Marian: Fast Neural Machine Translation in C++

Marcin Junczys-Dowmunt[†] **Roman Grundkiewicz**[*‡] **Tomasz Dwojak**[*]
Hieu Hoang **Kenneth Heafield**[‡] **Tom Neckermann**[‡]
Frank Seide[†] **Ulrich Germann**[‡] **Alham Fikri Aji**[‡]
Nikolay Bogoychev[‡] **André F. T. Martins**[¶] **Alexandra Birch**[‡]

[†]Microsoft [*]Adam Mickiewicz University in Poznań
[‡]University of Edinburgh [¶]Unbabel

Abstract

We present Marian, an efficient and self-contained Neural Machine Translation framework with an integrated automatic differentiation engine based on dynamic computation graphs. Marian is written entirely in C++. We describe the design of the encoder-decoder framework and demonstrate that a research-friendly toolkit can achieve high training and translation speed.

1 Introduction

In this paper, we present Marian,[1] an efficient Neural Machine Translation framework written in pure C++ with minimal dependencies. It has mainly been developed at the Adam Mickiewicz University in Poznań and at the University of Edinburgh. It is currently being deployed in multiple European projects and is the main translation and training engine behind the neural MT launch at the World Intellectual Property Organization.[2]

In the evolving eco-system of open-source NMT toolkits, Marian occupies its own niche best characterized by two aspects:

- It is written completely in C++11 and intentionally does not provide Python bindings; model code and meta-algorithms are meant to be implemented in efficient C++ code.
- It is self-contained with its own back end, which provides reverse-mode automatic differentiation based on dynamic graphs.

Marian has minimal dependencies (only Boost and CUDA or a BLAS library) and enables barrier-free optimization at all levels: meta-algorithms such as MPI-based multi-node training, efficient batched beam search, compact implementations of new models, custom operators, and custom GPU kernels. Intel has contributed and is optimizing a CPU backend.

Marian grew out of a C++ re-implementation of Nematus (Sennrich et al., 2017b), and still maintains binary-compatibility for common models. Hence, we will compare speed mostly against Nematus. OpenNMT (Klein et al., 2017), perhaps one of the most popular toolkits, has been reported to have training speed competitive to Nematus.

Marian is distributed under the MIT license and available from `https://marian-nmt.github.io` or the GitHub repository `https://github.com/marian-nmt/marian`.

2 Design Outline

We will very briefly discuss the design of Marian. Technical details of the implementations will be provided in later work.

2.1 Custom Auto-Differentiation Engine

The deep-learning back-end included in Marian is based on reverse-mode auto-differentiation with dynamic computation graphs and among the established machine learning platforms most similar in design to DyNet (Neubig et al., 2017). While the back-end could be used for other tasks than machine translation, we choose to optimize specifically for this and similar use cases. Optimization on this level include for instance efficient implementations of various fused RNN cells, attention mechanisms or an atomic layer-normalization (Ba et al., 2016) operator.

[1] Named after Marian Rejewski, a Polish mathematician and cryptologist who reconstructed the German military Enigma cipher machine sight-unseen in 1932. `https://en.wikipedia.org/wiki/Marian_Rejewski`.

[2] `https://slator.com/technology/neural-conquers-patent-translation-in-major-wipo-roll-out/`

2.2 Extensible Encoder-Decoder Framework

Inspired by the stateful feature function framework in Moses (Koehn et al., 2007), we implement encoders and decoders as classes with the following (strongly simplified) interface:

```
class Encoder {
  EncoderState build(Batch);
};

class Decoder {
  DecoderState startState(EncoderState[]);
  DecoderState step(DecoderState, Batch);
};
```

A Bahdanau-style encoder-decoder model would implement the entire encoder inside `Encoder::build` based on the content of the batch and place the resulting encoder context inside the `EncoderState` object.

`Decoder::startState` receives a list of `EncoderState` (one in the case of the Bahdanau model, multiple for multi-source models, none for language models) and creates the initial `DecoderState`.

