeled Recall</th>
<th>Labeled Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Charniak’s</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser</td>
<td>79.71</td>
<td>81.93</td>
<td>80.81</td>
</tr>
<tr>
<td>Original parser, no unary rules</td>
<td>81.68</td>
<td>82.01</td>
<td>81.85</td>
</tr>
<tr>
<td><strong>Latent Annotation PCFG</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original parser (w/ new word handling)</td>
<td>80.92</td>
<td>84.03</td>
<td>82.44</td>
</tr>
<tr>
<td>Better unknown word handling, no unary rules</td>
<td>81.08</td>
<td>84.57</td>
<td>82.79</td>
</tr>
<tr>
<td>Better unknown word handling, new word handling, no unary rules</td>
<td>83.69</td>
<td>84.16</td>
<td>83.93</td>
</tr>
<tr>
<td>Better unknown word handling, new word handling, no unary rules, &amp; parent</td>
<td>84.09</td>
<td>84.72</td>
<td>84.41</td>
</tr>
<tr>
<td>All on CTB 6.0</td>
<td>85.86</td>
<td>87.14</td>
<td>86.49</td>
</tr>
</tbody>
</table>
Adding Heads

LA-PCFG
\[ p(\pi, s) = \sum_{x \in \mathbb{X}} \prod_{c \in \pi} p(c \mid t_r(c)) \]

Head-Driven LA-PCFG
\[ p(\pi, s) = \sum_{x \in \mathbb{X}} \prod_{c \in \pi} p(h(c) \mid t_r(c)) \cdot p(c \mid t_r(c)) \]

• 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
Parse for “What did you learn”

<table>
<thead>
<tr>
<th></th>
<th>1 what</th>
<th>2 did</th>
<th>3 you</th>
<th>4 learn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pronoun</td>
<td>verb</td>
<td>pronoun</td>
<td>verb</td>
</tr>
<tr>
<td></td>
<td>case=common</td>
<td>subcat=base</td>
<td>case=common</td>
<td>subcat=base</td>
</tr>
<tr>
<td></td>
<td>behavior=nominal</td>
<td>vtype=past</td>
<td>behavior=nominal</td>
<td>vtype=infinitive</td>
</tr>
<tr>
<td></td>
<td>type=interrogative</td>
<td>voice=active</td>
<td>type=nominal</td>
<td>voice=active</td>
</tr>
<tr>
<td></td>
<td>agr=3s</td>
<td>inverted=yes</td>
<td>agr=2s</td>
<td>inverted=no</td>
</tr>
<tr>
<td></td>
<td>G=NP-4</td>
<td>gapp=yes</td>
<td>G=VP-1</td>
<td>mood=whquestion</td>
</tr>
<tr>
<td></td>
<td>Need1=S-3</td>
<td>mood=whquestion</td>
<td>Need1=S-3</td>
<td>mood=whquestion</td>
</tr>
<tr>
<td></td>
<td>Need2=S-4</td>
<td>agr=all</td>
<td>Need2=S-4</td>
<td>agr=all</td>
</tr>
<tr>
<td></td>
<td>Need3=S-2</td>
<td></td>
<td>Need3=S-4</td>
<td></td>
</tr>
</tbody>
</table>

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:

\[
P(W_i, T_i), W_i = w_1 \ldots w_i, T_i = t_1 \ldots t_i
\]

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

ASR LM RT’02 Results

<table>
<thead>
<tr>
<th>Condition</th>
<th>WER (%)</th>
<th>(absolute reduction)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline LM</td>
<td>SuperARV LM</td>
</tr>
<tr>
<td>2-gram recognition</td>
<td>36.8</td>
<td>36.1 (-0.7%)</td>
</tr>
<tr>
<td>4-gram recognition</td>
<td>32.1</td>
<td>30.8 (-1.3%)</td>
</tr>
<tr>
<td>Final result (combined system)</td>
<td>25.8</td>
<td>24.2 (-1.6%)</td>
</tr>
</tbody>
</table>
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](http://nlp.cs.berkeley.edu/Main.html#Parsing)
- 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/naacl07.pdf](http://www.eecs.berkeley.edu/~petrov/data/naacl07.pdf)
  - [http://www.eecs.berkeley.edu/~petrov/data/aaai07.pdf](http://www.eecs.berkeley.edu/~petrov/data/aaai07.pdf)
  - [http://acl.ldc.upenn.edu/P/P05/P05-1010.pdf](http://acl.ldc.upenn.edu/P/P05/P05-1010.pdf)