PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 108 JUNE 2017 37–48

Convolutional over Recurrent Encoder for Neural Machine Translation

Praveen Dakwale, Christof Monz

Informatics Institute, University of Amsterdam

Abstract

Neural machine translation is a recently proposed approach which has shown competitive results to traditional MT approaches. Standard neural MT is an end-to-end neural network where the source sentence is encoded by a recurrent neural network (RNN) called encoder and the target words are predicted using another RNN known as decoder. Recently, various models have been proposed which replace the RNN encoder with a convolutional neural network (CNN). In this paper, we propose to augment the standard RNN encoder in NMT with additional convolutional layers in order to capture wider context in the encoder output. Experiments on English to German translation demonstrate that our approach can achieve significant improvements over a standard RNN-based baseline.

1. Introduction

Recently proposed neural machine translation (NMT) has shown competitive results over traditional MT approaches such as Phrase-based MT. The most successful of these neural network approaches is the encoder-decoder framework of Bahdanau et al. (2015) in which the source sentence is converted into a vector representation by a recurrent neural network (RNN) called encoder, then another RNN called decoder generates a target sentence word by word based on the source representation and target history. Besides Machine translation, RNNs have shown promising results in modelling various other NLP tasks such as language modelling (Mikolov et al., 2010) and text similarity (Mueller and Thyagarajan, 2016). The strength of using RNNs for language processing lies in their ability to recurrently maintain a history for large input sequences, thus capturing the long distance dependencies which is an important occurrence in natural language texts.

 Corresponding author: p.dakwale@uva.nl
Cite as: Praveen Dakwale, Christof Monz. Convolutional over Recurrent Encoder for Neural Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 108, 2017, pp. 37–48. doi: 10.1515/pralin-2017-0007.

Although, modelling sequences using the recurrence property is important for most NLP tasks, there is a critical limitation in relying solely on the strengths of the RNN. In an RNN, at each timestep the encoder output is a global representation in which the information about the current word and the previous history are represented compositely. Although RNNs effectively model interdependence of words, they cannot capture phrases without prefix context and often capture too much of last words in the final vector.

To overcome the problem of compact fixed length vectors in neural MT, Bahdanau et al. (2015) and Luong et al. (2015) proposed an attention mechanism which is a very effective approach to solve this problem by representing the source sentence as a weighted average of the encoder outputs corresponding to each source word.

In this paper, we propose to modify the RNN encoder-decoder framework by adding multiple convolutional layers on top of the RNN output. Since the convolutional neural networks (CNNs) apply to a fixed-size window of the input sentence, at each layer, each output represents a relatively uniform composition of information from multiple words. This provides effective guidance to the network to focus on the relevant parts of the source sentence. At the same time, sequence to sequence modelling as in RNNs is necessary to capture the long-distance dependencies between the segments of the source sentence itself. Thus, in our model, a convolutional encoder complements the standard RNN encoder. Such a combination of RNN and CNN has successfully been used in various tasks such as saliency detection for image recognition (Tang et al., 2016), document modeling (Tang et al., 2015) and music classification (Choi et al., 2016).

We first briefly discuss properties of RNNs, the neural MT framework of Bahdanau et al. (2015) and convolutional neural networks in Section 2 and subsequently discuss the related work on Convolutional neural networks in machine translation in Section 3. We introduce our model in Section 4 and discuss the its details. Experiments and results are discussed in Sections 5 and 6 respectively.

2. Background

2.1. Recurrent Neural Network

Given a sequence $[(x_1, x_2, ..., x_n)]$ of length *'n'*, at any timestep *'i'*, an RNN represents the hidden state output as function of the previous hidden state h_{i-1} output and the current input x_i

$$h_i = f(h_{i-1}, x_i) \tag{1}$$

f is commonly a nonlinear function. Thus RNNs represent a sequence as a vector by a function of previous history and current input. It is this recurrence property of RNNs that makes them capable to capture larger context such as long distance dependencies commonly observed in variable length texts.

