The Prague Bulletin of Mathematical Linguistics
NUMBER 108 JUNE 2017 61–72

Integration of a Multilingual Preordering Component into a Commercial SMT Platform

Anita Ramm,[a] Riccardo Superbo,[b] Dimitar Shterionov,[b] Tony O'Dowd,[b] Alexander Fraser[c]

[a] IMS, University of Stuttgart
[b] KantanMT.com
[c] CIS, LMU Munich

Abstract

We present a multilingual preordering component tailored for a commercial Statistical Machine translation platform. In commercial settings, issues such as processing speed as well as the ability to adapt models to the customers' needs play a significant role and have a big impact on the choice of approaches that are added to the custom pipeline to deal with specific problems such as long-range reorderings.

We developed a fast and customisable preordering component, also available as an open-source tool, which comes along with a generic implementation that is restricted neither to the translation platform nor to the Machine Translation paradigm. We test preordering on three language pairs: English→Japanese/German/Chinese for both Statistical Machine Translation (SMT) and Neural Machine Translation (NMT). Our experiments confirm previously reported improvements in the SMT output when the models are trained on preordered data, but they also show that preordering does not improve NMT.

1. Introduction

Statistical Machine Translation (SMT) is still the most widely used machine translation paradigm in commercial translation services. Unlike previous approaches, SMT models can be trained very fast and do not require any language-specific knowledge, but only parallel bilingual data. Translation quality in SMT typically depends on the quality and quantity of the training data, but also on the syntactic and morphological

Corresponding author: ramm@ims.uni-stuttgart.de

differences between the source and the target languages. One of the most common and well analysed problems in SMT is how to place translated words in the correct order with respect to the target language. Often, when the source language (SL) and the target language (TL) have a different syntax, SMT places the TL words in incorrect positions or even omits them. The former case hinders translation fluency, but usually does not strongly affect the meaning of the translation. The latter case, however, damages translation adequacy and may have a negative effect on the conveyed meaning, because specific information given in the source may be missing in the translation.

One of the simplest, yet most effective ways to deal with reordering problems in SMT is to move the words in the SL to positions that are typical of the TL prior to training and translation. This approach, called *preordering*, is performed using rules which describe movements of words or word sequences, typically expressed in terms of part-of-speech (POS) tags or syntactic subtrees. Preordering decreases the syntactic differences between SL and TL sentences and allows for a correct alignment of words in discontiguous phrases. By making the SL and TL look more similar, the long-range reorderings, which are troublesome for automatically-learnt lexicalised reordering models, become much less problematic. The work published on preordering (see Section 2) reports very impressive improvements in the translation quality.

In this paper, we describe the design and the implementation of a preordering approach as well as its integration into KantanMT[1], a commercial custom MT platform. Both the implementation and the integration need to observe the following set of requirements: (i) the implementation must be customisable according to the clients' needs; (ii) the integration of preordering into the training and translation pipelines of the platform should happen seamlessly and sustain backward compatibility; and (iii) the newly integrated preordering component should add as little overhead as possible to the total training/translation time. We focus on the extendible implementation of a preordering component and show how it can be tailored to each user's individual needs. We tested our preordering component on three language pairs: English (EN)$\rightarrow$German (DE)/Japanese (JA)/Chinese (ZH)[2], and report results gained when different parsers are used. Despite the fact that our preordering component is inspired by SMT, it can seamlessly be applied to NMT as well. Our experiments will however show that preordering does not improve NMT.

The remainder of the paper is structured as follows. Section 2 briefly outlines the relevant previous work and motivates the development of the preordering component in the present work. In Section 3, we present the reordering rules for the language pairs under consideration. In Section 4, we describe in detail the implementation of the preordering component. In Section 5, we evaluate its effects within the extended MT platform. Finally, we draw conclusions in Section 6.

[1]`https://kantanmt.com/`

[2]By Chinese, we mean Simplified Mandarin Chinese.

