Fast and Accurate Preordering for SMT using Neural Networks

Adrià de Gispert **Gonzalo Iglesias** **Bill Byrne**
SDL Research
East Road, Cambridge CB1 1BH, U.K.
{agispert|giglesias|bbyrne}@sdl.com

Abstract

We propose the use of neural networks to model source-side preordering for faster and better statistical machine translation. The neural network trains a logistic regression model to predict whether two sibling nodes of the source-side parse tree should be swapped in order to obtain a more monotonic parallel corpus, based on samples extracted from the word-aligned parallel corpus. For multiple language pairs and domains, we show that this yields the best reordering performance against other state-of-the-art techniques, resulting in improved translation quality and very fast decoding.

1 Introduction

Preordering is a pre-processing task in translation that aims to reorder the source sentence so that it best resembles the order of the target sentence. If done correctly, it has a doubly beneficial effect: it allows a better estimation of word alignment and translation models which results in higher translation quality for distant language pairs, *and* it speeds up decoding enormously as less word movement is required.

Preordering schemes can be automatically learnt from source-side parsed, word-aligned parallel corpora. Recently Jehl et al (2014) described a scheme based on a feature-rich logistic regression model that predicts whether a pair of sibling nodes in the source-side dependency tree need to be permuted. Based on the node-pair swapping probability predictions of this model, a branch-and-bound search returns the best ordering of nodes in the tree.

We propose using a neural network (NN) to estimate this node-swapping probability. We find that this straightforward change to their scheme has multiple advantages:

1. The superior modeling capabilities of NNs achieve better performance at preordering and overall translation quality when using the same set of features.
2. There is no need to manually define which feature combinations are to be considered in training.
3. Preordering is even faster as a result of the previous point. Our results in translating from English to Japanese, Korean, Chinese, Arabic and Hindi support these findings by comparing against two other preordering schemes.

1.1 Related Work

There is a strong research and commercial interest in preordering, as reflected by the extensive previous work on the subject (Collins et al., 2005; Xu et al., 2009; DeNero and Uszkoreit, 2011; Neubig et al., 2012). We are interested in practical, language-independent preordering approaches that rely only on automatic source-language parsers (Genzel, 2010). The most recent work in this area uses large-scale feature-rich discriminative models, effectively treating preordering either as a learning to rank (Yang et al., 2012), multi-classification (Lerner and Petrov, 2013) or logistic regression (Jehl et al., 2014) problem. In this paper we incorporate NNs into the latter approach.

Lately an increasing body of work that uses NNs for various NLP tasks has been published, including language modeling (Bengio et al., 2003), POS tagging (Collobert et al., 2011), or dependency parsing (Chen and Manning, 2014). In translation, NNs have been used for improved word alignment (Yang et al., 2013; Tamura et al., 2014; Songyot and Chiang, 2014), to model reordering under an ITG grammar (Li et al., 2013), and to define additional feature functions to be used in decoding (Sundermeyer et al., 2014; Devlin et al., 2014). End-to-end translation systems based on NNs have also been proposed (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014).

Despite the gains reported, only the approaches that do not dramatically affect decoding times can be directly applied to today's commercial SMT systems. Our paper is a step towards this direction, and to the best of our knowledge, it is the first one to describe the usage of NNs in preordering for SMT.

2 Preordering as node-pair swapping

Jehl et al (2014) describe a preordering scheme based on a logistic regression model that predicts whether a pair of sibling nodes in the source-side dependency tree need to be permuted in order to have a more monotonically-aligned parallel corpus. Their method can be briefly summarised by the pseudocode of Figure 1.

Let N be the set of nodes in the source tree, and let C_n be the set of children nodes of node n. For each node with at least two children, first extract the node features (lines 1-2). Then, for each pair of its children nodes: extract their respective features (lines 4-5), produce all relevant feature combinations (line 6), and store the node-pair swapping probability predicted by a logistic regression model based on all available features (line 7). Once all pair-wise probabilities are stored, search for the best global permutation and sort C_n accordingly (lines 9-10).