A common problem observed while training RNN is the decay of gradient over long distance dependencies. To resolve this problem, Hochreiter and Schmidhuber (1997) proposed long-short term memory networks (LSTM) which use input, output, and forget gates to control the amount of information that can pass through a cell unit in the RNN.

2.2. Neural Machine Translation

We employ an NMT system based on Luong et al. (2015) which is a simple encoder-decoder network. The encoder is a multi-layer recurrent network (we use LSTMs) which converts an input sentence $[(x_1, x_2, ..., x_n)]$ into a sequence of hidden states $[(h_1, h_2, ..., h_n)]$.

$$h_i = f_{enc}(x_i, h_{i-1}) \tag{2}$$

Here, f_{enc} is an LSTM unit. The decoder is another multi-layer recurrent network which predicts a target sequence $y = (y_1, y_2, ...y_m)$. Each word in the sequence is predicted based on the last target word y_{i-1}, the current hidden state of the decoder s_j and the context vector c_j. The probability of the sentence is modelled as product of the probability of each target word.

$$p(\mathbf{y}) = \prod_{j}^{m} p(y_j|y_1, ...y_{j-1}, \mathbf{x}) = \prod_{j}^{m} g(y_j, s_j, c_j) \tag{3}$$

where g is a multi-layer feed forward neural network with nonlinear transformation and a softmax layer which generates the probability of each word in the target vocabulary. The end-to-end network is trained by maximizing log-likelihood over the training data. In Equation 3, s_j is the decoder hidden state generated by LSTM units similar to the encoder.

$$s_j = f_{dec}(s_{j-1}, y_{j-1}, c_j) \tag{4}$$

The context vector c_j in turn is calculated using an attention mechanism (Luong et al., 2015) as weighted sum of annotations of the encoder states h_i's.

$$c_j = \sum_{i=1}^{n} \alpha_{ji} h_i \tag{5}$$

where α_{ji} are attention weights corresponding to each encoder hidden state output h_i calculated as follows :

$$\alpha_{ji} = \frac{exp(z_i)}{\sum_{k=1}^{n} exp(z_k)} \tag{6}$$

Activations $z_k = a(s_{j-1}, h_k)$ are calculated by using a context function such as the dot product between the current decoder state s_{j-1} and each of the hidden states

of the encoder h_k's. Figure 1 shows the NMT framework with the encoder-decoder architecture and attention modeling.

In order to reduce the memory requirement for softmax operation on large number of words, source and target vocabulary are usually clipped to a fixed number of most frequent words.Translation is performed by a simple left-to-right beam search algorithm which maintains a small set of *'b'* best hypotheses for each target word. A hypothesis is complete as soon as end of sentence ("$< EOS >$") symbol is produced. A more detailed description of the decoding algorithm can be found in Sutskever et al. (2014).

2.3. Convolutional Neural Networks

Unlike recurrent neural networks, which are applied to a sequence of inputs, feeding the hidden layer from one timestep to the next, convolutional neural networks apply filters of fixed length over a window of inputs and generate outputs of fixed size. As discussed in (Kim, 2014), a narrow convolution operation involves applying a filter θ over a window of *'w'* inputs in order to generate a new feature. *'w'* is known as the width of the filter. The new feature CN_i applied to input window x_i to x_{i+w} is then defined as :

$$CN_i = \sigma(\theta \cdot x_{i-[(w-1)/2]:i+[(w-1)/2]} + b) \tag{7}$$

This feature extraction capability of CNNs makes them suitable for image processing. In NLP, CNNs have been used for tasks such as sentence classification which require computation of all possible phrases or segments of the input sentence regardless of their grammaticality.

3. Related Work

Although recurrent neural networks are very popular for NLP tasks, CNNs have also been used to model tasks such as text or sentence classification (Kim, 2014), sentiment analysis (dos Santos and Gatti, 2014), document modeling (Tang et al., 2015) and sentence modeling (Kalchbrenner et al., 2014) where specific features such as n-grams and phrases are more important than location specific or grammatical features of the sentence.