2. Related work

Many approaches have been proposed to deal with reordering problems within SMT. One of the simplest, yet most effective methods is *preordering*, which involves a modification of the SL data prior to training and translation. The reordering rules are usually defined on the basis of POS tags and/or syntactic node labels in the source language parse trees. The rules may be hand-crafted or automatically derived from the word-aligned parallel texts. They may be deterministic (i.e., leading to a single reordered variant of the given source sentence) or non-deterministic (i.e., leading to several variants of the source sentence). An extensive overview of different preordering approaches is presented by Bisazza and Federico (2016).

Xu et al. (2009) and Nakagawa (2015) proposed preordering methods which can be applied to many different language pairs. In a multilingual environment, such methods are certainly very convenient. Moreover, the method advanced by Nakagawa (2015) is very fast, as it does not require any linguistic preprocessing (e.g., tagging or parsing) of the training data. Thus, it additionally fits the speed requirements of commercial MT software. However, the approach discussed by Nakagawa (2015) is non-deterministic: a single source sentence is transformed into a number of different reordered variants. As such, it requires lattice-based tuning and decoding which is not supported by the in-house pipeline that we aim to extend. When many different rules can be applied to a single sentence, it also becomes difficult to track errors in the MT output which may be caused by incorrect reordering rules. In the context of commercial settings, however, we need to have the possibility to (manually) improve the rules in order to further increase the quality of the generated translations.

To allow for modification and adaptation of the reordering rules, and encouraged by the simplicity of the deterministic preordering approaches as well as by the improvements reported for the deterministic rules, we present the implementation and integration of the deterministic preordering approach into a commercial, multilingual MT platform. Like Xu et al. (2009), we work with a single source language (English) which translates into three different target languages: German, Japanese and Chinese. Our method relates to the approaches proposed by Gojun and Fraser (2012) for EN→DE and Lee et al. (2010) for EN→JA translation. Both approaches use a set of deterministic hand-crafted reordering rules and apply them to the source-side (i.e., English) constituency parse trees. Both works report on significant improvements in the MT outputs.

3. Reordering rules

Our rule sets are hand-crafted by the language-pair experts taking into account the rules described by Gojun and Fraser (2012) and Lee et al. (2010).

English-German. The main syntactic difference between English and German is the position of the verbs. Depending on the clause type, the verbs in German may be in the second position (SVO) or in the last position (SOV) in a clause, while in English the sentence structure is always SVO. Our rule set is based on the rules described by Gojun and Fraser (2012). We defined nine reordering rules which move the verbal elements of the English verbal phrases, as well as the negation particle *not*. The rules are conditioned by the clause types (e.g., S, SBAR) since the position of the German verbs depends on the type of clause in which they occur.

English-Japanese. English and Japanese differ in many syntactic aspects: the order of the clauses is different, as well as the order of the words within the clauses. An extensive overview of the differences on various syntactic levels can be found in Bisazza and Federico (2016). The rule set for Japanese is taken from Lee et al. (2010). We only applied context-free grammar (CFG) rules and omitted context-sensitive grammar (CSG) rules, since the information required for such rules is not given in our parse trees. In total, we defined seven reordering rules which change the position of specific subordinate clauses and parts-of-speech (i.e., verbs or conjunctions). Additionaly, we defined one insertion rule to handle null subjects in Japanese.

English-Chinese. To our knowledge, there is no previous work on preordering for EN→ZH. Wu (2016) manually inspected Chinese SMT output and categorised errors related to reordering problems. Relying on this work, as well as on the work on preordering for ZH→EN proposed by Wang et al. (2007), we defined a set of rules which include clause movements, such as moving subordinate clauses before main clauses, as well as various phrase movements. However, this set of rules did not lead to satisfying results and, after further investigation, we reduced the set to only two different rules: (i) moving all PPs before the modifying noun, (ii) moving only PPs with the preposition *of* in front of the modifying noun (this is a subset of the movements defined by the preceding rule). We report results for using either rule (i) or rule (ii)).

4. Implementation

The KantanMT platform is based on the SMT Moses toolkit (Koehn et al., 2007). The preordering component acts as part of the data preprocessing step in the training and translation pipelines.[3] It is applied to the training/testing data before all the other corpus-processing steps, such as lowercasing, cleansing, etc. Preordering is invoked only if reordering rules for the given language pair are provided. Otherwise, the processing pipeline simply skips the preordering step.

