Procesamiento del Lenguaje Natural, Revista nº 55, septiembre de 2015, pp 109-116 recibido 26-03-2015 revisado 27-04-2015 aceptado 14-05-2015

A Bidirectional Recurrent Neural Language Model for Machine Translation*

Un modelo de lenguaje neuronal recurrente bidireccional para la traducción automática

Álvaro Peris, Francisco Casacuberta
PRHLT – Universitat Politècnica de València
Camino de Vera s/n, 46022, Valencia (Spain)
lvapeab@fiv.upv.es, fcn@prhlt.upv.es

Resumen: Se presenta un modelo de lenguaje basado en representaciones continuas de las palabras, el cual se ha aplicado a una tarea de traducción automática estadística. Este modelo está implementado por una red neuronal recurrente bidireccional, la cual es capaz de tener en cuenta el contexto pasado y futuro de una palabra para realizar predicciones. Debido su alto coste temporal de entrenamiento, para obtener datos de entrenamiento relevantes se emplea un algoritmo de selección de oraciones, el cual busca capturar información útil para traducir un determinado conjunto de test. Los resultados obtenidos muestran que el modelo neuronal entrenado con los datos seleccionados es capaz de mejorar los resultados obtenidos por un modelo de lenguaje de n-gramas.
Palabras clave: Modelado de lenguaje, redes neuronales recurrentes bidireccionales, selección de datos, traducción automática estadística.

Abstract: A language model based in continuous representations of words is presented, which has been applied to a statistical machine translation task. This model is implemented by means of a bidirectional recurrent neural network, which is able to take into account both the past and the future context of a word in order to perform predictions. Due to its high temporal cost at training time, for obtaining relevant training data an instance selection algorithm is used, which aims to capture useful information for translating a test set. Obtained results show that the neural model trained with the selected data outperforms the results obtained by an n-gram language model.
Keywords: Language modelling, bidirectional recurrent neural networks, instance selection, statistical machine translation.

1 Introduction

Many natural language processing applications, such as automatic speech recognition (ASR), handwritten text recognition (HTR) or statistical machine translation (SMT), require the use of a language model, which determines how well a word sequence is formed. The classical approach, the n-gram language model, is a count-based technique in a discrete representation space. Thanks to the smoothing techniques (e.g. Chen and Goodman (1998)), the n-gram models tackle the data sparseness problem, and are capable of obtaining predictions for non-seen events. This leads to simple, robust and fast models, which are may be trained over huge amounts of data. On the other hand, since words are treated as indices in a vector, there are no concepts such as similarity nor semantic relationships between words. In addition, the n-gram model, only considers few context words: Typically, the order of the n-gram models ranges from 2 to 5, therefore the model takes from 1 to 4 context words, and long-term relationships are lost (Bengio et al., 2003). In the last years, more complex language models have been successfully developed. One of these approaches rely in a distributed representation of words: A real-valued, dense and low-dimensional represen-

* The research leading to these results has received funding from the the Generalitat Valenciana under grant Prometeo/2009/014.

ISSN 1135-5948

tation in a continuous space. In these models, probability estimation is carried out in this continuous space, typically by means of a neural network. Furthermore, given the nature of continuous models, the learned function is inherently smoothed.

The proposed model belongs to this latter family and aims to overcome the aforementioned drawbacks of n-gram language models: First, by projecting the words into a continuous space, the model profits from a richer representations of words. Second, by using a bidirectional recurrent neural network (BRNN), the context of a word is determined not only by its preceding words, but also by its following words. This allows the model to produce more informed predictions.

Since the computational cost of such model is high, only a subset of the available training corpora is used to train the model. This can be seen as an instance of a domain adaptation problem, where the goal is to choose the most adequate sentences for translating a given test set from a sentence pool – in this case, the full training corpus. The selection of the training events is performed using a method belonging to the so-called *feature-decay* selection algorithms. The model is then applied to a SMT task, reranking N-best lists of translation hypotheses.

In this paper, we develop an extension of recurrent neural network language models, using a bidirectional neural network for carrying out the probability estimation. We show that this model can be appropriately trained for a given test set using a subset of all available data. This subset of data is chosen using an instance selection algorithm. Results show that the neural model combined with an n-gram language model enhances the performance of a SMT system.