Similarly, the standard approach to neural Machine translation is the RNN based encoder-decoder network. However, there have been various attempts recently towards using convolutional networks in neural MT as well as additional models in Phrase-based MT. The first attempt to use convolutional networks in an end-to-end NMT framework is Cho et al. (2014). They fully replace the recurrent encoder with a gated recursive convolutional network whose weights are recursively applied to the input sequence until it outputs a single fixed-length vector. However, their experiments demonstrate that translation performance of such a network cannot surpass that of fully recurrent encoder.

Recently, Gehring et al. (2016) also proposed a similar architecture where the recurrent encoder is again fully replaced by a deep convolutional neural network. An important feature in their architecture is the use of a position embedding which encodes the relative position of each word in the source sentence. Their experiments demonstrate that while translation performance of the network is improved by using a very deep convolutional network, without the position embeddings, it drops substantially below the standard RNN/LSTM encoder baseline. This implies that a CNN encoder by itself with simple word embeddings alone cannot encode position-dependent features which are otherwise efficiently captured by an RNN encoder. Another recent approach using convolutional networks in neural MT is the *ByteNet* system by Kalchbrenner et al. (2016). They attempt to replace both the encoder and decoder with dilated convolutional networks stacked on each other.

All of the above approaches either aim to fully replace the recurrent encoders with convolutional encoders with which they aim to reduce the complexity of the network and the training speed, or to address the variable lengths of input sequences. In order to achieve performance comparable to RNN encoders, these approaches have to employ different mechanisms such as position embeddings to effectively capture the long distance dependencies and position-dependent features.

A related line of research is the character-level approach to NMT (Lee et al., 2016) where the main idea is to model the words as a combination of characters using convolutions and then feed the output as word embeddings to the RNN encoder. Their aim is to avoid the constraint on limited vocabulary by character modeling.

An approach which has shown the strength of convolutional network as an additional feature for Phrase-based MT is Meng et al. (2015). They show improvements over a standard Phrase-based MT by encoding the source sentence with a convolutional network and using it as a neural language model as an additional feature. To the best of our knowledge ours is the first attempt to combine recurrent and convolutional networks to model an encoder for neural machine translation.

4. Convolutional over Recurrent model (CoveR):

As discussed in Section 2, using the standard RNN framework, the context vector is a weighted sum of encoder hidden states h_i. The attention weights as in Equation (6), are also calculated by a similarity function between the decoder state s_j and encoder states $h_i s$. The attention weights mainly score how well the inputs around position *j* and the output at position *i* match (Bahdanau et al., 2015). Since each of these vectors h_i is compact summary of the source sentence up to word *i*, the previous or future context available to the alignment function is only given by these compact global representations. We propose that instead of relying only on these single recurrent outputs, a composition of multiple hidden state outputs of the encoder can provide the attention function with additional context about the relevant features of the source sentence.