The `Decoder::step` function consumes the target part of a batch to produce the output logits of a model. The time dimension is either expanded by broadcasting of single tensors or by looping over the individual time-steps (for instance in the case of RNNs). Loops and other control structures are just the standard built-in C++ operations. The same function can then be used to expand over all given time steps at once during training and scoring or step-by-step during translation. Current hypotheses state (e.g. RNN vectors) and current logits are placed in the next `DecoderState` object.

Decoder states are used mostly during translation to select the next set of translation hypotheses. Complex encoder-decoder models can derive from `DecoderState` to implement non-standard selection behavior, for instance hard-attention models need to increase attention indices based on the top-scoring hypotheses.

This framework makes it possible to combine different encoders and decoders (e.g. RNN-based encoder with a Transformer decoder) and reduces implementation effort. In most cases it is enough to implement a single inference step in order to train, score and translate with a new model.

2.3 Efficient Meta-algorithms

On top of the auto-diff engine and encoder-decoder framework we implemented many efficient meta-algorithms. These include multi-device (GPU or CPU) training, scoring and batched beam search, ensembling of heterogeneous models (e.g. Deep RNN models and Transformer or language models), multi-node training and more.

3 Case Studies

In this section we will illustrate how we used the Marian toolkit to facilitate our own research across several NLP problems. Each subsection is meant as a showcase for different components of the toolkit and demonstrates the maturity and flexibility of the toolkit. Unless stated otherwise, all mentioned features are included in the Marian toolkit.

3.1 Improving over WMT2017 systems

Sennrich et al. (2017a) proposed the highest scoring NMT system in terms of BLEU during the WMT 2017 shared task on English-German news translation (Bojar et al., 2017a), trained with the Nematus toolkit (Sennrich et al., 2017b). In this section, we demonstrate that we can replicate and slightly outperform these results with an identical model architecture implemented in Marian and improve on the recipe with a Transformer-style (Vaswani et al., 2017) model.

3.1.1 Deep Transition RNN Architecture

The model architecture in Sennrich et al. (2017a) is a sequence-to-sequence model with single-layer RNNs in both, the encoder and decoder. The RNN in the encoder is bi-directional. Depth is achieved by building stacked GRU-blocks resulting in very tall RNN cells for every recurrent step (deep transitions). The encoder consists of four GRU-blocks per cell, the decoder of eight GRU-blocks with an attention mechanism placed between the first and second block. As in Sennrich et al. (2017a), embeddings size is 512, RNN state size is 1024. We use layer-normalization (Ba et al., 2016) and variational drop-out with $p = 0.1$ (Gal and Ghahramani, 2016) inside GRU-blocks and attention.

3.1.2 Transformer Architecture

We very closely follow the architecture described in Vaswani et al. (2017) and their ”base” model.

3.1.3 Training Recipe

Modeled after the description[3] from Sennrich et al. (2017a), we perform the following steps:

[3]The entire recipe is available in form of multiple scripts at https://github.com/marian-nmt/marian-examples.

System	test2016	test2017
UEdin WMT17 (single)	33.9	27.5
+Ensemble of 4	35.1	28.3
+R2L Reranking	36.2	28.3
Deep RNN (single)	34.3	27.7
+Ensemble of 4	35.3	28.2
+R2L Reranking	35.9	28.7
Transformer (single)	35.6	28.8
+Ensemble of 4	36.4	29.4
+R2L Reranking	36.8	29.5

Table 1: BLEU results for our replication of the UEdin WMT17 system for the en-de news translation task. We reproduced most steps and replaced the deep RNN model with a Transformer model.

- preprocessing of training data, tokenization, true-casing[4], vocabulary reduction to 36,000 joint BPE subword units (Sennrich et al., 2016) with a separate tool.[5]
- training of a shallow model for back-translation on parallel WMT17 data;
- translation of 10M German monolingual news sentences to English; concatenation of artificial training corpus with original data (times two) to produce new training data;
- training of four left-to-right (L2R) deep models (either RNN-based or Transformer-based);
- training of four additional deep models with right-to-left (R2L) orientation; [6]
- ensemble-decoding with four L2R models resulting in an n-best list of 12 hypotheses per input sentence;
- rescoring of n-best list with four R2L models, all model scores are weighted equally;
- evaluation on newstest-2016 (validation set) and newstest-2017 with sacreBLEU.[7]

We train the deep models with synchronous Adam on 8 NVIDIA Titan X Pascal GPUs with 12GB RAM for 7 epochs each. The back-translation model is trained with asynchronous Adam on 8 GPUs. We do not specify a batch size as Marian adjusts the batch based on available memory to maximize speed and memory usage. This guarantees that a chosen memory budget will not be exceeded during training.

