Montreal Neural Machine Translation Systems for WMT'15

Sébastien Jean⋆
University of Montreal
jeasebas@iro.umontreal.ca

Orhan Firat⋆
Middle East Technical University, Turkey
orhan.firat@ceng.metu.edu.tr

Kyunghyun Cho
University of Montreal

Roland Memisevic
University of Montreal

Yoshua Bengio
University of Montreal
CIFAR Senior Fellow

firstname.lastname@umontreal.ca

Abstract

Neural machine translation (NMT) systems have recently achieved results comparable to the state of the art on a few translation tasks, including English→French and English→German. The main purpose of the Montreal Institute for Learning Algorithms (MILA) submission to WMT'15 is to evaluate this new approach on a greater variety of language pairs. Furthermore, the human evaluation campaign may help us and the research community to better understand the behaviour of our systems. We use the RNNsearch architecture, which adds an attention mechanism to the encoder-decoder. We also leverage some of the recent developments in NMT, including the use of large vocabularies, unknown word replacement and, to a limited degree, the inclusion of monolingual language models.

1 Introduction

Neural machine translation (NMT) is a recently proposed approach for machine translation that relies only on neural networks. The NMT system is trained end-to-end to maximize the conditional probability of a correct translation given a source sentence (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). Although NMT has only recently been introduced, its performance has been found to be comparable to the state-of-the-art statistical machine translation (SMT) systems on a number of translation tasks (Luong et al., 2015; Jean et al., 2015). The main purpose of our submission to WMT'15 is to test the NMT system on a greater variety of language pairs. As such, we trained systems on Czech↔English, German↔English and Finnish→English. Furthermore, the human evaluation campaign of WMT'15 will help us better understand the quality of NMT systems which have mainly been evaluated using the automatic evaluation metric such as BLEU (Papineni et al., 2002).

Most NMT systems are based on the *encoder-decoder* architecture (Cho et al., 2014; Sutskever et al., 2014; Kalchbrenner and Blunsom, 2013). The source sentence is first read by the encoder, which compresses it into a real-valued vector. From this vector representation the decoder may then generate a translation word-by-word. One limitation of this approach is that a source sentence of any length must be encoded into a fixed-length vector. To address this issue, our systems for WMT'15 use the *RNNsearch* architecture from (Bahdanau et al., 2015). In this case, the encoder assigns a context-dependent vector, or annotation, to every source word. The decoder then selectively combines the most relevant annotations to generate each target word.

NMT systems often use a limited vocabulary of approximately $30,000$ to $80,000$ target words, which leads them to generate many out-of-vocabulary tokens ($\langle$UNK$\rangle$). This may easily lead to the degraded quality of the translations. To sidestep this problem, we employ a variant of importance sampling to help increase the target vocabulary size (Jean et al., 2015). Even with a larger vocabulary, there will almost assuredly be words in the test set that were unseen during training. As such, we replace generated out-of-vocaulbary tokens with the corresponding source words with a technique similar to those proposed by (Luong et al., 2015).

Most NMT systems rely only on parallel data, ignoring the wealth of information found in large monolingual corpora. On Finnish→English, we combine our systems with a recurrent neural net-

⋆ equal contribution

Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 134–140,
Lisboa, Portugal, 17-18 September 2015.

work (RNN) language model by recently proposed *deep fusion* (Gülçehre et al., 2015). For the other language pairs, we tried reranking n-best lists with 5-gram language models (Chen and Goodman., 1998).

2 System Description

In this section, we describe the RNNsearch architecture as well as the additional techniques we used.

Mathematical Notations Capital letters are used for matrices, and lower-case letters for vectors and scalars. x and y are used for a word in source and target sentences, respectively. We boldface them into $\mathbf{x}$, $\mathbf{y}$ and $\hat{\mathbf{y}}$ to denote their continuous-space representation (word embeddings).