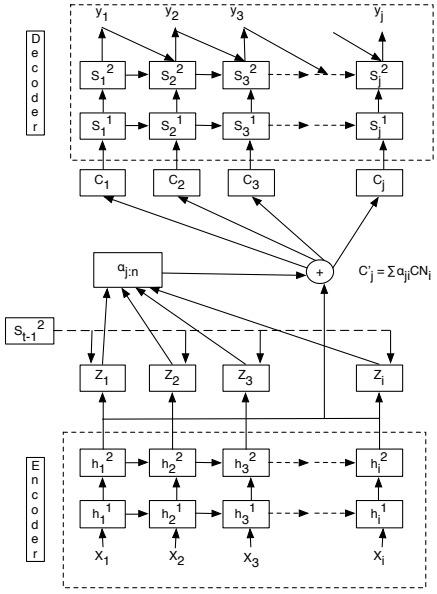

Figure 1. NMT encoder-decoder framework

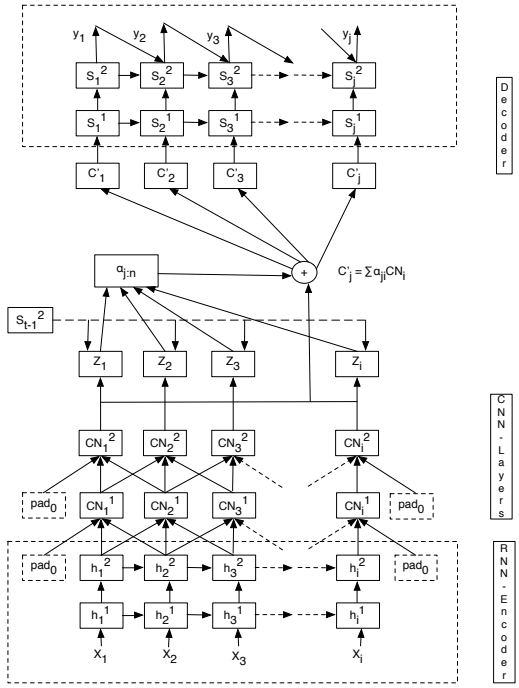

Figure 2. Convolution over Recurrent model

In order to do this, we apply multiple layers of fixed size convolution filters over the output of the RNN encoder at each time step. As shown in Figure 2, for our model the input to the first convolution layer is the hidden state output of the RNN encoder. Thus CN_i^1 is defined as:

$$CN_i^1 = \sigma(\theta \cdot h_{i-[(w-1)/2]:i+[(w-1)/2]} + b) \tag{8}$$

At each layer, we apply a number of filters equal to the original input sentence length. Each filter is of width 3. Note that the length of the output of the convolution filters reduces depending on the input length and the kernel width. In order to retain the original sequence length of the source sentence we apply padding at each layer. That is, for each convolutional layer, the input is zero-padded so that the output length remains the same. The output of the final convolution layer is a set of vectors $[CN_1, CN_2, ...CN_n]$ generated by multiple convolution operations. The modified context vectors c'_i are then calculated similar to c_i using an attention mechanism by

calculating the context function between CN_is and s_j.

$$\alpha_{ji} = \frac{exp(a(s_{j-1}, CN_i))}{\sum_{k=1}^{n} exp(a(s_{j-1}, CN_k))} \tag{9}$$

$$c'_j = \sum_{i=1}^{n} \alpha_{ji} CN_i \tag{10}$$

Finally, the decoder is provided with the context vectors c'_i as follows:

$$p(y_i|y_1, ...y_{i-1}, \mathbf{x}) = g(y_i, s_i, c'_i) \tag{11}$$

Note that each of the vectors CN_i now represents a feature produced by multiple kernels over h_i. Thus each CN_i represents a wider context as compared to h_i.

It is common practice to use pooling along with convolutional filters in order to down-sample the features. However, since in the proposed model, we want to widen the context of the encoder output while still retaining the information represented in the RNN output h_i, and also retaining the original sequence length, we do not apply pooling in our model.

With the increasing depth of the network, the training of the network becomes unstable. In order to ease and stabilize the training with multiple layers, we use residual connections (He et al., 2015) between the input and output of each convolutional layer.

5. Experimental Set-Up

5.1. Data

We conduct experiments on English-German translation. We use the translation data provided for WMT-2015 (Bojar et al., 2015). The training data provided for the task is approximately 4.2 million sentence pairs. We keep source sentences with a maximum sequence length of 80 words. After filtering out sentences longer than this limit and also removing duplicate sentence pairs, we are left with a parallel text of approximately 4 million sentence pairs. We reserve 5000 sentence from this bitext for perplexity validation and use the rest for training. We use wmt-newstest2013 as development set and wmt-newstest2014 and wmt-newstest2015 as test-sets. Results are reported in terms of case-insensitive BLEU-4 (Papineni et al., 2002). Approximate randomization (Noreen., 1989; Riezler and Maxwell, 2005) is used to detect statistically significant differences.

