Neural Lattice Search for Domain Adaptation in Machine Translation

Huda Khayrallah[†] **Gaurav Kumar**[†] **Kevin Duh**[‡] **Matt Post**[‡] **Philipp Koehn**[†]
[†]Center for Language and Speech Processing
[‡]Human Language Technology Center of Excellence
Johns Hopkins University
{huda, gkumar, kevinduh, post, phi}@cs.jhu.edu

Abstract

Domain adaptation is a major challenge for neural machine translation (NMT). Given unknown words or new domains, NMT systems tend to generate fluent translations at the expense of adequacy. We present a stack-based lattice search algorithm for NMT and show that constraining its search space with lattices generated by phrase-based machine translation (PBMT) improves robustness. We report consistent BLEU score gains across four diverse domain adaptation tasks involving medical, IT, Koran, or subtitles texts.

1 Introduction

Domain adaptation is a major challenge for neural machine translation (NMT). Although impressive improvements have been achieved in recent years (c.f. Bojar et al. (2016)), NMT systems require a large amount of training data and thus perform poorly relative to phrase-based machine translation (PBMT) systems in low resource and domain adaptation scenarios (Koehn and Knowles, 2017). In such situations, neural systems often produce fluent output that unfortunately contains words not licensed by the unfamiliar source sentence (Arthur et al., 2016; Tu et al., 2016). Phrase-based systems, in contrast, explicitly model the translation of all source words via coverage vectors, and tend to produce translations that are adequate but less fluent. This situation is depicted in Table 1, which contains examples of PBMT and NMT systems trained on WMT training sets which are then applied to IT texts.

We present an approach that combines the best of both worlds by using the lattice output of PBMT to constrain the search space available to an NMT decoder, thereby bringing together the *adequacy* and the *fluency* properties of PBMT and NMT systems. The final line of Table 1 demonstrates the improvement this can bring. Our contributions are (1) a simple stack-based lattice search algorithm for NMT,[1] and (2) a set of domain adaptation experiments showing that PBMT lattice constraints are effective in achieving robust results compared to NMT decoding with standard beam search.

src	Versionsinformationen ausgeben und beenden
ref	output version information and exit
PBMT	Spend version information and end
NMT	Spend and end versionary information
NMT$_l$	Print version information and exit

Table 1: Translations of sentence #925 from the IT corpus with systems trained on WMT data. The NMT$_l$ line was produced by a WMT-trained NMT search over a WMT-trained PBMT lattice.

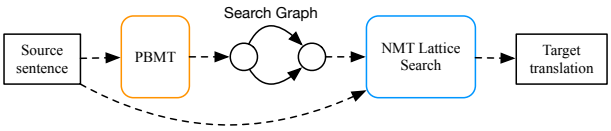

Figure 1: NMT lattice search over a PBMT-generated lattice.

2 Stack-based Neural Lattice Search

Figure 1 demonstrates our system; given an input sentence, the PBMT system generates a lattice, which is then used as input to the neural lattice search algorithm. We would like to score every path in the lattice with the NMT system and then search. However, this is generally prohibitively expensive because the RNN architectures in NMT

[1] github.com/khayrallah/nematus-lattice-search

Proceedings of the The 8th International Joint Conference on Natural Language Processing, pages 20–25,
Taipei, Taiwan, November 27 – December 1, 2017

do not permit recombination of hypotheses on the lattice, since NMT states encode the entire sentence history. This explodes the search lattice into an exponentially sized tree. To address this problem, we use a *stack decoding* algorithm that groups hypotheses by the number of target words, extending items from each stack in order of score, and adding them to later stacks.[2] This strategy allows us to group together roughly equivalent intermediate nodes, allowing for pruning.

