A Comparison of Neural Models for Word Ordering

Eva Hasler\textsuperscript{1,2}, Felix Stahlberg\textsuperscript{1}, Marcus Tomalin\textsuperscript{1}, Adrià de Gispert\textsuperscript{1,2}, Bill Byrne\textsuperscript{1,2}

\textsuperscript{1}Department of Engineering, University of Cambridge, UK
\textsuperscript{2}SDL Research, Cambridge, UK
{ech57,fs439,mt126,ad465,wjb31}@cam.ac.uk
{ehasler,agispert,bbyrne}@sdl.com

Abstract

We compare several language models for the word-ordering task and propose a new bag-to-sequence neural model based on attention-based sequence-to-sequence models. We evaluate the model on a large German WMT data set where it significantly outperforms existing models. We also describe a novel search strategy for LM-based word ordering and report results on the English Penn Treebank. Our best model setup outperforms prior work both in terms of speed and quality.

1 Introduction

Finding the best permutation of a multi-set of words is a non-trivial task due to linguistic aspects such as “syntactic structure, selective restrictions, subcategorization, and discourse considerations” (Elman, 1990). This makes the word-ordering task useful for studying and comparing different kinds of models that produce text in tasks such as general natural language generation (Reiter and Dale, 1997), image caption generation (Xu et al., 2015), or machine translation (Bahdanau et al., 2015). Since plausible word order is an essential criterion of output fluency for all of these tasks, progress on the word-ordering problem is likely to have a positive impact on these tasks as well. Word ordering has often been addressed as syntactic linearization which is a strategy that involves using syntactic structures or part-of-speech and dependency labels (Zhang and Clark, 2011; Zhang et al., 2012; Zhang and Clark, 2015; Liu et al., 2015; Puduppully et al., 2016). It has also been addressed as LM-based linearization which relies solely on language models and obtains better scores (de Gispert et al., 2014; Schmaltz et al., 2016). Recently, Schmaltz et al. (2016) showed that recurrent neural network language models (Mikolov et al., 2010, RNNLMs) with long short-term memory (Hochreiter and Schmidhuber, 1997) LSTM cells are very effective for word ordering even without any explicit syntactic information.

We continue this line of work and make the following contributions. We compare several language models on the word-ordering task and propose a bag-to-sequence neural architecture that equips an LSTM decoder with explicit context of the bag-of-words (BOW) to be ordered. This model performs particularly strongly on WMT data and is complementary to an RNNLM: combining both yields large BLEU gains even for small beam sizes. We also propose a novel search strategy which outperforms a previous heuristic. Both techniques together surpass prior work on the Penn Treebank at $\sim4$x the speed.

2 Bag-to-Sequence Modeling with Attentional Neural Networks

Given the BOW \{at, bottom, heap, now, of, the, the, we, ’re, ,\}, a word-ordering model may generate an output string $w = \text{“now we ’re at the bottom of the heap .\text{“}}$. We can use an RNNLM (Mikolov et al., 2010) to assign it a probability $P(w)$ by decomposing into conditionals:

$$P(w_T^T) = \prod_{t=1}^T P(w_t|w_{t-1}^t)$$

Since we have access to the input Bows, we extend the model representation by providing the network additionally with the BOW to be ordered, thereby allowing it to focus explicitly on all tokens it generates in the output during decoding. Thus, instead of mod-
valid permutations
model outputs are put bag at each time step, thereby ensuring that all output vocabulary to the remaining tokens in the in-
our beam decoder by limiting its available constrain to appear in the output. Therefore, we explicitly architecture does not prevent words outside the B
OW ⟨sequence trained to recover word order in a sentence by using translation between a source sentence \( x = x_1^T \) and a target sentence \( y = y_1^T \). Fig. 1a illustrates how seq2seq generates the \( j \)-th target token \( y_j \) using the decoder state \( s_j \) and the context vector \( c_j \). The context vector is the weighted sum of source side annotations \( h_i \) which encode sequence information.

To modify seq2seq for problems with unordered input, we make the encoder architecture order-invariant by replacing the recurrent layer with non-recurrent transformations of the word embeddings, as indicated by the missing arrows between source positions in Fig. 1b. For convenience, we formalize Bow(\( w \)) as sequence \( \langle \tilde{w}_1, \ldots, \tilde{w}_T \rangle \) in which words are sorted, e.g. alphabetically, so that we can refer to the \( t \)-th word in the Bow. The model can be trained to recover word order in a sentence by using Bow(\( w \)) = \( \langle \tilde{w}_1, \ldots, \tilde{w}_T \rangle \) as input and the original sequence \( \langle w_1, \ldots, w_T \rangle \) as target. This network architecture does not prevent words outside the Bow to appear in the output. Therefore, we explicitly constrain our beam decoder by limiting its available output vocabulary to the remaining tokens in the input bag at each time step, thereby ensuring that all model outputs are valid permutations of the input.