5.2. Baselines

We train a baseline NMT system based on Luong et al. (2015) using the Torch deep learning framework. It is a two layer unidirectional LSTM encoder-decoder with an

	newstest'13 (dev)	newstest'14	newstest'15
Baseline	17.9	15.8	18.5
Deep RNN encoder	18.3	16.2	18.7
CoveR	**18.47**$^{\triangle}$	**16.9**$^{\triangle}$	**19.00**

Table 1. BLEU scores over dev and test sets for Baseline, Deep RNN and CoveR (proposed model). Results marked with $\triangle$ are statistically significant at $p < 0.05$ over baseline

attention (dot product) mechanism. Both the encoder and decoder have input embedding and hidden layer of size 1000. As it is common practice, we limit the vocabulary sizes to 60k for the source and 40k for the target side. Parameters are optimized using stochastic gradient descent. We set the initial learning rate as 1 with a decay rate of 0.5 for each epoch after 5th epoch. Model weights are initialized uniformly within [-0.02, 0.02]. A dropout value of 0.2 is applied for each layer. We train for maximum of 20 epochs and decode with standard beam search with beam size of 10. All models are trained on NVIDIA Titan-X (Pascal) devices.

5.3. CoveR model

As discussed in Section 3, our model is a simple extension of the standard NMT model in which the RNN encoder is extended with additional convolution layers. We add three convolution layers on top of the output of the second RNN layer of the encoder. Note that similar to the baseline system, the RNN decoder has the same number of layers as the RNN encoder i.e., 2. For all layers we apply convolution filters of fixed width 3. The number of filters at each layer is same as the input sequence length. Each filter operates on a window of 3 consecutive inputs and generates a single output with a dimension equal to the input. Thus at each layer the output sequence length is reduced by 2 as compared to input as shown in Figure 2. In order to retain the full sequence length, we apply one zero-padding on both sides of the input. All other optimization parameters are the same as for the baseline.

5.4. Deep RNN encoder

In order to verify that the improvements achieved by the proposed model are due to the convolutions and not just because of the increased number of parameters, we also compare our model to another RNN baseline with an increased number of recurrent layers for encoder. Since we added three convolution layers to the encoder in our proposed CoveR model resulting in a total of 5 layers (2 recurrent + 3 convolution), for a fair comparison, we train a deep encoder with five recurrent layers. For this deep NMT system, the number of layers in decoder remains the same as for the baseline i.e., 2. The initial states of the decoder layers are initialized through a nonlinear transformation of all layers of the encoder RNN. This is done by concatenation of

Example 1:	
Source :	as the reverend martin luther king jr. said fifty years ago
Reference :	wie pastor martin luther king jr. vor fünfzig jahren sagte :
Baseline :	wie der martin luther king jr. sagte
Cover :	wie der martin luther king jr. sagte **vor fünfzig jahren** :
Example 2:	
Source :	he said the itinerary is still being worked out .
Reference :	er sagte , das genaue reiseroute werde noch ausgearbeitet .
Baseline :	er sagte , dass die strecke noch <unk> ist .
Cover :	er sagte , die **reiseroute** wird noch **ausgearbeitet** .

Table 2. Translation examples. Words in bold show correct translations produced by our model as compared to the baseline.

the final states of all the five layers of the encoder resulting in a vector of size 5xD (*'D'* is the dimension of the hidden layer) and then downgrading it to size 2xD by a simple non-linear transformation and finally splitting it in two vectors of size 'D' which are used to initialize each of the layers of the decoder.

6. Results

Table 1 shows the results for our English-German translations experiments. The first column indicates the best BLEU scores on the development set newstest'13 for all three models after 20 epochs. Results are reported on the newstest'14 and newstest'15 test sets. Our CoveR model shows improvements of 1.1 and 0.5 BLEU points respectively over the two test sets. Although the deep RNN encoder performs better than the baseline, the improvements achieved are lower than that of the CoveR model.

6.1. Qualitative analysis and discussion

Table 2 provides some of the translation examples produced by the baseline system and our CoveR model. A general observation is the improved translations by our model over the baseline with regard to the reference translation which is also reflected by the improved BLEU scores.