Algorithm 1: Stack decoding over a lattice

```
   Data: lattice root N, NMT init state I, beam
         size b
   Result: output string s
1  goalStack = []; stacks = []
2  heappush(stacks[0], (0.0, N, I, null, 0))
3  for i = 0; i <len(stacks); i++ do
4      for b_i in 1 . . . b do
5          score, node, state, _, len =
            heappop(stacks[i])
6          for arc in node.arcs() do
7              newState, cost = scorer(state, arc)
8              newScore = score + cost
9              newLen = len + arc.len
10             if isFinalState(node) then
11                 stack = goalStack
12             else
13                 stack = stack[newLen]
14             heappush(stack, (newScore,
                arc.head, newState, arc,
                newLen))
15 return extractBest(heappop(goalStack))
```

The pseudocode is in Algorithm 1, and a graphical depiction in Figure 2. In the lattice (Figure 2(a)), arcs are annotated with phrases of one or more words indicating the target sides of phrases that were applied during PBMT decoding. Nodes represent recombined states in the PBMT search space (i.e., states that have identical source coverage vectors and language model states). The search nodes contain the cumulative score, the current lattice node, the current neural state, the incoming arc, and the target length along this path. After initialization, the outer loop (line 3) proceeds over stacks, starting at stack 1, and continuing through the longest path through the lattice (subject to pruning). Upon visiting each stack, it considers the top b items (line 4). It pops each of them in turn and retrieves its node in the underlying lattice and the associated neural state (line 5). It then considers all of the node's outgoing arcs (line 6). The neural scorer is used to score each of them (line 7), returning a new neural state that is stored with a new item (line 14) on the appropriate stack.

Figure 2 depicts this process, but without pruning or sorting. A beam of size 2 would prune off one item from stack 2, along with all of its descendants, thus culling the exponentially sized tree.

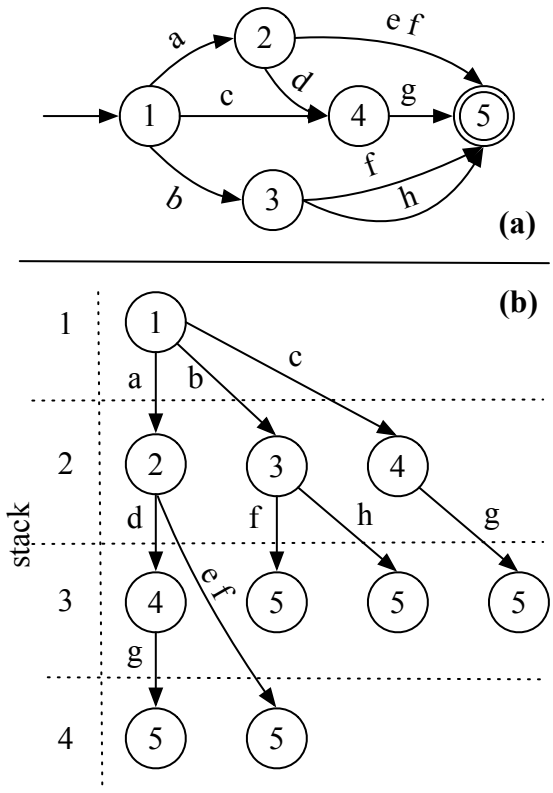

Figure 2: (a) A PBMT search lattice and (b) stack-based decoding over that lattice. Each letter represents a word.
In (b), the exponential expansion of the lattice in (a) is apparent, since states that had recombined in (a) due to identical n-gram history do not recombine in (b). This figure does not demonstrate pruning, descendants of items that fall off the beam would not be explored.

[2]This is similar to PBMT stack decoding. However, in PBMT stack decoding, stacks are grouped by the number of translated source words, which is not possible in NMT, since the translation of individual source words is not tracked.

Corpus	Words	Sentences	W/S
Medical	14,301,472	1,104,752	13
IT	3,041,677	337,817	9
Koran	9,848,539	480,421	21
Subtitles	114,371,754	13,873,398	8
WMT	113,165,079	4,562,102	25

Table 2: Size of the training data for each domain

3 Experiment Setup

Our German-to-English evaluation consists of a large out-of-domain bitext (WMT2016 (Bojar et al., 2016), news/parliamentary text) and four distinct in-domain bitexts from OPUS (Tiedemann, 2009, 2012): Medical (EMEA), IT (GNOME, KDE, PHP, Ubuntu, and OpenOffice), Koran (Tanzil), and Subtitles (OpenSubtitles[3]).