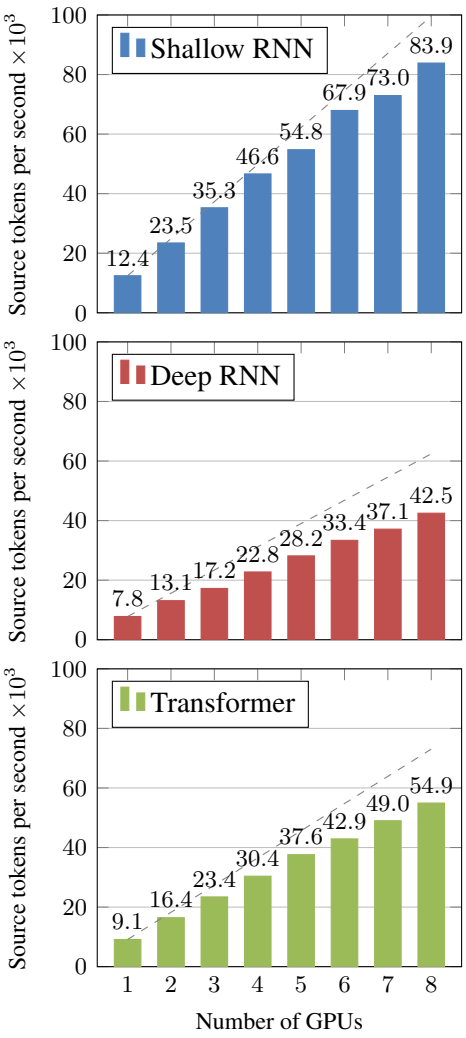

Figure 1: Training speed in thousands of source tokens per second for shallow RNN, deep RNN and Transformer model. Dashed line projects linear scale-up based on single-GPU performance.

All models use tied embeddings between source, target and output embeddings (Press and Wolf, 2017). Contrary to Sennrich et al. (2017a) or Vaswani et al. (2017), we do not average checkpoints, but maintain a continuously updated exponentially averaged model over the entire training. Following Vaswani et al. (2017), the learning rate is set to 0.0003 and decayed as the inverse square root of the number of updates after 16,000 updates. When training the transformer model, a linearly growing learning rate is used during the

[4]Proprocessing was performed using scripts from Moses (Koehn et al., 2007).

[5]https://github.com/rsennrich/subword-nmt

[6]R2L training, scoring or decoding does not require data processing, right-to-left inversion is built into Marian.