3 Search

Beam search is a popular decoding algorithm for neural sequence models (Sutskever et al., 2014; Bahdanau et al., 2015). However, standard beam search suffers from search errors when applied to word ordering and Schmaltz et al. (2016) reported that gains often do not saturate even with a large beam of 512. They suggested adding external unigram probabilities of the remaining words in the Bow as future cost estimates to the beam-search scoring function and reported large gains for an n-gram LM and RNNLM. We re-implement this future cost heuristic, \( f(\cdot) \), and further propose a new search heuristic, \( g(\cdot) \), which collects internal unigram statistics during decoding. We keep hypotheses in the beam if their score is close to a theoretical upper bound, the product of the best word probabilities given any history within the explored search space. For each word \( \tilde{w} \in \text{Bow}(w) \) we maintain a heuristic score estimate \( \hat{P}(\tilde{w}) \) which we initialize to 0. Each time the search algorithm visits a new context, we update the estimates such that \( \hat{P}(\tilde{w}) \) is the current best score for \( \tilde{w} \):

\[
\hat{P}(\tilde{w}) = \max_{c \in C_t} P(\tilde{w}|c, \text{Bow}(w))
\]

where \( C_t \) is the set of contexts (i.e. ordered prefixes in the form of \( w_1^T \)) explored by beam search so far. Thus, instead of computing a future cost, we compare the actual score of a partial hypothesis with the product of heuristic estimates of its words. This is especially useful for model combinations since all models are taken into account. We also implement hypothesis recombination to further reduce the number of search errors. More formally, at each time step \( t \) our beam search keeps the \( n \) best hypotheses according to scoring function \( S(\cdot) \) using partial model score \( s(\cdot) \) and estimates \( g(\cdot) \):

\[
S(w_1^T) = s(w_1^T) - g(w_1^T)
\]

\[
s(w_1^T) = \log P(w_1^T|\text{Bow}(w))
\]

\[
g(w_1^T) = \sum_{w' \in w_1^T} \log \hat{P}(w')
\]

4 Experimental Setup

We evaluate using data from the English-German news translation task (Bojar et al., 2015, WMT) and using the English Penn Treebank data (Marcus et al., 1993, PTB). Since additional knowledge sources
are often available in practice, such as access to the source sentence in a translation scenario, we also report on bilingual experiments for the WMT task.

4.1 Data and evaluation

The WMT parallel training data includes Europarl v7, Common Crawl, and News Commentary v10. We use news-test2013 for tuning model combinations and news-test2015 for testing. All monolingual models for the WMT task were trained on the German news2015 corpus (∼51.3M sentences). For PTB, we use preprocessed data by Schmaltz et al. (2016) for a fair comparison (∼40k sentences for training). We evaluate using the multi-bleu.perl script for WMT and mteval-v13.pl for PTB.

4.2 Model settings

For WMT, the bag2seq parameter settings follow the recent NMT systems trained on WMT data. We use a 50k vocabulary, 620 dimensional word embeddings and 1000 hidden units in the decoder LSTM cells. On the encoder side, the input tokens are embedded to form annotations of the same size as the hidden units in the decoder. The RNNLM is based on the “large” setup of Zaremba et al. (2014) which uses an LSTM. NPLM, a 5-gram neural feedforward language model, was trained for 10 epochs with a vocabulary size of 100k, 150 input and output units, 750 hidden units and 100 noise samples (Vaswani et al., 2013). The n-gram language model is a 5-gram model estimated with SRILM (Kneser and Ney, 1995). For the bilingual setting, we implemented a seq2seq NMT system following Bahdanau et al. (2015) using a beam size of 12 in line with recent NMT systems for WMT (Sennrich et al., 2016). RNNLM, bag2seq and seq2seq were implemented using TensorFlow (Abadi et al., 2015)[1] and we used sgnmt for beam decoding[2].

Following Schmaltz et al. (2016), our neural models for PTB have a vocabulary of 16,161 incl. two different unk tokens and the RNNLM is based on the “medium” setup of Zaremba et al. (2014). bag2seq uses 300 dimensional word embeddings and 500 hidden units in the decoder LSTM. We also compare to GYRO (de Gispert et al., 2014) which explicitly targets the word-ordering problem. We extracted 1-gram to 5-gram phrase rules from the PTB training data and used an n-gram LM for decoding. For model combinations, we combine the predictive distributions in a log-linear model and tune the weights by optimizing BLEU on the validation set with the BOBYQA algorithm (Powell, 2009).

5 Results

5.1 Word Ordering on WMT data

The top of Tab. 1 shows that bag2seq outperforms all other language models by up to 4.2 BLEU on ordering German (bold numbers highlight its improvements). This suggests that explicitly presenting all available tokens to the decoder during search enables it to make better word order choices. A combination of RNNLM, NPLM and n-gram LM yields a higher score than the individual models, but further adding bag2seq yields a large gain of 4.5 BLEU confirming its suitability for the word-ordering task.

In the bilingual setting in the bottom of Tab. 1, the seq2seq model is given English input text and the beam decoder is constrained to generate permutations of German BOWs. This is effectively a translation task with knowledge of the target BOWs and seq2seq provides a strong baseline since it uses source sequence information. Still, adding bag2seq yields a 2.9 BLEU gain and adding it to the combination of all other models still improves by 1.8 BLEU. This suggests that it could also help for machine translation rescoring by selecting hypotheses that constitute good word orderings.