The paper is structured as follows: In Section 2, related approaches are reviewed. In Section 3 the proposed language model is described. Section 4 states the motivation for selecting training instances. Performed experiments and results are shown in Section 5. Finally, conclusions about the work are obtained in Section 6.

2 Related work

The use of continuous spaces is nowadays a hot topic in the language modelling field. Since Bengio et al. (2003) proposed to perform a linear projection from the discrete to the continuous space and learn the probability function in this space, many other works followed these ideas. Bengio et al. and Schwenk (2013) performed the probability estimation through a feedforward neural network. Mikolov (2012) used a recurrent neural network (RNN) for that purpose. In his model, there was no projection layer, words were mapped directly to the hidden layer. Sundermeyer et al. (2012) combined both models, having a projection layer connected to a recurrent layer, with LSTM units. Pascanu et al. (2014) extended the RNN architecture, which led to *deep* RNN, and it was applied to language modelling. In the field of SMT, neural language models have also recent applications: Baltescu et al. (2014) coupled a feedforward neural language model into a SMT decoder. Wang et al. (2014) approximated a neural language model with an n-gram language model, according to bilingual information extracted from the phrase table.

Besides language modelling, continuous spaces have also been included as additional information sources in SMT systems. Sundermeyer et al. (2014) used a bidirectional LSTM network, architecturally similar to the proposed model, as translation model. Devlin et al. (2014) extended the original neural language model from Bengio et al. (2003), and developed a neural translation model. This model could be integrated into a hierarchical decoder and offered impressive results. Moreover, full-neural translation systems have been recently proposed (Bahdanau et al., 2014; Sutskever et al., 2014), offering encouraging results: In Luong et al. (2014), a full-neural system outperformed for the first time a state-of-the-art phrase-based system.

Instance selection techniques have been typically applied in the scope of domain adaptation or active learning. Gascó et al. (2012) showed that a SMT system trained with selected sentences outperformed a system trained with all available data. In this case, the selection criterion was to choose sentences which contained unseen (or seldom seen) n-grams. Other works used perplexity as selection criterion (Mandal et al., 2008). The selection method used in this paper is an instantiation of that proposed in Biçici and Yuret (2011), where the goal is to obtain a selection which maximizes the coverage of

the target part of the test set.

3 Bidirectional Recurrent Language Model

3.1 Recurrent neural networks

Recurrent architecture of neural networks are appropriate to model a temporal-discrete behaviour. Given an input sequence $\mathbf{x}_1^T = x_1, \ldots, x_T$, a RNN produces an output sequence $\mathbf{y}_1^T = y_1, \ldots, y_T$, computed as:

$$\mathbf{h}_t = f_h(x_t, \mathbf{h}_{t-1}) \quad (1)$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t) \quad (2)$$

where $\mathbf{h}_t$ is the hidden state at timestep t, f_h is the hidden state function (e.g. sigmoid or hyperbolic tangent) and f_o is the output function (e.g. a multi-layer perceptron with an output layer performing the *softmax* function).

RNNs are typically trained via stochastic gradient descent, using the backpropagation through time algorithm (Werbos, 1990), to minimize a cost function under some optimality criterion, typically cross-entropy between the output of the system and the training data probability distribution.

3.2 Bidirectional recurrent neural networks

A drawback of regular RNNs is that the input sequence is only scanned in one direction, normally from past to future. In order to capture both past and future context, bidirectional RNNs were proposed by Schuster and Paliwal (1997). The main idea is to have two independent recurrent layers: One layer process the input sequence in forward time direction (from 1 to T), while the other layer process the input sequence reversed in time (from T to 1). Since hidden layers have no interaction between them, bidirectional RNNs can be trained using the same algorithms as those used for unidirectional RNNs. Following prior notation, bidirectional RNN is defined as:

$$\mathbf{h}_t^{\mathrm{f}} = f_h(x_t, \mathbf{h}_{t-1}^{\mathrm{f}}) \quad (3)$$

$$\mathbf{h}_t^{\mathrm{b}} = f_h(x_t, \mathbf{h}_{t+1}^{\mathrm{b}}) \quad (4)$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t^{\mathrm{f}}, \mathbf{h}_t^{\mathrm{b}}) \quad (5)$$

where $\mathbf{h}_t^{\mathrm{f}}$ is the forward layer and $\mathbf{h}_t^{\mathrm{b}}$ is the backward layer. The output is a combination produced by the output function f_o of both backward and forward layers.