[7]https://github.com/mjpost/sacreBLEU

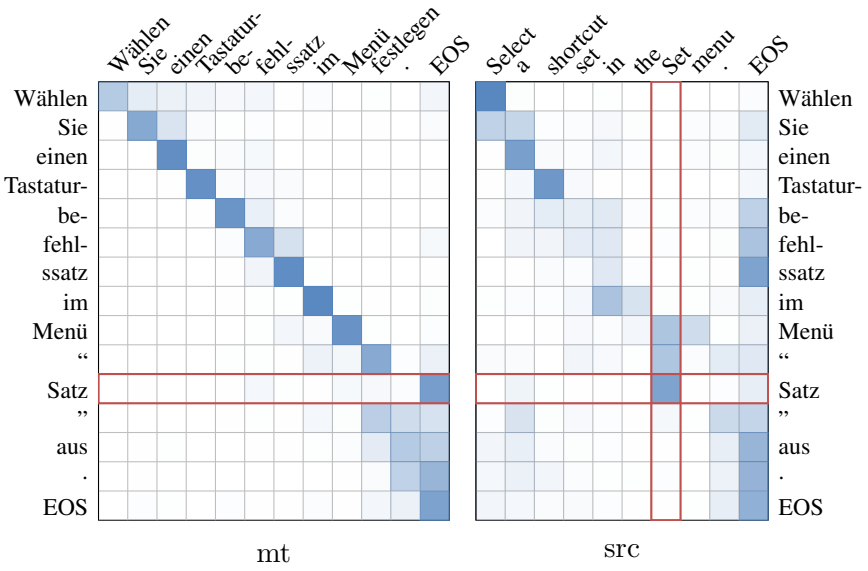

Figure 2: Example for error recovery based on dual attention. The missing word “Satz” could only be recovered based on the original source (marked in red) as it was dropped in the raw MT output.

Model	1	8	64
Shallow RNN	112.3	25.6	15.7
Deep Transition RNN	179.4	36.5	21.0
Transformer	362.7	98.5	71.3

Table 2: Translation time in seconds for newstest-2017 (3,004 sentences, 76,501 source BPE tokens) for different architectures and batch sizes.

first 16,000 iterations, starting with 0 until the base learning rate is reached.

3.1.4 Performance and Results

Quality. In terms of BLEU (Table 1), we match the original Nematus models from Sennrich et al. (2017a). Replacing the deep-transition RNN model with the transformer model results in a significant BLEU improvement of 1.2 BLEU on the WMT2017 test set.

Training speed. In Figure 1 we demonstrate the training speed as thousands of source tokens per second for the models trained in this recipe. All model types benefit from using more GPUs. Scaling is not linear (dashed lines), but close. The tokens-per-second rate (w/s) for Nematus on the same data on a single GPU is about 2800 w/s for the shallow model. Nematus does not have multi-GPU training. Marian achieves about 4 times faster training on a single GPU and about 30 times faster training on 8 GPUs for identical models.

Translation speed. The back-translation of 10M sentences with a shallow model takes about four hours on 8 GPUs at a speed of about 15,850 source tokens per second at a beam-size of 5 and a batch size of 64. Batches of sentences are translated in parallel on multiple GPUs.

In Table 2 we report the total number of seconds to translate newstest-2017 (3,004 sentences, 76,501 source BPE tokens) on a single GPU for different batch sizes. We omit model load time (usually below 10s). Beam size is 5.

3.2 State-of-the-art in Neural Automatic Post-Editing

In our submission to the Automatic Post-Editing shared task at WMT-2017 (Bojar et al., 2017b) and follow-up work (Junczys-Dowmunt and Grundkiewicz, 2017a,b), we explore multiple neural architectures adapted for the task of automatic post-editing of machine translation output as implementations in Marian. We focus on neural end-to-end models that combine both inputs mt (raw MT output) and src (source language input) in a single neural architecture, modeling $\{mt, src\} \rightarrow pe$ directly, where pe is post-edited corrected output.

These models are based on multi-source neural translation models introduced by Zoph and Knight (2016). Furthermore, we investigate the effect of hard-attention models or neural transductors (Aharoni and Goldberg, 2016) which seem to be well-suited for monolingual tasks, as well as combina-

tions of both ideas. Dual-attention models that are combined with hard attention remain competitive despite applying fewer changes to the input.

The encoder-decoder framework described in section 2.2, allowed to integrate dual encoders and hard-attention without changes to beam-search or ensembling mechanisms. The dual-attention mechanism over two encoders allowed to recover missing words that would not be recognized based on raw MT output alone, see Figure 2.

Our final system for the APE shared task scored second-best according to automatic metrics and best based on human evaluation.

3.3 State-of-the-art in Neural Grammatical Error Correction

In Junczys-Dowmunt and Grundkiewicz (2018), we use Marian for research on transferring methods from low-resource NMT on the ground of automatic grammatical error correction (GEC). Previously, neural methods in GEC did not reach state-of-the-art results compared to phrase-based SMT baselines. We successfully adapt several low-resource MT methods for GEC.

