Pieces of Eight: 8-bit Neural Machine Translation*

Jerry Quinn **Miguel Ballesteros**
IBM Research,
1101 Kitchawan Road, Route 134 Yorktown Heights, NY 10598. U.S
jlquinn@us.ibm.com, miguel.ballesteros@ibm.com

Abstract

Neural machine translation has achieved levels of fluency and adequacy that would have been surprising a short time ago. Output quality is extremely relevant for industry purposes, however it is equally important to produce results in the shortest time possible, mainly for latency-sensitive applications and to control cloud hosting costs. In this paper we show the effectiveness of translating with 8-bit quantization for models that have been trained using 32-bit floating point values. Results show that 8-bit translation makes a non-negligible impact in terms of speed with no degradation in accuracy and adequacy.

1 Introduction

Neural machine translation (NMT) (Bahdanau et al., 2014; Sutskever et al., 2014) has recently achieved remarkable performance improving fluency and adequacy over phrase-based machine translation and is being deployed in commercial settings (Koehn and Knowles, 2017). However, this comes at a cost of slow decoding speeds compared to phrase-based and syntax-based SMT (see section 3).

NMT models are generally trained using 32-bit floating point values. At training time, multiple sentences can be processed in parallel leveraging graphical processing units (GPUs) to good advantage since the data is processed in batches. This is also true for decoding for non-interactive applications such as bulk document translation.

Why is fast execution on CPUs important? First, CPUs are cheaper than GPUs. Fast CPU computation will reduce commercial deployment costs. Second, for low-latency applications such as speech-to-speech translation (Neubig et al., 2017a), it is important to translate individual sentences quickly enough so that users can have an application experience that responds seamlessly. Translating individual sentences with NMT requires many memory bandwidth intensive matrix-vector or matrix-narrow matrix multiplications (Abdelfattah et al., 2016). In addition, the batch size is 1 and GPUs do not have a speed advantage over CPUs due to the lack of adequate parallel work (as evidenced by increasingly difficult batching scenarios in dynamic frameworks (Neubig et al., 2017b)).

Others have successfully used low precision approximations to neural net models. Vanhoucke et al. (2011) explored 8-bit quantization for feed-forward neural nets for speech recognition. Devlin (2017) explored 16-bit quantization for machine translation. In this paper we show the effectiveness of 8-bit decoding for models that have been trained using 32-bit floating point values. Results show that 8-bit decoding does not hurt the fluency or adequacy of the output, while producing results up to 4-6x times faster. In addition, implementation is straightforward and we can use the models as is without altering training.

The paper is organized as follows: Section 2 reviews the attentional model of translation to be sped up, Section 3 presents our 8-bit quantization in our implementation, Section 4 presents automatic measurements of speed and translation quality plus human evaluations, Section 5 discusses the results and some illustrative examples, Section 6 describes prior work, and Section 7 concludes the paper.

2 The Attentional Model of Translation

Our translation system implements the attentional model of translation (Bahdanau et al., 2014) consisting of an encoder-decoder network with an at-

* A piece of eight was a Spanish dollar that was divided into 8 reales, also known as Real de a Ocho.

Proceedings of NAACL-HLT 2018, pages 114–120
New Orleans, Louisiana, June 1 - 6, 2018. ©2017 Association for Computational Linguistics

tention mechanism.

The encoder uses a bidirectional GRU recurrent neural network (Cho et al., 2014) to encode a source sentence $\mathbf{x} = (x_1, ..., x_l)$, where x_i is the embedding vector for the ith word and l is the sentence length. The encoded form is a sequence of hidden states $\mathbf{h} = (h_1, ..., h_l)$ where each h_i is computed as follows

$$h_i = \begin{bmatrix} \overleftarrow{h_i} \\ \overrightarrow{h_i} \end{bmatrix} = \begin{bmatrix} \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}) \\ \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \end{bmatrix}, \quad (1)$$

where $\overrightarrow{h_0} = \overleftarrow{h_0} = 0$. Here $\overleftarrow{f}$ and $\overrightarrow{f}$ are GRU cells.

Given $\mathbf{h}$, the decoder predicts the target translation $\mathbf{y}$ by computing the output token sequence $(y_1, ...y_m)$, where m is the length of the sequence. At each time t, the probability of each token y_t from a target vocabulary is

$$p(y_t|\mathbf{h}, y_{t-1}..y_1) = g(s_t, y_{t-1}, H_t), \quad (2)$$

where g is a two layer feed-forward network over the embedding of the previous target word (y_{t-1}), the decoder hidden state (s_t), and the weighted sum of encoder states $\mathbf{h}$ (H_t), followed by a softmax to predict the probability distribution over the output vocabulary.