3.3 Bidirectional recurrent language model

The task of statistical language modelling consists in estimating the probability distribution over a sequence of words $\mathbf{x}_1^T = x_1, \ldots, x_T$. Applying the chain rule, the sequence probability $p(\mathbf{x}_1^T) = p(x_1, \ldots, x_T)$ can be decomposed as:

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}) \quad (6)$$

In the RNN framework, information about the history $(x_1, \ldots, x_{t-1})$ is represented in the hidden recurrent layer. Thus, sequence probability is rewritten as:

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | \mathbf{h}_t) \quad (7)$$

As we move to a BRNN, probability is conditioned by both forward and backward states:

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | \mathbf{h}_t^{\mathrm{f}}, \mathbf{h}_t^{\mathrm{b}}) \quad (8)$$

In our language model architecture, input words are one-hot vectors, binary vectors with all elements set to 0 except the index that represents the input word, which is set to 1. Those vectors are projected into the continuous space through a projection layer and then fed to the BRNN. As architectural choices of the network, the hidden function (f_h) is the sigmoid function. The output function (f_o) is modelled with a 2-layer perceptron, which its first layer makes use of the sigmoid activation function and it is fully connected to the output layer, which makes use of the softmax cost function in order to obtain correct output probabilities:

$$\sigma(z_k) = \frac{\exp(z_k)}{\sum_{k'=1}^{K} \exp(z_{k'})} \quad (9)$$

where z_k is the k-th output unit. Each output unit represents a word in the vocabulary, hence, the output layer is vocabulary-sized. At test time, the probability of a sentence is normalized with respect to the length of the

sentence, in order to prevent benefits to short sentences (Graves, 2013). Since using the full vocabulary is computationally unaffordable, a shortlist is used: Only the most K frequent words are taken into account. The rest are mapped to a special token `<unk>`. Figure 1 shows a scheme of the model.

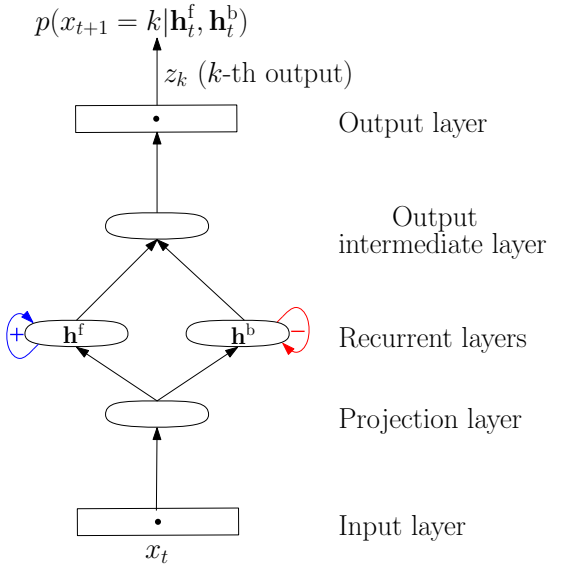

Figure 1: Architecture of the bidirectional language model, similarly depicted as Sundermeyer et al. (2014). Input and output layers have the size of the vocabulary.

4 Instance selection

Typically, corpora used for training SMT systems are much larger than the test set. Therefore, the training set contains noise and sentences that are irrelevant for a specific task. Moreover, neural models have a high time complexity and such large corpora present computational challenges at training time. Techniques for selecting the most appropriate set of training sentences for a given test set are suitable in this scenario. Within these approaches, feature-decay algorithms perform the selection of sentences from the training set aiming to maximize the coverage of the target language n-grams (features) of the test set (Biçici and Yuret, 2011).