[3]A simplified version of the preordering component is freely available for research purposes: https://github.com/KantanLabs/KantanPreorder.

4.1. Pipeline overview

The preordering component is implemented as a three-step process that uses the training, tuning and testing data in the SL (English) as input data, and reorders it sentence by sentence. The processing steps are illustrated in Figure 1. For a general sentence ω, we first generate the constituent parse tree T_ω. We then apply the tree modifications according to our reordering rules to generate the reordered tree T_ω^r. Finally, we read out the reordered sentence ω^r from the modified tree T_ω^r.

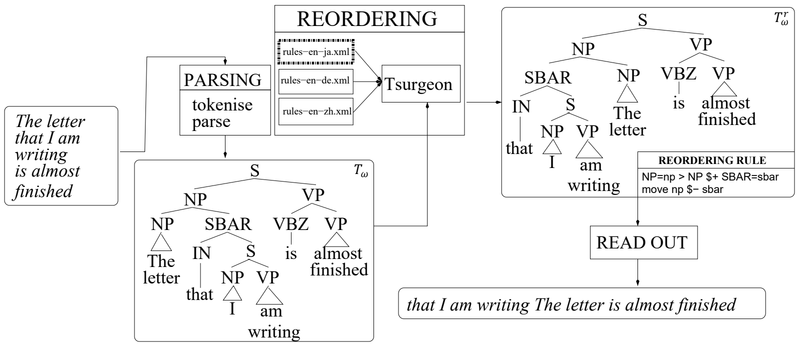
The letter that I am writing is almost finished
PARSING
tokenise parse
REORDERING
rules−en−ja.xml
rules−en−de.xml
rules−en−zh.xml
Tsurgeon
T_ω
S NP VP NP SBAR VBZ VP The letter IN S is almost finished that NP VP I am writing
T_ω^r
S NP VP SBAR NP VBZ VP IN S The letter is almost finished that NP VP I am writing
REORDERING RULE
NP=np > NP $+ SBAR=sbar
move np $− sbar
READ OUT
that I am writing The letter is almost finished

Figure 1. Pipeline for the multilingual preordering. The figure shows tree modification for one of the rules used for Japanese.

Parsing of SL data. Our approach employs reordering of English constituent parse trees. We use the Stanford Shift-Reduce (SR) constituency parser (Zhu et al., 2013) to generate these trees mainly because of its speed (see Section 5). But our implementation also works with two other parsers: the Charniak-Johnson parser (Charniak and Johnson, 2005) and the Stanford PCFG constituency parser (Klein and Manning, 2003). The preordering component does not parse nor reorder sentences that are longer than 60 words, shorter than 5 words or contain many special characters. We impose this restriction because parsers may generate incorrect parse trees or take too long to parse such sentences.

Tsurgeon-based reordering. For modifications of the parse trees, we employ Tsurgeon (Levy and Andrew, 2006) – a tool for parse tree editing based on regular expressions. Tsurgeon first uses a pattern, defined as a Tregex expression, to identify specific subtrees. These expressions make use of relational dependencies between tree nodes, such as *immediate dominance* and *precedence*. Once a subtree matches the pattern, Tsurgeon applies basic tree transformations to it (e.g., move, insert, etc.). An example is given in Figure 1: the applied rule shows the movement (*move np $- sbar*) of the relative clause under the SBAR node (*SBAR=sbar*) in front ($-) of the modifying noun phrase (*NP=np*). Before applying reordering rules to a parse tree, we modify the tree to ensure that reordering operations will not cross the clause boundaries. That is to say, no word movement will place a word outside of the corresponding clause. Afterwards, we apply the language-pair specific reordering operations.

Reading out the reordered sentences. Given the modified parse trees, the reordered sentences are read out by gathering the terminal nodes. Subsequently, the entire source language data undergoes the tokenisation and lowercasing steps.

4.2. Optimised performance and scalability

Training and translation speeds are crucial for a commercial MT system's quality of service. To perform preordering efficiently, we developed the preordering component with a distributed software architecture, and optimised both the parser and Tsurgeon.