2.1 Bidirectional Encoder

To encode a source sentence $(x_1, \ldots, x_{T_x})$ of length T_x into a sequence of annotations, we use a bidirectional recurrent neural network (Schuster and Paliwal, 1997). The bidirectional recurrent neural network (BiRNN) consists of two recurrent neural networks (RNN) that read the sentence either forward (from left to right) or backward. These RNNs respectively compute the sequences of hidden states $(\overrightarrow{h}_1, \ldots, \overrightarrow{h}_{T_x})$ and $(\overleftarrow{h}_1, \ldots, \overleftarrow{h}_{T_x})$. These two sequences are concatenated at each time step to form the annotations $(h_1, \ldots, h_{T_x})$. Each annotation h_i summarizes the entire sentence, albeit with more emphasis on word x_i and the neighbouring words.

We built the BiRNN with gated recurrent units (GRU, (Cho et al., 2014)), although long short-term memory (LSTM) units could also be used (Hochreiter and Schmidhuber, 1997), as in (Sutskever et al., 2014). More precisely, for the forward RNN, the hidden state at the i-th word is computed as

$$\overrightarrow{h}_i = \begin{cases} (1 - \overrightarrow{z}_i) \odot \overrightarrow{h}_{i-1} + \overrightarrow{z}_i \odot \overrightarrow{\underline{h}}_i & \text{, if } i > 0 \\ 0 & \text{, if } i = 0 \end{cases}$$

where

$$\begin{aligned} \overrightarrow{\underline{h}}_i &= \tanh\left(\overrightarrow{W}\mathbf{x}_i + \overrightarrow{U}\left[\overrightarrow{r}_i \odot \overrightarrow{h}_{i-1}\right] + \overrightarrow{b}\right) \\ \overrightarrow{z}_i &= \sigma\left(\overrightarrow{W}_z\mathbf{x}_i + \overrightarrow{U}_z\overrightarrow{h}_{i-1}\right) \\ \overrightarrow{r}_i &= \sigma\left(\overrightarrow{W}_r\mathbf{x}_i + \overrightarrow{U}_r\overrightarrow{h}_{i-1}\right). \end{aligned}$$

To form the new hidden state, the network first computes a proposal $\overrightarrow{\underline{h}}_i$. This is then additively combined with the previous hidden state $\overrightarrow{h}_{i-1}$, and this combination is controlled by the update gate $\overrightarrow{z}_i$. Such gated units facilitate capturing long-term dependencies.

2.2 Attentive Decoder

After computing the initial hidden state $s_0 = \tanh\left(W_s\overleftarrow{h}_1\right) + b_s$, the *RNNsearch* decoder alternates between three steps: *Look*, *Generate* and *Update*.

During the *Look* phase, the network determines which parts of the source sentence are most relevant. Given the previous hidden state s_{i-1} of the decoder recurrent neural network (RNN), each annotation h_j is assigned a score e_{ij}:

$$e_{ij} = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right).$$

Although a more complex scoring function can potentially learn more non-trivial alignments, we observed that this single-hidden-layer function is enough for most of the language pairs we considered.

These scores e_{ij} are then normalized to sum to 1:

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x}\exp\left(e_{ik}\right)}, \quad (1)$$

which we call alignment weights.

The context vector c_i is computed as a weighted sum of the annotations $(h_1, ..., h_{T_x})$ according to the alignment weights:

$$c_i = \sum_{j=1}^{T_x}\alpha_{ij}h_j.$$

This formulation allows the annotations with higher alignment weights to be more represented in the context vector c_i.

In the *Generate* phase, the decoder predicts the next target word. We first combine the previous hidden state s_{i-1}, the previous word y_{i-1} and the current context vector c_i into a vector $\tilde{t}_i$:

$$\tilde{t}_i = U_o s_{i-1} + V_o\mathbf{y}_{i-1} + C_o c_i + b_o.$$

We then transform $\tilde{t}_i$ into a hidden state m_i with an arbitrary feedforward network. In our submission, we apply the maxout non-linearity (Goodfellow et al., 2013) to $\tilde{t}_i$, followed by an affine transformation.