The target side of the test set is unknown, only the corresponding source sentences are available. Since a source sentence potentially has many translations, performing a selection which objective is maximizing the coverage of the source part of the test does not guarantee an adequate coverage of the target part of the test. For treating this issue, feature-decay algorithms try to maximize the diversity of the selected instances. The method provides initial scores to the features, according to an initialization function. Iteratively, the features with highest scores are selected and scores of these features included in the selection are reduced. Therefore, it is expected that, in following iterations, different features will be included in the selection. Particular choices in the initialization, scoring and decaying functions provide different selection methods, such as n-gram coverage or TF-IDF (Eck et al., 2005).

FDA5, a parametrization of feature-decay algorithms has been recently proposed (Bicici and Yuret, 2015). In this algorithm, the selection of the data is performed according to 5 parameters: The feature initialization function considers frequency and inverse frequency of the tokens in a feature. Scores of the features decay following a polynomial and an exponential factor. Finally, the sentence scoring function is the sum of all feature values of a sentence, scaled by a sentence-length factor.

5 Experiments and results

The model was tested in the Spanish–English *EU* translation task – a selection from the *Bulletin of the European Union* (Khadivi and Goutte, 2003). The Thot toolkit (Ortiz-Martínez and Casacuberta, 2014) was used for building the translation models. The neural language model was used to rescore N-best lists, which were obtained executing a weight adjustment process, by means of the downhill simplex optimization method (Melder and Nead, 1965), using BLEU as function to maximize. At each iteration of the optimization process, a 200-best list was generated and merged with the list of the previous iteration. The process continued until no new elements were included in the N-best list. As result, the majority of the probability mass of translation hypotheses was included in the N-best list. The average size of the lists was $N = 4300$.

5.1 Data selection

For selecting an appropriate number of sentences, different selection sizes were tested. An n-gram language model was trained over the target side of the selected data and its perplexity was computed. We chose a selection which provide a good balance be-

tween complexity and quality. Figure 2 shows the relation between the number of source words selected and the number of training sentences selected (corpus coverage). If a word is included in the selection, all the sentences which contain such word belong to the selection. Figure 3 shows bigram coverage for the test set, according the number of source words selected. Finally, Figure 4 shows the number of out-of-vocabulary (OOV) words in the test test set with respect to the number of words selected. It was observed that performing a selection from one million words ahead, produced small variations both in test OOV words and in bigram coverage values.

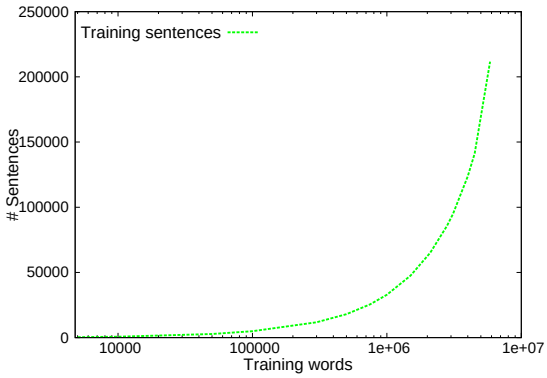

Figure 2: Training corpus coverage according the number of source words selected.

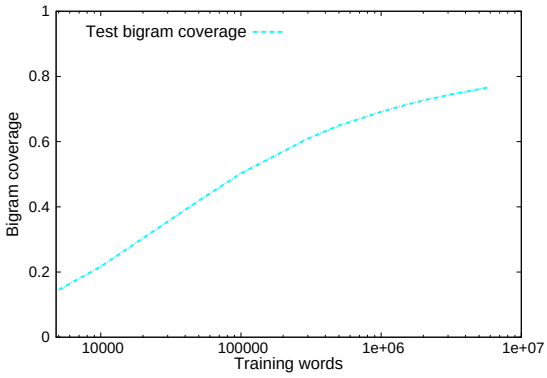

Figure 3: Relation between the test set bigram coverage of the selected corpus and the number of source words selected.