The in-domain corpora use the same train/tune/test splits as Koehn and Knowles (2017), and for each in-domain training set we build PBMT and NMT models, termed PBMT$_{in}$ and NMT$_{in}$. We also build PBMT and NMT models on the out-of-domain WMT bitext, termed PBMT$_{out}$ and NMT$_{out}$. For each **in-domain test set**, we consider four configurations:

1. PBMT$_{out}$ $\times$ NMT$_{out}$: the unsupervised domain adaptation setting where no training data is available for the domain of interest.

2. PBMT$_{in}$ $\times$ NMT$_{in}$: the matched domain setting where the training data matches the test data in terms of domain, but the training data is not large (relative to WMT).

3. PBMT$_{in}$ $\times$ NMT$_{out}$: PBMT is trained on small in-domain data while NMT is trained on larger out-of-domain data.

4. PBMT$_{out}$ $\times$ NMT$_{in}$: NMT is trained on small in-domain data while PBMT is trained on larger out-of-domain data.

For each training configuration, we are interested in seeing how our proposed NMT lattice search compares to standard NMT beam search. Additionally, we compare the results of PBMT 1-best decoding and PBMT N-best lists rescoring (N=500) using the same NMT model.

The PBMT models are trained with Moses (Koehn et al., 2007). The PBMT$_{out}$ models include German specific processing and Neural Network Joint Models (Devlin et al., 2014), replicating Ding et al. (2016). The PBMT$_{in}$ models are Moses models with standard settings, replicating Koehn and Knowles (2017). The NMT models are trained with Nematus (Sennrich et al., 2017). The NMT$_{out}$ models replicate Sennrich et al. (2016);[4] the NMT$_{in}$ models replicate Koehn and Knowles (2017). We use Marian (Junczys-Dowmunt et al., 2016a) to rescore N-best lists.

The search graphs are pre-processed by converting them to the OpenFST format (Allauzen et al., 2007) and applying operations to remove epsilon arcs, determinize, minimize and topsort. Since the search graphs may be prohibitively large in size, we prune them with a threshold.[5] We perform 5-fold cross-validation over pruning thresholds (.1, .25, .5) and lattice search beamsizes (1, 10, 100).

Very aggressive pruning with a small beam limits the search to be very similar to the PBMT output. In contrast, a very deep lattice with a large beam begins to approach the unconstrained search space of standard decoding in NMT.

4 Results

Table 3 summarizes the BLEU results on each test domain. Note that PBMT 1-best results are equivalent for PBMT$_{in}$ $\times$ NMT$_{in}$ and PBMT$_{in}$ $\times$ NMT$_{out}$ since the same PBMT model is used and NMT is not relevant. For both PBMT 1-best and NMT Standard Search, there are two sets of equivalent results among the four training configurations.

We want to highlight the fact that the PBMT 1-best in-domain models outperform the out of domain ones, despite being much simpler models. Additionally, the BLEU scores for NMT standard search are higher for the in-domain models, despite the smaller amount of training data. This emphasizes the importance of the domain of the training corpora.

In cross-validation for our domains, smaller beams and aggressive pruning tend to perform well. This follows from the fact that PBMT 1-best outperforms NMT standard search. We want to strongly limit the search space given to NMT in such a scenario. However, these parameters need to be tuned to a specific domain and language.

[3] opensubtitles.org

[4] github.com/rsennrich/wmt16-scripts

[5] Pruning removes arcs that do not appear on a lattice path whose score is within than $t \otimes w$, where w is the weight of the FST's shortest path, and t is the pruning threshold.