As features, Jehl et al (2014) use POS tags and dependency labels, as well as the identity and class of the head word (for the parent node) or the left/rightmost word (for children nodes). These are combined into *conjunctions* of 2 or 3 features to create new features. For logistic regression, they train a L1-regularised linear model using LIBLINEAR (Fan et al., 2008). The training samples are either positive/negative depending on whether swapping the nodes reduces/increases the number of crossed links in the aligned parallel corpus.

```
PREORDERPARSETREE
1  for each node $n \in N, |C_n| > 1$
2    $F \leftarrow$ GETFEATURES($n$)
3    for each pair of nodes $i, j \in C_n, i \neq j$
4      $F \leftarrow F \cup$ GETFEATURES($i$)
5      $F \leftarrow F \cup$ GETFEATURES($j$)
6      $F_c \leftarrow$ FEATURECOMBINATIONS($F$)
7      $p_n(i, j) =$ LOGREGPREDICT($F, F_c$)
8    end for
9    $\pi_n \leftarrow$ SEARCHPERMUTATION($p_n$)
10   SORT($C_n, \pi_n$)
```

Figure 1: Pseudocode for the preordering scheme of Jehl et al (2014)

2.1 Applying Neural Networks

"A (feedforward) neural network is a series of logistic regression models stacked on top of each other, with the final layer being either another logistic regression model or a linear regression model" (Murphy, 2012).

Given this, we propose a straightforward alternative to the above framework: replace the linear logistic regression model by a neural network (NN). This way a superior modeling performance of the node-swapping phenomenon is to be expected. Additionally, feature combination need not be engineered anymore because that is *learnt* by the NN in training (line 6 in Figure 1 is skipped).

Training the neural network requires the same labeled samples that were used by Jehl et al (2014). We use the NPLM toolkit out-of-the-box (Vaswani et al., 2013). The architecture is a feed-forward neural network (Bengio et al., 2003) with four layers. The first layer i contains the input embeddings. The next two hidden layers (h_1, h_2) use rectified linear units; the last one is the softmax layer (o). We did not experiment with deeper NNs.

For our purposes, the input vocabulary of the NN is the set of all possible feature indicator names that are used for preordering[1]. There are no OOVs. Given the sequence of $\sim$ 20 features seen by the

[1]Using a vocabulary of the 5K top-frequency English words, 50 word classes, approximately 40 POS tags and 50 dependency labels, the largest input vocabulary in our experiments is roughly 30,000.

preorderer, the NN is trained to predict whether the nodes should be reordered or not, i.e. $|o| = 2$. For the rest of the layers, we use $|i| = 50$, $|h_1| = 100$, $|h_2| = 50$. We set the learning rate to 1, the mini-batch size to 64 and the number of epochs to 20.

3 Experiments

3.1 Data and setup

We report translation results in English into Japanese, Korean, Chinese, Arabic and Hindi. For each language pair, we use generic parallel data extracted from the web. The number of words is about 100M for Japanese and Korean, 300M for Chinese, 200M for Arabic and 9M for Hindi.

We use two types of dev/test sets: in-domain and mix-domain. The in-domain sets have 2K sentences each and were created by randomly extracting parallel sentences from the corpus, ensuring no repetitions remained. The mix-domain sets have about 1K sentences each and were created to evenly represent 10 different domains, including world news, chat/SMS, health, sport, science, business, and others.

Additionally, we report results on the English-to-Hindi WMT 2014 shared task (Bojar et al., 2014a) using the data provided[2]. The dev and test sets have 520 and 2507 sentences each. All dev and test sets have one single reference. We use SVM-Tool (Giménez and Màrquez, 2004) for POS Tagging, and MaltParser (Nivre et al., 2007) for dependency parsing.

3.2 Intrinsic reordering evaluation

We evaluate the intrinsic preordering task on a random 5K-sentence subset of the training data which is excluded from model estimation. We report the normalized crossing score c/s, where c is the number of crossing links (Genzel, 2010; Yang et al., 2012) in the aligned parallel corpus, and s is the number of source (e.g. English) words. Ideally we would like this metric to be zero, meaning a completely monotonic parallel corpus[3]; the more monotonic the corpus, the better the translation models will be and the faster decoding will run as less distortion will be needed.

[2]HindEndCorp v0.5 (Bojar et al., 2014b)

[3]However this may not be achievable given the alignment links and parse tree available. In this approach, only permutations of sibling nodes in the single source parse tree are permitted.

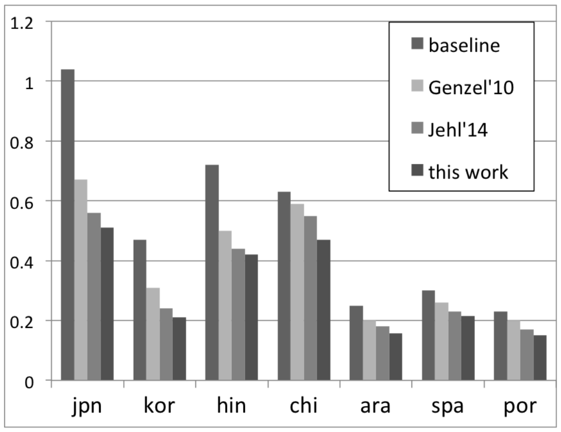

Figure 2: Normalized crossing score for English into Japanese, Korean, Hindi, Chinese, Arabic, Spanish and Portuguese.