Figure 5 reports the perplexity obtained by different n-gram language models trained over the selected data. In order to obtain a fair comparison between language models, for computing perplexity, all neural and n-gram language models were trained using the same vocabulary. 4-gram performed slightly better than 5-gram when the full corpus was not selected. Thus, in following experiments, the

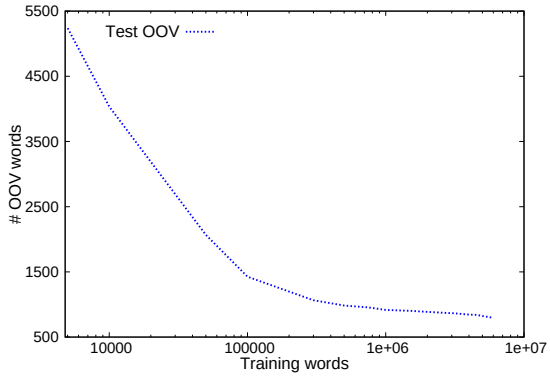

Figure 4: Number of OOV words in the test set according the number of words selected.

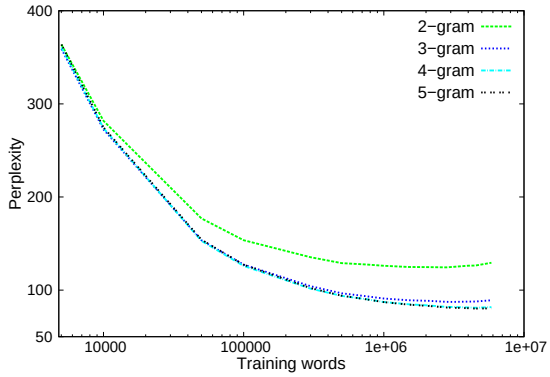

Figure 5: Perplexity obtained by different n-gram language models according the number of selected words.

order of the n-gram language model was set to 4. These results corroborated those given by the OOV words and bigram coverage: For selections larger than a million of words, obtaining a humble perplexity enhancement was expensive, as the number of instances that should be included rapidly grew.

Therefore for obtaining the reduced corpus, FDA5 considered 1 million words from the source corpus. Statistics of the selected instances are shown in Table 1. The obtained corpus contained only a 15% sentences of the original corpus, but the perplexity of the models trained over it, for the test set, was increased just a 6.4% with respect to the full corpus.

5.2 Language models

The hyperparameters of the neural language model (size of the projection layer, size of the recurrent layers and size of the output intermediate layer), were chosen following a perplexity minimization criterion. The learning rate was initially set to 1 and was halved at

		En	Sp
Full training set	Sentences	213k	
	Running words	5.2M	5.9M
	Vocabulary	50k	64k
Reduced training set	Sentences	33k	
	Running words	891k	1M
	Vocabulary	21k	28k
Test set	Sentences	800	
	Running words	20k	23k

Table 1: Statistics for full EU corpus, selected instances and test set (k and M stand for thousands and millions, respectively). The reduced corpus was obtained selecting one million words from the source part (Spanish) of the training set.

the start of each training epoch if the validation entropy did not decrease a 0.3% with respect the previous one (Mikolov, 2012). Following Pascanu et al. (2014), the network was initialized using the standard deviations of a Gaussian white noise distribution. The size of the shortlist was set to $K = 10,000$. An analogous unidirectional model was also trained.

Table 2 shows the perplexities obtained by the different language models over the test set. The bidirectional RNN language model offered a performance similar to that of the n-gram language model trained with the same data, while the perplexity of the unidirectional RNN was slightly higher.

Language model	**Perplexity**
Full n-gram	81.7
Reduced n-gram	87.3
Unidirectional RNN	95.6
Bidirectional RNN	87.9

Table 2: Test set perplexity for different language models. Full n-gram row refers to an n-gram trained over the complete corpus. Reduced n-gram refers to an n-gram trained only over the selected instances. Both neural models are trained over selected instances. All models were trained using the same vocabulary (10,000 words).