We compute s_t with a two layer GRU as

$$s'_t = r(s_{t-1}, y^*_{t-1}) \quad (3)$$

and

$$s_t = q(s'_t, H_t), \quad (4)$$

where s'_t is an intermediate state and $s_0 = \overleftarrow{h_0}$. The two GRU units r and q together with the attention constitute the conditional GRU layer of Sennrich et al. (2017). H_t is computed as

$$H_t = \begin{bmatrix} \sum_{i=1}^{l}(\alpha_{t,i} \cdot \overleftarrow{h}_i) \\ \sum_{i=1}^{l}(\alpha_{t,i} \cdot \overrightarrow{h}_i) \end{bmatrix}, \quad (5)$$

where $\alpha_{t,i}$ are the elements of α_t which is the output vector of the attention model. This is computed with a two layer feed-forward network

$$\alpha'_t = v(\tanh(w(h_i) + u(s'_{t-1}))), \quad (6)$$

where w and u are weight matrices, and v is another matrix resulting in one real value per encoder state h_i. α_t is then the softmax over α'_t.

We train our model using a program written using the Theano framework (Bastien et al., 2012). Generally models are trained with batch sizes ranging from 64 to 128 and unbiased Adam stochastic optimizer (Kingma and Ba, 2014). We use an embedding size of 620 and hidden layer sizes of 1000. We select model parameters according to the best BLEU score on a held-out development set over 10 epochs.

3 8-bit Translation

Our translation engine is a C++ implementation. The engine is implemented using the Eigen matrix library, which provides efficient matrix operations. Each CPU core translates a single sentence at a time. The same engine supports both batch and interactive applications, the latter making single-sentence translation latency important. We report speed numbers as both words per second (WPS) and words per core second (WPCS), which is WPS divided by the number of cores running. This gives us a measure of overall scaling across many cores and memory buses as well as the single-sentence speed.

Phrase-based SMT systems, such as (Tillmann, 2006), for English-German run at 170 words per core second (3400 words per second) on a 20 core Xeon 2690v2 system. Similarly, syntax-based SMT systems, such as (Zhao and Al-onaizan, 2008), for the same language pair run at 21.5 words per core second (430 words per second).

In contrast, our NMT system (described in Section 2) with 32-bit decoding runs at 6.5 words per core second (131 words per second). Our goal is to increase decoding speed for the NMT system to what can be achieved with phrase-based systems while maintaining the levels of fluency and adequacy that NMT offers.

Benchmarks of our NMT decoder unsurprisingly show matrix multiplication as the number one source of compute cycles. In Table 1 we see that more than 85% of computation is spent in Eigen's matrix and vector multiply routines (Eigen matrix vector product and Eigen matrix multiply). It dwarfs the costs of the transcendental function computations as well as the bias additions.

Given this distribution of computing time, it makes sense to try to accelerate the matrix operations as much as possible. One approach to increasing speed is to quantize matrix operations. Replacing 32-bit floating point math operations with 8-bit integer approximations in neural nets has been shown to give speedups and similar ac-

Time	Function
50.77%	Eigen matrix vector product
35.02%	Eigen matrix multiply
1.95%	NMT decoder layer
1.68%	Eigen fast tanh
1.35%	NMT tanh wrapper

Table 1: Profile before 8-bit conversion. More than 85% is spent in Eigen matrix/vector multiply routines.

curacy (Vanhoucke et al., 2011). We chose to apply similar optimization to our translation system, both to reduce memory traffic as well as increase parallelism in the CPU.

Our 8-bit matrix multiply routine uses a naive implementation with no blocking or copy. The code is implemented using Intel SSE4 vector instructions and computes 4 rows at a time, similar to (Devlin, 2017). Simplicity led to implementing 8-bit matrix multiplication with the results being placed into a 32-bit floating point result. This has the advantage of not needing to know the scale of the result. In addition, the output is a vector or narrow matrix, so little extra memory bandwidth is consumed.