Test Domain	Training Configuration	PBMT 1-best	NMT Standard Search	N-best Rescoring	NMT Lattice Search
IT	PBMT$_{out}$ × NMT$_{out}$	25.1 (-0.3)	22.5 (-2.9)	22.2 (-3.2)	25.4
	PBMT$_{in}$ × NMT$_{in}$	47.4 (-4.2)	34.2 (-17.4)	47.6 (-4.0)	51.6
	PBMT$_{in}$ × NMT$_{out}$	47.4 (-5.2)	22.5 (-30.1)	47.6 (-5.0)	**52.6***
	PBMT$_{out}$ × NMT$_{in}$	25.1 (-2.2)	34.2 (6.9)	22.4 (-4.9)	27.3
Medical	PBMT$_{out}$ × NMT$_{out}$	33.3 (-0.9)	32.9 (-1.3)	30.8 (-3.4)	34.2
	PBMT$_{in}$ × NMT$_{in}$	47.4 (-0.7)	37.8 (-10.3)	40.2 (-7.9)	**48.1***
	PBMT$_{in}$ × NMT$_{out}$	47.4 (-0.4)	32.9 (-14.9)	39.7 (-8.1)	**47.8**
	PBMT$_{out}$ × NMT$_{in}$	33.3 (-2.7)	37.8 (1.8)	31.2 (-4.8)	36.0
Koran	PBMT$_{out}$ × NMT$_{out}$	14.7 (-0.2)	10.8 (-4.1)	13.9 (-1.0)	14.9
	PBMT$_{in}$ × NMT$_{in}$	**20.6 (-0.1)**	15.9 (-4.8)	19.3 (-1.4)	**20.7**
	PBMT$_{in}$ × NMT$_{out}$	**20.6 (-0.2)**	10.8 (-10.0)	19.4 (-1.4)	**20.8***
	PBMT$_{out}$ × NMT$_{in}$	14.7 (-1.4)	15.9 (-0.2)	13.9 (-2.2)	16.1
Subtitle	PBMT$_{out}$ × NMT$_{out}$	26.6 (-0.9)	25.3 (-2.2)	19.7 (-7.8)	27.5
	PBMT$_{in}$ × NMT$_{in}$	26.8 (-1.1)	24.9 (-3.0)	17.8 (-10.1)	**27.9**
	PBMT$_{in}$ × NMT$_{out}$	26.8 (-1.6)	25.3 (-3.1)	17.1 (-11.3)	**28.4***
	PBMT$_{out}$ × NMT$_{in}$	26.6 (-1.0)	24.9 (-2.7)	19.8 (-7.8)	27.6

Table 3: Results across test domains and training configurations. For each system, we show the BLEU score and its difference with NMT Lattice Search under the same training configuration (same row) in parentheses. E.g. in the last row, NMT Lattice Search achieves 27.6 BLEU and is better than PBMT 1-best by 1.0 BLEU, and better than NMT Standard Search by 2.7 BLEU. For each test domain we mark the best score among all systems and training configurations with an asterisk, and bold any score with less than a 0.5 BLEU difference.

Our research questions are as follows:

Does lattice search perform best across training configurations? As observed across each row in Table 3, lattice search typically outperforms the three other systems. Importantly, the BLEU gains against standard beam search in NMT and N-best rescoring of PBMT with NMT are noticeable regardless of training configuration. E.g., in the Subtitles task the gains range from 2.2 to 3.1 BLEU. There are also consistent gains compared to PBMT 1-best (e.g. 0.9-1.6 BLEU gain), which forms the basis of the search space; this implies that PBMT and NMT can serve as effective hybrid systems, where the former provides the potential translation candidates and the latter scores them.

Given the choice, which training configuration is best for domain adaptation? While the answer depends on the amount of in-domain and out-of-domain data, we find that PBMT$_{in}$ × NMT$_{in}$ and PBMT$_{in}$ × NMT$_{out}$ perform the best. This supports previous findings (Koehn and Knowles, 2017) that PBMT$_{in}$ is robust when training data is insufficient. In conclusion, we recommend using lattice search with search graphs from PBMT$_{in}$, and NMT models can be trained on either in-domain or out-of-domain corpora.

5 Related Work

Previous work on domain adaptation in NMT focuses on training methods such as transfer learning or fine-tuning (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016; Chu et al., 2017). This strategy begins with a strong model trained on a large out-of-domain corpus and then continuesx training on an in-domain corpus. Our approach is orthogonal in that we focus on search. Conceivably, advances in training methods might be incorporated to improve our individual NMT$_{in}$ models.

