Word Ordering Without Syntax

Allen Schmaltz  Alexander M. Rush  Stuart M. Shieber

Harvard University

EMNLP, 2016
Outline

1. Task: Word Ordering, or Linearization
2. Models
3. Experiments
4. Results
Task: Word Ordering, or Linearization

Word Ordering

- Task: Recover the original order of a shuffled sentence

Given a bag of words

\{ the, ., Investors, move, welcomed \}

Goal is to recover the original sentence

Investors welcomed the move.
Task: Word Ordering, or Linearization

**Word Ordering**

- Task: Recover the original order of a shuffled sentence

**Variant:** Shuffle, retaining base noun phrases (BNPs)

\{ the move, ., Investors, welcomed \}

\[\downarrow\]

Goal is to recover the original sentence

Investors welcomed the move .
Word Ordering

Early work


*The order of words in sentences reflects a number of constraints. . . Syntactic structure, selective restrictions, subcategorization, and discourse considerations are among the many factors which join together to fix the order in which words occur. . . [T]here is an abstract structure which underlies the surface strings and it is this structure which provides a more insightful basis for understanding the constraints on word order. . . . It is, therefore, an interesting question to ask whether a network can learn any aspects of that underlying abstract structure.*

The word ordering task also appears in Brown et al. (1990) and Brew (1992).
Word Ordering, Recent Work (Zhang and Clark, 2011; Liu et al., 2015; Liu and Zhang, 2015; Zhang and Clark, 2015)

- Liu et al. (2015) (known as ZGEN)
  - State of art on PTB
  - Uses a transition-based parser with beam search to construct a sentence and a parse tree

  - Claims syntactic models yield improvements over pure surface n-gram models
    - Particularly on longer sentences
    - Even when syntactic trees used in training are of low quality
Revisiting comparison between syntactic & surface-level models

Simple takeaway:

- **Prior work:** Jointly recovering explicit syntactic structure is important, or even required, for effectively recovering word order
- **We find:** Surface-level language models with a simple heuristic give much stronger results on this task
Models - Inference

- Scoring function:

\[ f(x, y) = \sum_{n=1}^{N} \log p(x_y(n) \mid x_y(1), \ldots, x_y(n-1)) \]

\[ y^* = \arg \max_{y \in \mathcal{Y}} f(x, y) \]

- Beam search: Maintain multiple beams, as in stack decoding for phrase-based MT

- Include an estimate of future cost in order to improve search accuracy: Unigram cost of uncovered tokens in the bag
Beam Search ($K = 3$): Unigram Future Cost Example

Shuffled bag
{ the, ., Investors, move, welcomed }

- Timestep 1:
  - $\text{score(Investors)} = \log p(\text{Investors} \mid \text{START}) + \log p(\text{the}) + \log p(.) + \log p(\text{move}) + \log p(\text{welcomed})$
Beam Search \((K = 3)\): Unigram Future Cost Example

Shuffled bag

\{ the, ., Investors, move, welcomed \}

- Timestep 2
Beam Search \((K = 3)\): Unigram Future Cost Example

Shuffled bag

\{ the, ., Investors, move, welcomed \}

- **Timestep 3:**
  - \(\text{score}(\text{Investors welcomed the}) = \log p(\text{Investors} \mid \text{START}) + \log p(\text{welcomed} \mid \text{START, Investors}) + \log p(\text{the} \mid \text{START, Investors, welcomed}) + \log p(.) + \log p(\text{move})\)
Experiments

Data, matches past work:
- PTB, standard splits, Liu et al. (2015)
- PTB + Gigaword sample (gw), Liu and Zhang (2015)
- Words and Words+BNPs tasks

Baseline: Syntactic ZGen model (Liu et al., 2015)
- With/without POS tags

Our LM models: NGRAM and LSTM
- With/without unigram future costs
- Varying beam size (64, 512)
### Test Set Performance (BLEU), **Words** task

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ZGen-64</strong></td>
<td>30.9</td>
</tr>
<tr>
<td>NGram-64 (no future cost)</td>
<td>32.0</td>
</tr>
<tr>
<td>NGram-64</td>
<td>37.0</td>
</tr>
<tr>
<td>NGram-512</td>
<td>38.6</td>
</tr>
<tr>
<td>LSTM-64</td>
<td>40.5</td>
</tr>
<tr>
<td>LSTM-512</td>
<td>42.7</td>
</tr>
</tbody>
</table>
### Test Set Performance (BLEU), *Words+BNPs* task

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZGen-64</td>
<td>49.4</td>
</tr>
<tr>
<td>ZGen-64+pos</td>
<td>50.8</td>
</tr>
<tr>
<td>NGram-64 (no future cost)</td>
<td>51.3</td>
</tr>
<tr>
<td>NGram-64</td>
<td>54.3</td>
</tr>
<tr>
<td>NGram-512</td>
<td>55.6</td>
</tr>
<tr>
<td>LSTM-64</td>
<td>60.9</td>
</tr>
<tr>
<td>LSTM-512</td>
<td>63.2</td>
</tr>
<tr>
<td>ZGen-64+lm+gw+pos</td>
<td>52.4</td>
</tr>
<tr>
<td>LSTM-64+gw</td>
<td>63.1</td>
</tr>
<tr>
<td>LSTM-512+gw</td>
<td>65.8</td>
</tr>
</tbody>
</table>
Performance by sentence length

Figure: Performance on PTB validation by length (\textsc{Words+BNPs} models)
## Additional Comparisons

<table>
<thead>
<tr>
<th></th>
<th>BNP</th>
<th>g</th>
<th>GW</th>
<th>1</th>
<th>10</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>41.7</td>
<td>53.6</td>
<td>58.0</td>
<td>59.1</td>
<td>60.0</td>
<td>60.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>47.6</td>
<td>59.4</td>
<td>62.2</td>
<td>62.9</td>
<td>63.6</td>
<td>64.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>48.4</td>
<td>60.1</td>
<td>64.2</td>
<td>64.9</td>
<td>65.6</td>
<td>66.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>15.4</td>
<td>26.8</td>
<td>33.8</td>
<td>35.3</td>
<td>36.5</td>
<td>38.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>25.0</td>
<td>36.8</td>
<td>40.7</td>
<td>41.7</td>
<td>42.0</td>
<td>42.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>23.8</td>
<td>35.5</td>
<td>40.7</td>
<td>41.7</td>
<td>42.9</td>
<td>43.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NGram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>40.6</td>
<td>49.7</td>
<td>52.6</td>
<td>53.2</td>
<td>54.0</td>
<td>54.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>45.7</td>
<td>53.6</td>
<td>55.6</td>
<td>56.2</td>
<td>56.6</td>
<td>56.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>14.6</td>
<td>27.1</td>
<td>32.6</td>
<td>33.8</td>
<td>35.1</td>
<td>35.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>27.1</td>
<td>34.6</td>
<td>37.5</td>
<td>38.1</td>
<td>38.4</td>
<td>38.7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

- **Strong surface-level language models** recover word order more accurately than the models trained with explicit syntactic annotations.
- **LSTM LMs** with a simple **future cost heuristic** are particularly effective.
Conclusion

- **Strong surface-level language models** recover word order more accurately than the models trained with explicit syntactic annotations.
- **LSTM LMs** with a simple **future cost heuristic** are particularly effective.

**Implications**
- Begin to question the utility of costly syntactic annotations in generation models (e.g., grammar correction).
- Part of larger discussion as to whether LSTMs, themselves, are capturing syntactic phenomena.
Replication code is available at
https://github.com/allenschmaltz/word_ordering