Multilayer matrix multiply algorithms result in significantly faster performance than naive algorithms (Goto and Geijn, 2008). This is due to the fact that there are $O(N^3)$ math operations on $O(N^2)$ elements when multiplying NxN matrices, therefore it is worth significant effort to minimize memory operations while maximizing math operations. However, when multiplying an NxN matrix by an NxP matrix where P is very small ($<$10), memory operations dominate and performance does not benefit from the complex algorithm. When decoding single sentences, we typically set our beam size to a value less than 8 following standard practice in this kind of systems (Koehn and Knowles, 2017). We actually find that at such small values of P, the naive algorithm is a bit faster.

Time	Function
69.54%	8-bit matrix multiply
6.37%	Eigen fast tanh
2.06%	NMT decoder layer
0.95%	NMT tanh wrapper

Table 2: Profile after 8-bit conversion. Matrix multiply includes matrix-vector multiply. Matrix multiply is still 70% of computation. Tanh is larger but still relatively small.

Table 2 shows the profile after converting the matrix routines to 8-bit integer computation. There is only one entry for matrix-matrix and matrix-vector multiplies since they are handled by the same routine. After conversion, tanh and sigmoid still consume less than 7% of CPU time. We decided not to convert these operations to integer in light of that fact.

It is possible to replace all the operations with 8-bit approximations (Wu et al., 2016), but this makes implementation more complex, as the scale of the result of a matrix multiplication must be known to correctly output 8-bit numbers without dangerous loss of precision.

Assuming we have 2 matrices of size 1000x1000 with a range of values $[-10, 10]$, the individual dot products in the result could be as large as 10^8. In practice with neural nets, the scale of the result is similar to that of the input matrices. So if we scale the result to $[-127, 127]$ assuming the worst case, the loss of precision will give us a matrix full of zeros. The choices are to either scale the result of the matrix multiplication with a reasonable value, or to store the result as floating point. We opted for the latter.

8-bit computation achieves 32.3 words per core second (646 words per second), compared to the 6.5 words per core second (131 words per second) of the 32-bit system (both systems load parameters from the same model). This is even faster than the syntax-based system that runs at 21.5 words per core second (430 words per second). Table 3 summarizes running speeds for the phrase-based SMT system, syntax-based system and NMT with 32-bit decoding and 8-bit decoding.

System	WPCS
Phrase-based	170
Syntax-based	21.5
NMT 32-bit	6.5
NMT 8-bit	32.3

Table 3: Running speed (in words per core second) of the phrase-based SMT system, syntax-based system, NMT with 32-bit decoding and NMT with 8-bit decoding.

4 Measurements

To demonstrate the effectiveness of approximating the floating point math with 8-bit integer computation, we show automatic evaluation results

on several models, as well as independent human evaluations. We report results on Dutch-English, English-Dutch, Russian-English, German-English and English-German models. Table 4 shows training data sizes and vocabulary sizes. All models have 620 dimension embeddings and 1000 dimension hidden states.

Lang	Training Sentences	Source Vocabulary	Target Vocabulary
En-Nl	17M	42112	33658
Nl-En	17M	33658	42212
Ru-En	31M	42388	42840
En-De	31M	57867	63644
De-En	31M	63644	57867

Table 4: Model training data and vocabulary sizes

4.1 Automatic results

Here we report automatic results comparing decoding results on 32-bit and 8-bit implementations. As others have found (Wu et al., 2016), 8-bit implementations impact quality very little.

In Table 6, we compared automatic scores and speeds for Dutch-English, English-Dutch, Russian-English, German-English and English-German models on news data. The English-German model was run with both a single model (1x) and an ensemble of two models (2x) (Freitag et al., 2017). Table 5 gives the number of sentences and average sentence length for the test sets used.

Lang	Test Sentences	Src Sent Length	Tgt Sent Length
En-Nl	990	22.5	25.9
Nl-En	990	25.9	22.5
Ru-En	555	27.2	35.2
En-De	168	51.8	46.0
De-En	168	46.0	51.8

Table 5: Test data sizes and sentence lengths

Speed is reported in words per core second (WPCS). This gives us a better sense of the speed of individual engines when deployed on multi-core systems with all cores performing translations. Total throughput is simply the product of WPCS and the number of cores in the machine. The reported speed is the median of 9 runs to ensure consistent numbers. The results show that we see a 4-6x speedup over 32-bit floating point decoding. German-English shows the largest deficit for the 8-bit mode versus the 32-bit mode. The German-English test set only includes 168 sentences so this may be a spurious difference.