Phase	Output ← Input
Look	$c_i \leftarrow s_{i-1}, (h_1, ..., h_{T_x})$
Generate	$y_i \leftarrow s_{i-1}, y_{i-1}, c_i$
Update	$s_i \leftarrow s_{i-1}, y_i, c_i$

Table 1: Summary of *RNNsearch* decoder phases

For a target vocabulary V, the probability of word y_i is then

$$p(y_i|s_{i-1}, y_{i-1}, c_i) = \frac{\exp\left(\hat{\mathbf{y}}_i^\top m_i + b_{y_i}\right)}{\sum_{y \in V} \exp\left(\hat{\mathbf{y}}^\top m_i + b_y\right)}. \quad (2)$$

Finally, in the *Update* phase, the decoder computes the next recurrent hidden state s_i from the context c_i, the generated word y_i and the previous hidden state s_{i-1}. As with the encoder we use gated recurrent units (GRU).

Table 1 summarizes this three-step procedure. We observed that it is important to have *Update* to follow *Generate*. Otherwise, the next step's *Look* would not be able to resolve the uncertainty embedded in the previous hidden state about the previously generated word.

2.3 Very Large Target Vocabulary Extension

Training an RNNsearch model with hundreds of thousands of target words easily becomes prohibitively time-consuming due to the normalization constant in the softmax output (see Eq. (2).) To address this problem, we use the approach presented in (Jean et al., 2015), which is based on importance sampling (Bengio and Sénécal, 2008). During training, we choose a smaller vocabulary size τ and divide the training set into partitions, each of which contains approximately τ unique target words. For each partition, we train the model as if only the unique words within it existed, leaving the embeddings of all the other words fixed.

At test time, the corresponding subset of target words for each source sentence is not known in advance, yet we still want to keep computational complexity manageable. To overcome this, we run an existing word alignment tool on the training corpus in advance to obtain word-based conditional probabilities (Brown et al., 1993). During decoding, we start with an initial target vocabulary containing the K most frequent words. Then, reading a few sentences at once, we arbitrarily replace some of these initial words by the K' most likely ones for each source word.[1]

No matter how large the target vocabulary is, there will almost always be those words, such as proper names or numbers, that will appear only in the development or test set, but not during training. To handle this difficulty, we replace unknown words in a manner similar to (Luong et al., 2015). More precisely, for every predicted out-of-vocabulary token ($\langle$UNK$\rangle$), we determine its most likely origin by choosing the source word with the largest alignment weight α_{ij} (see Eq. (1).) We may then replace $\langle$UNK$\rangle$ by either the most likely word according to a dictionary, or simply by the source word itself. Depending on the language pairs, we used different heuristics according to performance on the development set.

2.4 Integrating Language Models

Unlike some data-rich language pairs, most of the translation tasks do not have enough parallel text to train end-to-end machine translation systems. To overcome with this issue of low-resource language pairs, external monolingual corpora is exploited by using the method of *deep fusion* (Gülçehre et al., 2015).

In addition to the RNNsearch model, we train a separate language model (LM) with a large monolingual corpus. Then, the trained LM is plugged into the decoder of the trained RNNsearch with an additional controller network which modulates the contributions from the RNNsearch and LM. The controller network takes as input the hidden state of the LM, and optionally RNNsearch's hidden state, and outputs a scalar value in the range $[0, 1]$. This value is multiplied to the LM's hidden state, controlling the amount of information coming from the LM. The combined model, the RNNsearch, the LM and the controller network, is jointly tuned as the final translation model for a low-resource pair.

In our submission, we used recurrent neural network language model (RNNLM). More specifically, let s_i^{LM} be the hidden state of a pre-trained RNNLM and s_i^{TM} be that of a pre-trained RNNsearch at time i. The controller network is defined as

$$g_t = \sigma\left(V_g^\top s_t^{\text{LM}} + W_g^\top s_t^{\text{TM}} + b_g\right),$$

[1] This step differs very slightly from (Jean et al., 2015), where the sentence-specific words were added on top of the K common ones instead of replacing them.

where σ is a logistic sigmoid function, v_g, w_g and b_g are model parameters. The output of the controller network is multiplied to the LM's hidden state s_i^{LM}:

$$p_t^{\text{LM}} = s_t^{\text{LM}} \odot g_t.$$

The *Generate* phase in Sec. 2.2 is updated as,

$$\tilde{t}_i = U_o^{\text{TM}} s_{i-1}^{\text{TM}} + U_o^{\text{LM}} p_{t-1}^{\text{LM}} + V_o \mathbf{y}_{i-1} + C_o c_i + b_o.$$

This lets the decoder fully use the signal from the translation model, while the the signal from the LM is modulated by the controller output.