Normalizing over the number of source words allows us to compare this metric across language pairs, and so the potential impact of preordering in translation performance becomes apparent. See Figure 2 for results across several language pairs. In all cases our proposed NN-based preorderer achieves the lowest normalized crossing score among all preordering schemes.

3.3 Translation performance

For translation experiments, we use a phrase-based decoder that incorporates a set of standard features and a hierarchical reordering model (Galley and Manning, 2008). The decoder stack size is set to 1000. Weights are tuned using MERT to optimize BLEU on the dev set. In English-to-Japanese and Chinese we use character-BLEU instead. To minimise optimization noise, we tune all our systems from flat parameters three times and report average BLEU score and standard deviation on the test set.

Table 1 contrasts the performance obtained by the system when using no preordering capabilities (baseline), and when using three alternative preordering schemes: the rule-based approach of Genzel (2010), the linear-model logistic-regression approach of Jehl et al (2014) and our NN-based preorderer. We report two baselines: one with distortion limit $d = 10$ and another with $d = 3$. For systems

d	system	speed ratio	eng-jpn in	eng-jpn mixed	eng-kor in	eng-kor mixed
10	baseline	1x	54.5 ±0.2	26.2 ±0.2	**33.5** ±0.3	9.7 ±0.2
3	baseline	3.2x	50.9 ±0.2	25.0 ±0.2	28.7 ±0.1	8.2 ±0.1
3	Genzel (2010)	2.7x	54.0 ±0.1	26.4 ±0.2	30.5 ±0.2	9.8 ±0.2
3	Jehl et al (2014)	2.3x	55.0 ±0.1	26.9 ±0.2	33.1 ±0.1	10.4 ±0.1
3	*this work*	2.7x	**55.6** ±0.2	**27.2** ±0.1	**33.4** ±0.1	**10.6** ±0.2

d	system	eng-chi in	eng-chi mixed	eng-ara in	eng-ara mixed	eng-hin mixed	eng-hin wmt14
10	baseline	**46.9** ±0.5	18.4 ±0.6	25.1 ±0.1	**22.7** ±0.2	10.1 ±0.3	11.7 ±0.1
3	baseline	44.8 ±0.7	18.3 ±0.4	24.6 ±0.1	21.9 ±0.2	8.3 ±0.2	9.3 ±0.3
3	Genzel (2010)	45.4 ±0.2	17.9 ±0.2	24.8 ±0.1	21.6 ±0.3	9.6 ±0.2	11.4 ±0.3
3	Jehl et al (2014)	45.8 ±0.1	18.5 ±0.3	25.1 ±0.2	22.4 ±0.2	10.0 ±0.1	**12.7** ±0.3
3	*this work*	**46.5** ±0.4	**19.2** ±0.2	**25.5** ±0.2	**22.6** ±0.1	**10.6** ±0.1	**12.6** ±0.3
				best WMT14 constrained system			11.1

Table 1: Translation performance for various language pairs using no preordering (baseline), and three alternative preordering systems. Average test BLEU score and standard deviation across 3 independent tuning runs. Speed ratio is calculated with respect to the speed of the slower baseline that uses a $d = 10$. Stack size is 1000. For eng-jpn and eng-chi, character-based BLEU is used.

with preordering we only report $d = 3$, as increasing d does not improve performance.

As shown, our preorderer obtains the best BLEU scores for all reported languages and domains, proving that the neural network is modeling the dependency tree node-swapping phenomenon more accurately, and that the reductions in crossing score reported in the previous section have a positive impact in the final translation performance.

The bottom right-most column reports results on the WMT 2014 English-to-Hindi task. Our system achieves better results than the best score reported for this task[4]. In this case, the two logistic regression preorderers perform similarly, as standard deviation is higher, possibly due to the small size of the dev set.

3.4 Decoding Speed

The main purpose of preordering is to find a better translation performance in fast decoding conditions. In other words, by preordering the text we expect to be able to decode with less distortion or phrase reordering, resulting in faster decoding. This is shown in Table 1, which reports the speed ratio between each system and the speed of the top-most baseline, as measured in English-to-Japanese in-domain[5]. We find that decoding with a $d = 3$ is about 3 times faster than for $d = 10$.