First, we run the parser as a simple web service on the machine used for training or translation. This ensures that the parsing model is loaded into the memory prior to parsing any sentence. Furthermore, running the parser as a simple web service makes the implementation independent from the parsing software used. Next, we modify Tsurgeon and introduce a limited-depth search in order to avoid infinite loops and ensure that the reordering of a single sentence will always terminate. In case either the parser or Tsurgeon fails, we output the original sentence. This way, we preserve coherence within the training/translation data.

Our preordering component uses GNU parallel (Tange, 2011) to distribute the workload on all available cores. In particular, we divide the data into as many parts as the CPUs and run preordering for each part in parallel and asynchronously. The GNU parallel tool orchestrates the execution and ensures that the output is serialised correctly. The parallel architecture leads to a substantially lower reordering time as compared to the non-parallel implementation. In a particular test case for EN→DE, involving the reordering of 5000 segments with the Stanford shift-reduce parser, the parallel implementation on 8 cores took $46.10s$, whereas the serial took $263.24s$.

The run-time depends on the complexity of the sentences, number of the rules, the parser, as well as parsing model used, etc. Analysis of the effects of these factors on the efficiency is out of the scope of this paper and shall be addressed in future work.

4.3. Customisability

Since the tree modifications are based on Tsurgeon, they are independent of the language pair for which the reordering is to be performed (as long as there is a corresponding constituent parse tree). Thanks to the well-defined syntax of the rules, it is easy to extend and/or modify the existing rule sets. In order to add reordering for a new target language (and English as the source language), it is only necessary to specify the new reordering rules. That is, no further adaptation of the training/translation pipelines is needed. Applying reordering to a new source language would, however, require the pipelines to be adapted and new parsers or models to be incorporated. In principle, this is an easy task thanks to the generic implementation of the preordering component.[4] Furthermore, since some rules are shared across languages, there are cases where already existing rules can be re-used for new language pairs. For instance, a rule for moving verbs at the end of the clause can be used for all SVO-SOV language pairs. Rules can be developed by anyone familiar with Tsurgeon syntax. However, language proficiency and translation experience are required in order to create a valid set of rules.

5. Evaluation

We evaluated our preordering component in different settings and examined the benefit of preordering on both SMT and NMT. All models were trained on the same sets of data from the legal domain. The German models were trained on 1,018,738 parallel sentences, while for Japanese and Chinese, the training data consisted of 213,592 and 387,275 sentences, respectively. The models were tuned and tested on 500 in-domain sentences. The MT quality is evaluated in terms of BLEU (Papineni et al., 2002), TER (Snover et al., 2006) and F-Measure (Melamed et al., 2003). In addition, we give the time required to preorder the SL data used to build the translation models.

5.1. Preordering and SMT

The models were trained with the SMT Moses toolkit (Koehn et al., 2007). We used Moses default settings, including the lexicalised reordering model with distortion limit of 6 words. The 5-gram language models were trained with the target side of the parallel training data. We used *fast_align* for word alignment (Dyer et al., 2013). Model weights were tuned with MERT (Och, 2003) with a maximum of 25 iterations.

The evaluation results, as well as the total training time (including reordering and tuning time), are given in Table 1. The scores show that the quality of the translations varies when different parsers are used. For all target languages, the MT output improves for all parsers. The highest MT improvement for German (+1,39 BLEU) is obtained when the BLLIP parser is used, while the Japanese translations improve the most when reordering is performed on the output produced through the SR parser (+1,89 BLEU). The Chinese MT output profits the most from the reordering of *of*-PPs in the PCFG trees (+0,13 BLEU).

Parsing accuracy depends on the domain and type of training data (Kummerfeld et al., 2012). It may thus be interesting to extend the preordering component with domain-dependent parsing software, as well as domain-specific parsing models.

[4]In Section 5, we present our experiments with two different parsers and three different models, which is evidence of the extendibility of our tool.