5.2 Word Ordering on the Penn Treebank

Tab. 2 shows the performance of different models and search heuristics on the Penn Treebank: using

| RNNLM | NPLM | n-gram | bag2seq | seq2seq | BLEU |
|-------|------|--------|--------|--------|------|
| ✔     | ✔    | ✔      | ✔      | ✔      | 29.4 |
|       |       |        |        |        | 30.3 |
|       |       |        |        |        | 32.5 |
| ✔     | ✔    | ✔      | ✔      | ✔      | 33.6 |
|       |       |        |        |        | 34.9 |
| ✔     | ✔    | ✔      | ✔      | ✔      | 39.4 |

Table 1: German word ordering on news-test2015 with beam=12, single models/combinations. Monolingual models use heuristic $f(\cdot)$, bag2seq as a single model and bilingual models use no heuristic.
Table 2: BLEU scores for PTB word-ordering task (test). NGRAM-512 and LSTM-512 are quoted from Schmaltz et al. (2016).

| Model          | none | $f(\cdot)$ | $g(\cdot)$ |
|---------------|------|------------|------------|
| Previous work | beam=512 |
| GYRC         | 42.2 | -          | -          |
| NGRAM-512    | -    | 38.6       | -          |
| LSTM-512     | -    | 42.7       | -          |
| This work    | beam=512 |
| n-gram       | 35.7 | 38.6       | **38.9**   |
| RNNLM        | 38.6 | 43.2       | **44.2**   |
| bag2seq      | **37.1** | 33.6 | **37.1**   |

Table 3: BLEU scores for PTB word-ordering task for different search heuristics and beam sizes (test).

| Model          | none | $f(\cdot)$ | $g(\cdot)$ |
|---------------|------|------------|------------|
| beam=5       |      |            |            |
| n-gram       | 23.3 | **30.1**   | 26.5       |
| RNNLM        | 24.5 | **33.6**   | 29.7       |
| bag2seq      | **33.4** | 27.0 | 31.7       |
| RNNLM-ensemble |       |            |            |
| bag2seq-ensemble |     |            |            |
| RNNLM+bag2seq | 25.5 | **34.2**   | 30.6       |
| RNNLM+bag2seq | 34.8 | **35.1**   | 32.8       |
| RNNLM+bag2seq | 35.7 | **37.9**   | 34.4       |

Figure 2: Decoding time in relation to beam size for PTB word ordering task (test).

6 Conclusion

We have compared various models for the word-ordering task and proposed a new model architecture inspired by attention-based sequence-to-sequence models that helps performance for both German and English tasks. We have also proposed a novel search heuristic and found that using a model combination together with this heuristic and a modest beam size provides a good trade-off between speed and quality and outperforms prior work on the PTB task.
Acknowledgments

This work was partially supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC grant EP/L027623/1). We thank the authors of Schmaltz et al. (2016) for sharing their preprocessed data and helping to reproduce their results.

References

[Abadi et al.2015] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. 1. Software available from http://www.tensorflow.org/.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR.

[Bojar et al.2015] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 1–46. Association for Computational Linguistics.

[de Gispert et al.2014] Adrià de Gispert, Marcus Tomalin, and W Byrne. 2014. Word ordering with phrase-based grammars. In EACL, pages 259–268.

[Elman1990] Jeffrey L Elman. 1990. Finding structure in time. Cognitive science, 14(2):179–211.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

[Kneser and Ney1995] Reinhard Kneser and Hermann Ney. 1995. Improved back-off for m-gram language modeling. In ICASSP, volume 1, pages 181–184.

[Liu et al.2015] Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In NAACL.

[Marcus et al.1993] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. Computational linguistics, 19(2):313–330.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In Interspeech, volume 2, page 3.

[Powell2009] Michael JD Powell. 2009. The BOBYQA algorithm for bound constrained optimization without derivatives. Cambridge NA Report NA2009/06, University of Cambridge, Cambridge.

[Puduppully et al.2016] Ratiash Puduppully, Yue Zhang, and Manish Shrivastava. 2016. Transition-based syntactic linearization with lookahead features. In Proceedings of NAACL-HLT, pages 488–493.

[Reiter and Dale1997] Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. Natural Language Engineering, 3(01):57–87.

[Schmaltz et al.2016] Allen Schmaltz, Alexander M Rush, and Stuart M Shieber. 2016. Word ordering without syntax. In EMNLP.

[Sennrich et al.2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In Proceedings of the First Conference on Machine Translation, pages 371–376.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of NIPS.

[Vaswani et al.2013] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In EMNLP.

[Xu et al.2015] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. arXiv preprint arXiv:1502.03044.

[Zaremba et al.2014] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329.

[Zhang and Clark2011] Yue Zhang and Stephen Clark. 2011. Syntax-based grammaticality improvement using CCG and guided search. In EMNLP, pages 1147–1157.

[Zhang and Clark2015] Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. Computational Linguistics, 41(3):503–538.

[Zhang et al.2012] Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-based word ordering incorporating a large-scale language model. In EACL, pages 736–746.