Our lattice search algorithm is related to previous work in hybrid NMT/PBMT systems, which can be visualized on a spectrum depending on how tightly integrated the two systems are. On one end, NMT can easily be used to rerank N-best lists output by PBMT; on the other, NMT can be incorporated as features in PBMT (Junczys-Dowmunt et al., 2016b). In the middle of the spectrum is NMT search (or re-scoring) based on constraints from PBMT.

Our algorithm is conceptually very similar to Stahlberg et al. (2016), who rescore a WFSA reformulation of the Hiero formalism. Their

algorithm is a breadth-first search over all the nodes of the lattice, capped by a beam. Other hybrid methods include: constraining the output vocabulary of NMT on a per-sentence basis, using bilingual information provided by PBMT (Mi et al., 2016), Minimum Bayes Risk decoding with PBMT n-gram posteriors (Stahlberg et al., 2017), and incorporating PBMT hypotheses as additional input in a modified NMT architecture (Wang et al., 2017).

Related works in lattice search/re-scoring with RNNs (without NMT encoder-decoders) (Ladhak et al., 2016; Deoras et al., 2011; Hori et al., 2014) may serve as other interesting comparisons. Specifically, Auli et al. (2013) and Liu et al. (2016) provide alternatives to our approach to the problem of recombination. The former work allows the splitting of previously recombined decoder states (thresholded) while the latter clusters RNN states based on their n-gram context.

6 Conclusion

We present a stack-based lattice search algorithm for NMT, and show that constraining decoding to candidate translations in a PBMT search graph leads to robust improvements for domain adaptation. Our method can be viewed as as simple yet effective way to combine the *adequacy* advantages of PBMT, which stems from explicit models of coverage, with the *fluency* advantages of NMT. When presented with a domain adaptation problem we recommend using lattice search with search graphs from PBMT$_{in}$, with NMT models either trained on either in-domain or out-of-domain corpora.

Future work includes interpolation of the NMT and PBMT scores in the lattice search, which requires additional tuning but may further improve results.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-15-C-0113. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*. Springer. `http://www.openfst.org`.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating Discrete Translation Lexicons into Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas. Association for Computational Linguistics.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint Language and Translation Modeling with Recurrent Neural Networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany. Association for Computational Linguistics.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An Empirical Comparison of Simple Domain Adaptation Methods for Neural Machine Translation. *CoRR*, abs/1701.03214.

Anoop Deoras, Tomáš Mikolov, and Kenneth Church. 2011. A fast re-scoring strategy to capture long-distance dependencies. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland. Association for Computational Linguistics.

Shuoyang Ding, Kevin Duh, Huda Khayrallah, Philipp Koehn, and Matt Post. 2016. The JHU Machine Translation Systems for WMT 2016. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany. Association for Computational Linguistics.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast Domain Adaptation for Neural Machine Translation. *CoRR*, abs/1612.06897.

Takaaki Hori, Yotaro Kubo, and Atsushi Nakamura. 2014. Real-time one-pass decoding with recurrent neural network language model for speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016a. Is Neural Machine Translation Ready for Deployment? A Case Study on 30 Translation Directions. In *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*, Seattle, WA.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016b. The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany. Association for Computational Linguistics.

Philipp Koehn, 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. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the 1st Workshop on Neural Machine Translation (and Generation) at ACL*. Association for Computational Linguistics.

Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister. 2016. LATTICERNN: Recurrent Neural Networks over Lattices. In *Interspeech*.

Xunying Liu, Xie Chen, Yongqiang Wang, Mark J. F. Gales, and Philip C. Woodland. 2016. Two Efficient Lattice Rescoring Methods Using Recurrent Neural Network Language Models. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 24(8).

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary Manipulation for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain. Association for Computational Linguistics.

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*, Berlin, Germany. Association for Computational Linguistics.

Felix Stahlberg, Adrià de Gispert, Eva Hasler, and Bill Byrne. 2017. Neural Machine Translation by Minimising the Bayes-risk with Respect to Syntactic Translation Lattices. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain. Association for Computational Linguistics.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically Guided Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017. Neural Machine Translation Advised by Statistical Machine Translation.