Lang	Mode	BLEU	Speed (WPSC)
En-Nl	32-bit	31.2	12.6
En-Nl	8-bit	31.2	58.9
Nl-En	32-bit	36.1	10.3
Nl-En	8-bit	36.3	45.8
Ru-En	32-bit	24.5	8.9
Ru-En	8-bit	24.3	51.4
De-En	32-bit	32.6	7.3
De-En	8-bit	32.2	37.5
En-De 2x	32-bit	30.5	7.1
En-De 2x	8-bit	30.6	33.7
En-De 1x	32-bit	29.7	15.9
En-De 1x	8-bit	29.7	71.3

Table 6: BLEU scores and speeds for 8-bit and 32-bit versions of several models. Speeds are reported in words per core second.

4.2 Human evaluation

These automatic results suggest that 8-bit quantization can be done without perceptible degradation. To confirm this, we carried out a human evaluation experiment.

In Table 7, we show the results of performing human evaluations on some of the same language pairs in the previous section. An independent native speaker of the language being translated to/from different than English (who is also proficient in English) scored 100 randomly selected sentences. The sentences were shuffled during the evaluation to avoid evaluator bias towards different runs. We employ a scale from 0 to 5, with 0 being unintelligible and 5 being perfect translation.

Language	32-bit	8-bit
En-Nl	4.02	4.08
Nl-En	4.03	4.03
Ru-En	4.10	4.06
En-De 2x	4.05	4.16
En-De 1x	3.84	3.90

Table 7: Human evaluation scores for 8-bit and 32-bit systems. All tests are news domain.

The Table shows that the automatic scores shown in the previous section are also sustained

Source	Time	Sie standen seit 1946 an der Parteispitze
32-bit	720 ms	They **had been** at the **party leadership** since 1946
8-bit	180 ms	They **stood** at the **top of the party** since 1946.
Source	Time	So erwarten die Experten für dieses Jahr lediglich einen Anstieg der Weltproduktion von 3,7 statt der im Juni prognostizierten 3,9 Prozent. Für 2009 sagt das Kieler Institut sogar eine Abschwächung auf 3,3 statt 3,7 Prozent voraus.
32-bit	4440 ms	For this year, the experts expect only an increase in world production of 3.7 instead of the 3.9 percent forecast in June. In 2009, the Kiel Institute **predictated** a slowdown to **3.3 percent** instead of 3.7 **percent**.
8-bit	750 ms	For this year, the experts expect only an increase in world production of 3.7 instead of the 3.9 percent forecast in June. In 2009, the Kiel Institute **even forecast** a slowdown to **3.3%** instead of 3.7 **per cent**.
Source	Time	Heftige Regenfälle wegen "Ike" werden möglicherweise schwerere Schäden anrichten als seine Windböen. Besonders gefährdet sind dicht besiedelte Gebiete im Tal des Rio Grande, die noch immer unter den Folgen des Hurrikans "Dolly" im Juli leiden.
32-bit	6150 ms	Heavy rainfall due to "Ike" may cause **more severe** damage than its gusts of wind, particularly in densely populated areas in the Rio Grande valley, which **are still suffering** from the consequences of the "dolly" hurricane in July.
8-bit	1050 ms	Heavy rainfall due to "Ike" may cause **heavier** damage than its gusts of wind, particularly in densely populated areas in the Rio Grande valley, which **still suffer** from the consequences of the "dolly" hurricane in July.

Table 8: Examples of De-En news translation system comparing 32-bit and 8-bit decoding. Differences are in boldface. Sentence times are average of 10 runs.

Source	Time	Het is tijd om de kloof te overbruggen.
32-bit	730 ms	**It's** time to bridge the gap.
8-bit	180 ms	**It is** time to bridge the gap.
Source	Time	Niet dat Barientos met zijn vader van plaats zou willen wisselen.
32-bit	1120 ms	Not that Barientos would **want** to **change his father's place**.
8-bit	290 ms	Not that Barientos would **like** to **switch places with his father**.

Table 9: Examples of Nl-En news translation system comparing 32-bit and 8-bit decoding. Differences are in boldface. Sentence times are average of 10 runs.

by humans. 8-bit decoding is as good as 32-bit decoding according to the human evaluators.

5 Discussion

Having a faster NMT engine with no loss of accuracy is commercially useful. In our deployment scenarios, it is the difference between an interactive user experience that is sluggish and one that is not. Even in batch mode operation, the same throughput can be delivered with 1/4 the hardware.