We propose a set of model-independent methods for neural GEC that can be easily applied in most GEC settings. The combined effects of these methods result in better than state-of-the-art neural GEC models that outperform previously best neural GEC systems by more than 8% M^2 on the CoNLL-2014 benchmark and more than 4.5% on the JFLEG test set. Non-neural state-of-the-art systems are matched on the CoNLL-2014 benchmark and outperformed by 2% on JFLEG.

Figure 3 illustrates these results on the CoNLL-2014 test set. To produce this graph, 40 GEC models (four per entry) and 24 language models (one per GEC model with pre-training) have been trained. The language models follow the decoder architecture and can be used for transfer learning, weighted decode-time ensembling and re-ranking. This also includes a Transformer-style language model with self-attention layers.

Proposed methods include extensions to Marian, such as source-side noise, a GEC-specific weighted training-objective, usage of pre-trained embeddings, transfer learning with pre-trained language models, decode-time ensembling of independently trained GEC models and language models, and various deep architectures.

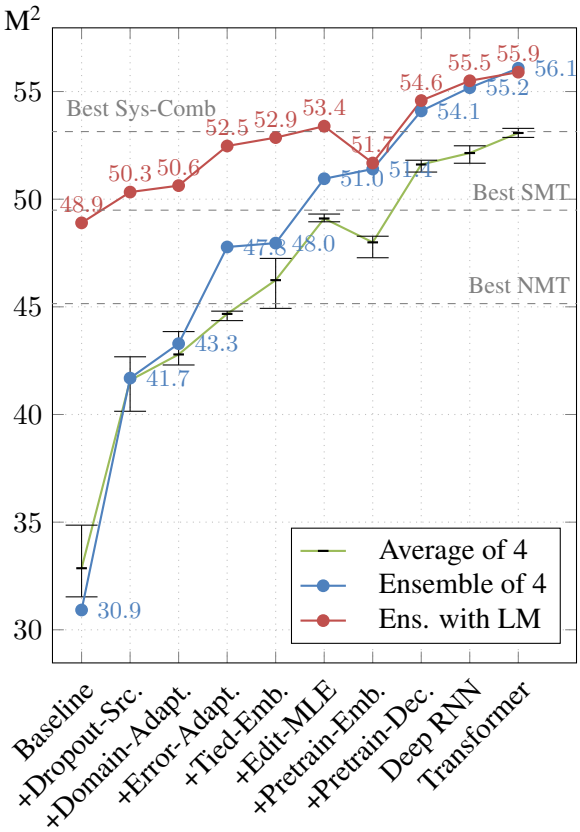

Figure 3: Comparison on the CoNLL-2014 test set for investigated methods.

4 Future Work and Conclusions

We introduced Marian, a self-contained neural machine translation toolkit written in C++ with focus on efficiency and research. Future work on Marian's back-end will look at faster CPU-bound computation, auto-batching mechanisms and automatic kernel fusion. On the front-end side we hope to keep up with future state-of-the-art models.

Acknowledgments

The development of Marian received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreements 688139 (SUMMA; 2016-2019), 645487 (Modern MT; 2015-2017), 644333 (TraMOOC; 2015-2017), 644402 (HimL; 2015-2017), the Amazon Academic Research Awards program, and the World Intellectual Property Organization. The CPU back-end was contributed by Intel under a partnership with the Alan Turing Institute.

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Roee Aharoni and Yoav Goldberg. 2016. Sequence to sequence transduction with hard monotonic attention. *arXiv preprint arXiv:1611.01487*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Ondrej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, and Julia Kreutzer, editors. 2017a. *Proc. of the 2nd Conference on Machine Translation, WMT 2017*. Association for Computational Linguistics.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017b. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017a. The AMU-UEdin submission to the WMT 2017 shared task on automatic post-editing. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 639–646, Copenhagen. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017b. An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 120–129. Asian Federation of Natural Language Processing.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of NAACL-HLT 2018*, New Orleans, USA. Association for Computational Linguistics. Accepted for publication.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72.

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 *ACL*. The Association for Computer Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: the dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017a. The University of Edinburgh's neural MT systems for WMT17. In (Bojar et al., 2017a), pages 389–399.

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. 2017b. 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*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *CoRR*, abs/1601.00710.