	Baseline				SR					PCFG					BLLIP				
	TER	F	BLEU	t_t	TER	F	BLEU	t_r	t_t	TER	F	BLEU	t_r	t_t	TER	F	BLEU	t_r	t_t
EN→DE	51.73	64.21	40.1	187	49.57	65.53	40.74	97	254	49.7	65.57	41.17	372	579	50.45	64.6	41.49	1279	1468
EN→JA	54.02	78.22	49.44	135	51.87	76.34	51.33	25	155	52.04	77.43	50.29	413	544	54.54	76.04	51.33	372	492
EN→ZH (ppNP)	66.27	61.04	24.99	197	66.1	60.55	24.4	50	245	65.97	61.01	24.47	252	460	66.76	60.75	24.66	627	819
EN→ZH (ofPP)					66.69	61.50	25.09	49	240	66.11	61.53	25.22	269	464	67.32	61.36	25.05	633	820

Table 1. Automatic evaluation scores (given as percentages) for the SMT models together with the time (given in minutes) for reordering (t_r) and the total training time (t_t), including tuning. The run times relate to the 8-core CPU machines. Parsers: SR: Stanford shift-reduce parser, PCFG: Stanford PCFG parser, BLLIP: Charniak/Johnson parser.

5.2. Preordering and NMT

Training setting. Our NMT models are built on the same data as our SMT models, after removing duplicates of source-target sentences. We used the open-source toolkit OpenNMT (Klein et al., 2017) to train a single RNN (Recurrent Neural Network) encoder-decoder model (Cho et al., 2014; Sutskever et al., 2014) with attention mechanism (Bahdanau et al., 2014). We used a word-segmentation with byte pair encoding (BPE) (Sennrich et al., 2016) of 25,000 operations for English and German. We built the BPE dictionary from normal-cased (i.e., lower- and upper-cased) tokens bypassing the requirement for a recasing model. For Chinese and Japanese, we used a character-based segmentation (Chung et al., 2016). Each network was trained on one GPU (NVIDIA K520, 4GB RAM) for a maximum of 15 epochs, using the ADAM (Kingma and Ba, 2015) learning optimisation function with initial learning rate of 0.005.

We need to highlight that our NMT pipeline is optimised for speed (e.g., EN→JA models are built in less than 8 hours); within the scope of this work, we did not aim to build NMT models that perform better than SMT (according to the automatic metrics), but rather to explore the impact of preordering on NMT.

Automatic evaluation. The evaluation scores are presented in Table 2. In addition to TER, F-Measure and BLEU, we also give the models' perplexity during training to assess the effect of reordering on NMT engines. The scores indicate that, overall, preordering does not improve the quality of NMT models. On the contrary, all metrics, including perplexity, are better for the baseline models. However, we ought to note that the EN-ZH (*of*PP) NMT model has the highest BLEU score for EN→ZH. This result, although episodic for our data, indicates that preordering can have a positive effect under certain conditions. This calls for further, in-depth analysis, which we plan to address in future work.

Human evaluation. Automatic evaluation metrics often tend to misjudge NMT quality (Shterionov et al., 2017). Therefore, we carried out human evaluation tests on Chinese (80 sentences) and German (250 sentences) MT output. For each of the two

	Baseline						SR					
	Perplexity	TER	F	BLEU	Human	t_e	Perplexity	TER	F	BLEU	Human	t_e
EN-DE	2.83	54.63	63.07	38.26	49.2	123	2.94	54.84	61.42	36.74	50.8	123
EN-JA	1.41	27.44	84.54	67.66	–	31	1.5	35.28	80.62	60.77	–	31
EN-ZH (ppNP)	3.46	63.34	61.01	27.65	36.9	91	3.71	67.15	59	26.67	30.7	91
EN-ZH (ofPP)							3.66	65.48	60.37	28.75	32.4	91

Table 2. Scores (given in percentages) together with training time (given in minutes) for one epoch (t_e) for the baseline and reordered NMT models. The human evaluation indicates the percentage of sentences for which the translation is deemed better.