Among all the pairs of languages in WMT'15, Finnish↔English translation has the least amount of parallel text, having approximately $2M$ aligned sentences only. Thus, we use the *deep fusion* for the Fi-En in the official submission. However, we further experimented German→English, having the second least parallel text, and Czech→English, which has comparably larger data. We include the results from these two language pairs here for completeness.

3 Experimental Details

We now describe the settings of our experiments. Except for minor differences, all the settings were similar across all the considered language pairs.

3.1 Data

All the systems, except for the English→German (En→De) system, were built using all the data made available for WMT'15. The En→De system, which was showcased in (Jean et al., 2015), was built earlier than the others, using only the data from the last year's workshop (WMT'14.)

Each corpus was tokenized, but neither lowercased nor truecased. We avoided badly aligned sentence pairs by removing any source-target sentence pair with a large mismatch between their lengths. Furthermore, we removed sentences that were likely written in an incorrect language, either with a simple heuristic for En→De, or with a publicly available toolkit for the other language pairs (Shuyo, 2010). In order to limit the memory use during training, we only trained the systems with sentences of length up to 50 words only. Finally, for some but not all models, we reshuffled the data a few times and concatenated the different segments before training.

In the case of German (De) source, we performed compound splitting (Koehn and Knight, 2003), as implemented in the Moses toolkit (Koehn et al., 2007). For Finnish (Fi), we used Morfessor 2.0 for morpheme segmentation (Virpioja et al., 2013) by using the default parameters.

An Issue with Apostrophes In the training data, apostrophes appear in many forms, such as a straight vertical line (U+0027) or as a right single quotation mark (U+0019). The use of, for instance, the normalize-punctuation script[2] could have helped, but we did not use it in our experiments. Consequently, we encountered an issue of the tokenizer from the Moses toolkit not applying the same rule for both kinds of apostrophes. We fixed this issue in time for Czech→English (Cs→En), but all the other systems were affected to some degree, in particular, the system for De→En.

3.2 Settings

We used the RNNsearch models of size identical to those presented in (Bahdanau et al., 2015; Jean et al., 2015). More specifically, all the words in both target and source vocabularies were projected into a 620-dimensional vector space. Each recurrent neural network (RNN) had a 1000-dimensional hidden state. The models were trained with Adadelta (Zeiler, 2012), and the norm of the gradient at each update was rescaled (Pascanu et al., 2013). For the language pairs other than Cs→En and Fi→En, we held the word embeddings fixed near the end of training, as described in (Jean et al., 2015).

With the very large target vocabulary technique in Sec. 2.3, we used 500K source and target words for the En→De system, while 200K source and target words were used for the De→En and Cs↔En systems.[3] During training we set τ between 15K and 50K, depending on the hardware availability. As for decoding, we mostly used $K = 30,000$ and $K' = 10$.

Given the small sizes of the Fi→En corpora, we simply used a fixed vocabulary size of 40K tokens to avoid any adverse effect of including every unique target word in the vocabulary. The inclusion of every unique word would prevent the network from decoding out $\langle$UNK$\rangle$ at all, even if

[2] `http://www.statmt.org/wmt11/normalize-punctuation.perl`

[3] This choice was made mainly to cope with the limited storage availability.

Language pair	BLEU-c		BLEU-c ranking		Human ranking
	single	ensemble	constrained	unconstrained	
En→Cs	15.7	18.3	1/6	2/7	4/8
En→De	22.4	24.8	1/11	1/13	1-2/16
Cs→En	20.2	23.3	3/6	3/6	3-4/7
De→En	25.6	27.6	6/9	6/10	6-7/13
Fi→En	10.1	13.6	7/9	9/12	10/14

Table 2: Results on the official WMT'15 test sets for single models and primary ensemble submissions. All our own systems are constrained. When ranking by BLEU, we only count one system from each submitter. Human rankings include all primary and online systems, but exclude those used in the Cs↔En tuning task.

out-of-vocabulary words will assuredly appear in the test set.