More specifically, Example 1 shows instances where the baseline suffers in some cases from incomplete coverage of the source sentence. One reason for such incomplete translations is the lack of coverage modeling which has been handled using coverage embeddings (Tu et al., 2016). We observe this problem frequently in instances where a specific word might signal completion of a sentence despite more words in the sequence remain to be be translated. These words can cause the generation of next target word as the end-of-sentence *'EOS'* symbol. Since the beam search decoding algorithm considers a hypothesis complete when the end of sentence is generated, in

such instances search stops, aborting further expansions, while ignoring the remaining words. For instance in Example 1 in Table 2, by relying on the attention mechanism, the baseline system generates the translation of *'said'* as *'sagte'*, the model might give a preference to the generation of an end-of-sentence *'EOS'* symbol immediately following the verb. On the other hand, for our CoveR model, at target position 8, a wider context is available to the model through convolutional layers from both directions signalling the presence of other words remaining in the input sentence, thus producing a more complete translation. Another difference between the baseline model

Figure 3. Attention distribution for Baseline

Figure 4. Attention distribution for CoveR model

and our CoveR model that can be observed in Example 2 is that attention weights are distributed more uniformly among the source words. Specifically, for target position 6, as shown in Figure 3 the baseline model pays attention mainly to *'itinerary'* and

'is' resulting in the generation of target word *'strecke'* which is a more common translation for the English word *'route'*. On the other hand as shown in Figure 4, for the same position, the CoveR model pays attention to *'itinerary'* as well as the last three words *'being worked out'*. This allows for the generation of the more specific and correct target word *'reiseroute'*. Note that the <unk> symbols produced are a result of the vocabulary restriction.

7. Conclusion

In this paper, we proposed a convolutional over recurrent network encoder model for neural machine translation. The model involves feeding outputs of the RNN encoder to multiple convolutional layers of fixed kernel size. Our experiments on English-German translation demonstrate that the proposed model improves translation as compared to a standard RNN encoder. An improvement of 0.5 to 1 BLEU points is observed on different test sets. A qualitative analysis of the translations of our model shows that CNNs capture the smaller context corresponding to each word more effectively while RNNs model the global information thus capturing grammaticality and dependencies with the source sentence.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*, 2015.

Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, September 2015.

Cho, Kyunghyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, 2014.

Choi, Keunwoo, George Fazekas, Mark B. Sandler, and Kyunghyun Cho. Convolutional Recurrent Neural Networks for Music Classification. *CoRR*, abs/1609.04243, 2016.

dos Santos, Cicero and Maira Gatti. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING 2014*, Dublin, Ireland, August 2014. ACL.

Gehring, Jonas, Michael Auli, David Grangier, and Yann N. Dauphin. A Convolutional Encoder Model for Neural Machine Translation. *CoRR*, abs/1611.02344, 2016.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

Hochreiter, Sepp and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8): 1735–1780, 1997. ISSN 0899-7667.

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. A Convolutional Neural Network for Modelling Sentences. *CoRR*, abs/1404.2188, 2014.

Kalchbrenner, Nal, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural Machine Translation in Linear Time. *CoRR*, abs/1610.10099, 2016.

Kim, Yoon. Convolutional Neural Networks for Sentence Classification. *CoRR*, abs/1408.5882, 2014.

Lee, Jason, Kyunghyun Cho, and Thomas Hofmann. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *CoRR*, abs/1610.03017, 2016.

Luong, Thang, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015. ACL.

Meng, Fandong, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. Encoding Source Language with Convolutional Neural Network for Machine Translation. In *Proceedings of the 53rd Annual Meeting of the ACL*, Beijing, China, July 2015. ACL.

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1045–1048. International Speech Communication Association, 2010. ISBN 978-1-61782-123-3.

Mueller, Jonas and Aditya Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Noreen., Eric W. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley- Interscience, 1989.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the ACL*, 2002.

Riezler, Stefan and John T. Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Ma- chine Translation and/or Summarization*, 2005.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215, 2014.

Tang, Duyu, Bing Qin, and Ting Liu. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, September 2015. ACL.

Tang, Youbao, Xiangqian Wu, and Wei Bu. Deeply-Supervised Recurrent Convolutional Neural Network for Saliency Detection. *CoRR*, abs/1608.05177, 2016.

Tu, Zhaopeng, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Coverage-based Neural Machine Translation. *CoRR*, abs/1601.04811, 2016.

Address for correspondence:
Praveen Dakwale
p.dakwale@uva.nl
C3.230, Science Park 904, Amsterdam 1098XH, The Netherlands