In addition, this speedup makes it practical to deploy small ensembles of models. As shown above in the En-De model in Table 6, an ensemble can deliver higher accuracy at the cost of a 2x slowdown. This work makes it possible to translate with higher quality while still being at least twice as fast as the previous baseline.

As the numbers reported in Section 4 demonstrate, 8-bit and 32-bit decoding have similar average quality. As expected, the outputs produced by the two decoders are not identical. In fact, on a run of 166 sentences of De-En translation, only 51 were identical between the two. In addition, our human evaluation results and the automatic scoring suggest that there is no specific degradation by the 8-bit decoder compared to the 32-bit decoder. In order to emphasize these claims, Table 8 shows several examples of output from the two systems for a German-English system. Table 9 shows 2 more examples from a Dutch-English system.

In general, there are minor differences without any loss in adequacy or fluency due to 8-bit decoding. Sentence 2 in Table 8 shows a spelling error ("predictated") in the 32-bit output due to re-

assembly of incorrect subword units.[1]

6 Related Work

Reducing the resources required for decoding neural nets in general and neural machine translation in particular has been the focus of some attention in recent years.

Vanhoucke et al. (2011) explored accelerating convolutional neural nets with 8-bit integer decoding for speech recognition. They demonstrated that low precision computation could be used with no significant loss of accuracy. Han et al. (2015) investigated highly compressing image classification neural networks using network pruning, quantization, and Huffman coding so as to fit completely into on-chip cache, seeing significant improvements in speed and energy efficiency while keeping accuracy losses small.

Focusing on machine translation, Devlin (2017) implemented 16-bit fixed-point integer math to speed up matrix multiplication operations, seeing a 2.59x improvement. They show competitive BLEU scores on WMT English-French NewsTest2014 while offering significant speedup. Similarly, (Wu et al., 2016) applies 8-bit end-to-end quantization in translation models. They also show that automatic metrics do not suffer as a result. In this work, quantization requires modification to model training to limit the size of matrix outputs.

7 Conclusions and Future Work

In this paper, we show that 8-bit decoding for neural machine translation runs up to 4-6x times faster than a similar optimized floating point implementation. We show that the quality of this approximation is similar to that of the 32-bit version. We also show that it is unnecessary to modify the training procedure to produce models compatible with 8-bit decoding.

To conclude, this paper shows that 8-bit decoding is as good as 32-bit decoding both in automatic measures and from a human perception perspective, while it improves latency substantially.

In the future we plan to implement a multi-layered matrix multiplication that falls back to the naive algorithm for matrix-panel multiplications. This will provide speed for batch decoding for applications that can take advantage of it. We also plan to explore training with low precision for faster experiment turnaround time.

Our results offer hints of improved accuracy rather than just parity. Other work has used training as part of the compression process. We would like to see if training quantized models changes the results for better or worse.

[1] In order to limit the vocabulary, we use BPE subword units (Sennrich et al., 2016) in all models.

References

Ahmad Abdelfattah, David Keyes, and Hatem Ltaief. 2016. Kblas: An optimized library for dense matrix-vector multiplication on gpu accelerators. *ACM Trans. Math. Softw.* 42(3):18:1–18:31. https://doi.org/10.1145/2818311.

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

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR* abs/1409.1259.

Jacob Devlin. 2017. Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the cpu. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2820–2825.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *CoRR* abs/1702.01802.

Kazushige Goto and Robert A. van de Geijn. 2008. Anatomy of high-performance matrix multiplication. *ACM Trans. Math. Softw.* 34(3):12:1–12:25.

Song Han, Huizi Mao, and William J. Dally. 2015. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR* abs/1510.00149.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, Vancouver, pages 28–39.

Graham Neubig, Kyunghyun Cho, Jiatao Gu, and Victor O. K. Li. 2017a. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 1053–1062.

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. 2017b. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

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*. Association for Computational Linguistics, Valencia, Spain, pages 65–68. http://aclweb.org/anthology/E17-3017.

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)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112.

Christoph Tillmann. 2006. Efficient dynamic programming search algorithms for phrase-based smt. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, CHSLP '06, pages 9–16.

Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. 2011. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Bing Zhao and Yaser Al-onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '08, pages 572–581. http://dl.acm.org/citation.cfm?id=1613715.1613785.