We now take this further by reducing the stack size from 1000 to 50; see results in Table 2. As expected, all systems accelerate with respect to our initial baseline. However, this usually comes at a cost in BLEU with respect to using a wider beam, *unless* preordering is used. In fact, the logistic regression preorderers achieve the same performance while decoding over 60 times faster than the baseline.

Interestingly, the NN-based preorderer turns out to be slightly faster than any of the other preordering approaches. This is because there is no need to explicitly create thousands of feature combinations for each node pair; simply performing the forward matrix multiplications given the input sequence of $\sim$20 features is more efficient. Similar observations have been noted recently in the context of dependency parsing with NNs (Chen and Manning, 2014). Note also that, on average, only 25 pair-wise probabilities are queried to the logistic regression model per source sentence. Overall, we incorporate the benefits of neural networks for preordering at no compu-

[4]Details at matrix.statmt.org/matrix/systems_list/1749

[5]Similar speed ratios were observed for other language pairs (not reported here for space)

d	system	speed	eng-jpn		eng-kor		eng-chi
	w/ stack=50	ratio	in	mixed	in	mixed	mixed
10	baseline	22x	53.6 (-0.9)	25.4 (-0.8)	32.8 (-0.7)	9.3 (-0.4)	17.9 (-0.5)
3	baseline	66x	50.5 (-0.4)	24.8 (-0.2)	28.8 (+0.1)	8.1 (-0.1)	18.0 (-0.3)
3	Genzel (2010)	64x	53.8 (-0.2)	26.3 (-0.1)	30.4 (-0.1)	9.8 (0.0)	18.1 (+0.2)
3	Jehl et al (2014)	61x	55.0 (0.0)	26.5 (-0.4)	33.0 (-0.1)	10.4 (0.0)	18.3 (-0.2)
3	*this work*	65x	**55.7** (+0.1)	**27.2** (0.0)	**33.2** (-0.2)	**10.8** (+0.2)	**19.1** (-0.1)

Table 2: Translation performance for maximum stack size of 50. The figures in parentheses indicate the difference in BLEU scores due to using a smaller stack size, that is, compared to the same systems in Table 1. Speed ratio is calculated with respect to the speed of the slower baseline that uses a stack of 1000, eg. the first row in Table 1.

tational cost.

Currently, our preorderer takes 6.3% of the total decoding time (including 2.6% for parsing and 3.7% for actually preordering). We believe that further improvements in preordering will result in more translation gains and faster decoding, as the distortion limit is lowered.

4 Conclusions

To the best of our knowledge, this is the first paper to describe the usage of NNs in preordering for SMT. We show that simply replacing the logistic regression node-swapping model with an NN model improves both crossing scores and translation performance across various language pairs. Feature combination engineering is avoided, which also results in even faster decoding times.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Machine Learning Research*, 3:1137–1155.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014a. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58.

Ondřej Bojar, Vojtěch Diatka, Pavel Rychlý, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014b. HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of LREC*, pages 3550–3555.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause Restructuring for Statistical Machine Translation. In *Proceedings of ACL*, pages 531–540.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

John DeNero and Jakob Uszkoreit. 2011. Inducing Sentence Structure from Parallel Corpora for Reordering. In *Proceedings of EMNLP*, pages 193–203.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of ACL*, pages 1370–1380.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of EMNLP*, pages 847–855.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of COLING*, pages 376–384.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of LREC*, pages 43–46.

Laura Jehl, Adrià de Gispert, Mark Hopkins, and Bill Byrne. 2014. Source-side preordering for translation using logistic regression and depth-first branch-and-bound search. In *Proceedings of EACL*, pages 239–248.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*, pages 1700–1709.

Uri Lerner and Slav Petrov. 2013. Source-Side Classifier Preordering for Machine Translation. In *Proceedings of EMNLP*, pages 513–523.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proceedings of EMNLP*, pages 567–577.

Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a Discriminative Parser to Optimize Machine Translation Reordering. In *Proceedings of EMNLP-CoNLL*, pages 843–853.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Glsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Theerawat Songyot and David Chiang. 2014. Improving word alignment using word similarity. In *Proceedings of EMNLP*, pages 1840–1845.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of EMNLP*, pages 14–25.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Proceedings of ACL*, pages 1470–1480.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of EMNLP*, pages 1387–1392.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a Dependency Parser to Improve SMT for Subject-Object-Verb Languages. In *Proceedings of HTL-NAACL*, pages 245–253.

Nan Yang, Mu Li, Dongdong Zhang, and Nenghai Yu. 2012. A ranking-based approach to word reordering for statistical machine translation. In *Proceedings of ACL*, pages 912–920.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Proceedings of ACL*, pages 166–175.