EN	The Commission *may*, in any case, *withdraw* **such products** or substances in accordance with Article37(2).
ENr	The Commission *may*, in any case, **such products** or substances in accordance with Article37(2) *withdraw*.
B	Die Kommission *kann* in jedem Fall **diese Produkte** oder Stoffe gemäß Artikel37 Absatz2 *zurückziehen*.
R	Die Kommission *kann* in jedem Fall **solche Erzeugnisse** oder Stoffe gemäß Artikel37 Absatz2 *zurückziehen*.
REF	Kommission *kann* in jedem Fall **solche Erzeugnisse** oder Stoffe gemäß Artikel37 Absatz2 *zurückziehen*.

Table 3. Example of baseline (B) and reordered (R) translation of a sentence EN and its reordered version ENr. The verbs are indicated in italic, while the differing object NPs are given in bold. The German reference is indicated as REF.

languages, two reviewers compared randomly selected MT sentence pairs (obtained using reordered (R) and baseline (B) training data). For EN-DE they had to indicate which of the two translations was better or whether they were the same; for EN-ZH they had to compare three translations (B, ppNP and ofPP) and score each of them on the scale of 1 to 5. We mainly notice: (i) the translation quality of the B translations is slightly better than that of the R translations, (ii) reordering does not seem to impact the placement of (single) words in the NMT output, but it may lead to syntactically completely different translations, as well as different lexical choices, and (iii) often the B models already correctly translate sentences which our preordering aims to correct.

Discussion. Bentivogli et al. (2016) showed that NMT deals very well with word order issues for EN→DE. Mainly, this is because the RNN encoder-decoder model encapsulates knowledge of the complete input sentence. That is, a complete input sentence is mapped to a complete output sentence, contrary to phrase-based SMT where one sentence is handled phrase by phrase. This allows NMT to deal with both short- and long-distance order issues much more efficiently. Despite showing great improvement when compared to SMT, NMT still makes some mistakes in relation to the placement of words in the translations. We applied preordering on NMT to examine the possibility to reach further improvement in the NMT quality. Our experiments showed that preordering is not beneficial for NMT based on RNNs. This may be explained by the fact that preordering is applied on some, but not all source sentences (depending on the parser's accuracy and coverage of the preordering rules), which leads to noisy training data. Although adding noise to a neural network may improve the generalisation abilities of the network (Jim et al., 1994; Bishop, 1995), in our experiments we did not use any technique to accommodate any excessive noise introduced by the reordering, which may result in a lower network performance.

For future works, we plan to investigate in depth this hypothesis, and upgrade our preordering component to address performance issues, aiming to improve the translation quality of NMT models.

5.3. Processing time vs. translation quality improvement

Since the processing time plays an important role for commercial MT, we ought to investigate whether the improvements reported in Section 5.1 justify the longer processing time. Given the baseline training time (see Table 1), the total training time increases by 36% for EN→DE and 15% for EN→JA when the fastest parser (SR parser) is used. BLEU improvements, and even more importantly, positive feedback of our clients, justify longer processing time for both language pairs. For Chinese, the increase is 24% for EN→ZH (*of*PP) and 22% for EN→ZH (ppNP). On the other hand, the PCFG (4-6 hours) and BLLIP (6-20 hours) parsers lead to a non-acceptable increase of the training time, although for German and Chinese, the best translations are obtained using the BLLIP and PCFG parser, respectively. Future work will aim at making these parsers faster so as to be usable within our commercial SMT platform.

In some settings, however, an increase of processing time may be acceptable if it promises high-quality translations. For example, for translation via API, where only a few segments are translated at once, the increase in time is negligible. Furthermore, if a single model is to be used for many decoding iterations, one could consider training it using a slower, but better parser. Ultimately, it is up to the clients to decide how fast the translations of the provided test sets are to be generated. Given the evaluation results and the training times of the SMT models, we suggest employing the fastest SR parser.

6. Conclusion

We presented a generic component for preordering that is integrated in the corpus preprocessing step for a commercial MT platform. Reordering of the SL sentences is based on Tsurgeon, a tool for editing parse trees based on regular expressions. Thanks to the well-defined syntax of the Tsurgeon expressions, the preordering component is easy to maintain and to extend to other language pairs.