Table 3 shows the BLEU scores obtained by the different language models for the test set. The bidirectional model offered a small improvement with respect the unidirectional one, but both were worse than the n-gram language model. The neural language model was also linearly interpolated with an n-gram language model. The interpolation coefficient (λ) was determined by sampling in a development set. The sampling interval was $[0.1, 0.9]$, with a step of 0.1. The optimal value was found at $\lambda = 0.6$. This interpolation provided an enhancement of the system performance. That means that both approaches were complementary: Because of their nature, n-gram language models are robust modelling local dependencies. The neural network introduced additional information, which was useful in order to enhance the performance of the system. Although differences in the results obtained were statistically non-significant, we observed a trend in them.

Language model	**BLEU**
n-gram	30.8
Unidirectional RNN	30.2
Bidirectional RNN	30.3
Bidirectional RNN + n-gram	**31.3**

Table 3: Test set BLEU score for the different language models.

6 Conclusions

In this paper, a neural language model implemented by means of a bidirectional neural network has been presented. Since the computational training cost of the model was high, a subset of the training corpus was selected, using domain adaptation techniques. The network was successfully trained with the reduced corpus, obtaining a perplexity similar to that of an n-gram language model trained with the full corpus. It was shown that both neural and n-gram language models are complementary: The latter model was focused in local dependencies, while the first one was able to incorporate additional dependencies. That was supported by the results: The combination of both models provided enhancements with respect the use of the models solely.

Finally, the bidirectional architecture of the neural network language model exhibited a better behaviour than the unidirectional architecture, in terms of perplexity and translation quality.

References

Bahdanau, D., K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. Technical report, arXiv preprint arXiv:1409.0473.

Baltescu, P., P. Blunsom, and H. Hoang. 2014. OxLM: A neural language modelling framework for machine translation. *The Prague Bulletin of Mathematical Linguistics*, 102(1):81–92.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Machine Learning Research*.

Biçici, E. and D. Yuret. 2011. Instance selection for machine translation using feature decay algorithms. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 272–283. Association for Computational Linguistics.

Bicici, E. and D. Yuret. 2015. Optimizing instance selection for statistical machine translation with feature decay algorithms. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(2):339–350.

Chen, F. and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard U., Cambridge, MA.

Devlin, J., R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380. Association for Computational Linguistics.

Eck, M., S. Vogel, and A. Waibel. 2005. Low cost portability for statistical machine translation based on n-gram frequency and TF-IDF. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 61–67.

Gascó, G., M. A. Rocha, G. Sanchis-Trilles, J. Andrés-Ferrer, and F. Casacuberta. 2012. Does more data always yield better translations? In *Proceedings of the 13th European Chapter of the Association for Computational Linguistics*, pages 152–161.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv:*1308.0850 [cs.NE].

Khadivi, S. and C. Goutte. 2003. Tools for corpus alignment and evaluation of the alignments (deliverable d4.9). Technical report, Technical report, TransType2 (IST-2001-32091).

Luong, T., I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. 2014. Addressing the rare word problem in neural machine translation. arXiv preprint arXiv:1410.8206.

Mandal, A., D. Vergyri, W. Wang, J. Zheng, A. Stolcke, G. Tur, D. Hakkani-Tur, and N. F. Ayan. 2008. Efficient data selection for machine translation. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 261–264. IEEE.

Melder, J. A. and R. Nead. 1965. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Mikolov, T. 2012. *Statistical Language Models based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Ortiz-Martínez, D. and F. Casacuberta. 2014. The new Thot toolkit for fully-automatic and interactive statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 45–48.

Pascanu, R., Ç. Gülçehre, K. Cho, and Y. Bengio. 2014. How to construct deep recurrent neural networks.

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

Schwenk, H. 2013. CSLM - a modular open-source continuous space language modeling toolkit. In *INTERSPEECH*, pages 1198–1202. ISCA.

Sundermeyer, M., T. Alkhouli, J. Wuebker, and H. Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural*

Language Processing (EMNLP), pages 14–25. Association for Computational Linguistics.

Sundermeyer, M., R. Schlüter, and H. Ney. 2012. LSTM neural networks for language modeling. In *Interspeech*, pages 194–197.

Sutskever, I., O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS 2014)*.

Wang, R., H. Zhao, B-L. Lu, M. Utiyama, and E. Sumita. 2014. Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 189–195. Association for Computational Linguistics.

Werbos, P. J. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.