For each language pair, we trained a total of four independent models that differed in parameter initialization and data shuffling, monitoring the training progress on either *newstest2012+2013*, *newstest2013* or *newsdevs2015*.[4] Translations were generated by beam search, with a beam width of 20, trying to find the sentence with the highest log-probability (single model), or highest average log-probability over all models (ensemble), divided by the sentence length (Boulanger-Lewandowski et al., 2013). This length normalization addresses the tendency of the recurrent neural network to output shorter sentences.

For Fi→En, we augmented models by *deep fusion* with an RNN-LM. The RNN-LM, which was built using the LSTM units, was trained on the English Gigaword corpus using the vocabulary comprising of the 42K most frequent words in the English side of the intersection of the parallel corpora of Fi→En, De→En and Cs→En. Importantly, we use the same RNN-LM for both Fi→En, Cs→En and De→En. In the experiments with deep fusion, we used the randomly selected $2/3$ of *newsdev2015* as a validation set and the rest as a held-out set. In the case of De→En, we used $newstest2013$ for validation and $newstest2014$ for test.

For all language pairs except Fi→En, we also simply built 5-gram language models, this time on all appropriate provided data, with the exception of the English Gigaword (Heafield, 2011). In our contrastive submissions only, we re-ranked our 20-best lists with the LM log-probabilities, once again divided by sentence length. The relative weight of the language model was manually chosen to maximize BLEU on the development set.

4 Results

Results for single systems and primary ensemble submissions are presented in Table 2.[5] When translating from English to another language, neural machine translation works particularly well, achieving the best BLEU-c scores among all the constrained systems. On the other hand, NMT is generally competitive even in the case of translating to English, but it not yet as good as well as the best SMT systems according to BLEU. If we rather rely on human judgement instead of automated metrics, the NMT systems still perform quite well over many language pairs, although they are in some instances surpassed by other statistical systems that have slightly lower BLEU scores.

In our contrastive submissions for Cs↔En and De↔En where we re-ranked 20-best lists with a 5-gram language model, BLEU scores went up modestly by 0.1 to 0.5 BLEU, but interestingly translation error rate (TER) always worsened. One possible drawback about the manner we integrated language models here is the lack of translation models in the reverse direction, meaning we do not implicitely leverage the Bayes' rule as most other translation systems do.

In our further experiments, which are not part of our WMT'15 submission, for single models we observed the improvements of approximately $1.0/0.5$ BLEU points for *dev/test* in $\{$Cs,De$\}$→En tasks, when we employ *deep fusion* for incorporating language models.[6]

[4] For En→De, we created eight semi-independent models. See (Jean et al., 2015) for more details.

[5] Also available at `http://matrix.statmt.org/matrix/`

[6] Improvements are for single models only. See (Gülçehre et al., 2015) for more details.

5 Conclusion

We presented the MILA neural machine translation (NMT) systems for WMT'15, using the encoder–decoder model with the attention mechanism (Bahdanau et al., 2015) and the recent developments in NMT (Jean et al., 2015; Gülçehre et al., 2015). We observed that the NMT systems are now competitive against the conventional SMT systems, ranking first by BLEU among the constrained submission on both the En→Cs and En→De tasks. In the future, more analysis is needed on the influence of the source and target languages for neural machine translation. For instance, it would be interesting to better understand why performance relative to other approaches was somewhat weaker when translating into English, or how the amount of reordering influences the translation quality of neural MT systems.

Acknowledgments

The authors would like to thank the developers of Theano (Bergstra et al., 2010; Bastien et al., 2012). We thank Kelvin Xu, Bart van Mërrienboer, Dzmitry Bahdanau and Étienne Simon for their help. We also thank the two anonymous reviewers and everyone who contributed to the manual evaluation campaign. We acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs, CIFAR, Samsung and TUBITAK.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR'2015, arXiv: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.

Yoshua Bengio and Jean-Sébastien Sénécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, 19(4):713–722.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *ISMIR*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Stanley F. Chen and Joshua T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, October.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1319–1327.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 187–193. 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, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Nakatani Shuyo. 2010. Language detection library for java.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS'2014*.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Matthew D Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv:*`1212.5701 [cs.LG]`.