We implemented deterministic rule-based reordering because it performs well, and we can control and adapt it, if needed, to maximise the translation quality. Furthermore, due to its deterministic character, we are not forced to choose between different reorderings of a single sentence or to modify the pipeline into which the preordering component has been integrated.

We described how to achieve preordering speeds that can satisfy the high performance demands of commercial MT software. We showed that a single preordering pipeline can successfully be applied to three different language pairs. Furthermore, the EN→ZH language pair has not been handled this way before. Our experiments confirmed previously reported improvements for combining preordering with SMT. Additionally, we applied preordering to NMT and observed that NMT does not generally benefit from the reordering of the source training data. In the future work, we will further investigate impact of the preordering approach on NMT.

Acknowledgements
This research was supported by the European Association for Machine Translation.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *ICLR*, May 2014.

Bentivogli, Luisa, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. Neural versus Phrase-Based Machine Translation Quality: a Case Study. In *EMNLP*, November 2016.

Bisazza, Arianna and Marcello Federico. A Survey of Word Reordering in Statistical Machine Translation: Computational Models and Language Phenomena. *Computational linguistics*, 42(2), 2016.

Bishop, Chris M. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1), January 1995.

Charniak, Eugene and Mark Johnson. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *ACL*, June 2005.

Cho, Kyunghyun, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*, 2014.

Chung, Junyoung, Kyunghyun Cho, and Yoshua Bengio. A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. In *ACL*, 2016.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the NAACL-HLT*, Atlanta, USA, June 2013.

Gojun, Anita and Alexander Fraser. Determining the Placement of German Verbs in English-to-German SMT. In *EACL*, 2012.

Jim, Kam, Bill G Horne, and C Lee Giles. Effects of noise on convergence and generalization in recurrent networks. In *NIPS*, 1994.

Kingma, Diederik P. and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.

Klein, Dan and Christopher D. Manning. Accurate Unlexicalized Parsing. In *ACL*, 2003.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. 2017.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL*, 2007.

Kummerfeld, Jonathan K., David Hall, James R. Curran, and Dan Klein. Parser Showdown at the Wall Street Corral: An Empirical Investigation of Error Types in Parser Output. In *EMNLP-CoNLL*, 2012.

Lee, Young-Suk, Bing Zhao, and Xiaoqiang Luo. Constituent Reordering and Syntax Models for English– to–Japanese Statistical Machine Translation. In *COLING*, 2010.

Levy, Roger and Galen Andrew. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *LREC*, 2006.

Melamed, I. Dan, Ryan Green, and Joseph P. Turian. Precision and Recall of Machine Translation. In *NAACL-HLT*, 2003.

Nakagawa, Tetsuji. Efficient Top-Down BTG Parsing for Machine Translation Preordering. In *ACL-NLP*, 2015.

Och, Franz J. Minimum Error Rate Training in Statistical Machine Translation. In *ACL*, 2003.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *ACL*, 2002.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *ACL*, 2016.

Shterionov, Dimitar, Pat Nagle, Laura Casanellas, Riccardo Superbo, and Tony O'Dowd. Empirical Evaluation of NMT and PBSMT Quality for Large-scale Translation Production. In *EAMT*, 2017.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *AMTA*, 2006.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 2014.

Tange, Ole. GNU Parallel: The Command-Line Power Tool. *;login: The USENIX Magazine*, February 2011.

Wang, Chao, Michael Collins, and Philipp Koehn. Chinese Syntactic Reordering for Statistical Machine Translation. In *EMNLP*, 2007.

Wu, Peiyu. Word order errors in Simplified Chinese MT. *MultiLingual*, October/November 2016.

Xu, Peng, Jaeho Kang, Michael Ringgaard, and Franz J. Och. Using a Dependency Parser to Improve SMT for Subject-object-verb Languages. In *NAACL-HLT*, 2009.

Zhu, Muhua, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and Accurate Shift-Reduce Constituent Parsing. In *ACL*, 2013.

Address for correspondence:
Anita Ramm
ramm@ims.uni-stuttgart.de
University of Stuttgart, Institute for Natural Language Processing,
Pfaffenwaldring 5b, 70569 Stuttgart, GERMANY