Using the Output Embedding to Improve Language Models

Ofir Press and **Lior Wolf**
School of Computer Science
Tel-Aviv University, Israel
{ofir.press,wolf}@cs.tau.ac.il

Abstract

We study the topmost weight matrix of neural network language models. We show that this matrix constitutes a valid word embedding. When training language models, we recommend tying the input embedding and this output embedding. We analyze the resulting update rules and show that the tied embedding evolves in a more similar way to the output embedding than to the input embedding in the untied model. We also offer a new method of regularizing the output embedding. Our methods lead to a significant reduction in perplexity, as we are able to show on a variety of neural network language models. Finally, we show that weight tying can reduce the size of neural translation models to less than half of their original size without harming their performance.

1 Introduction

In a common family of neural network language models, the current input word is represented as the vector $c \in \mathbb{R}^C$ and is projected to a dense representation using a word embedding matrix U. Some computation is then performed on the word embedding $U^\top c$, which results in a vector of activations h_2. A second matrix V then projects h_2 to a vector h_3 containing one score per vocabulary word: $h_3 = Vh_2$. The vector of scores is then converted to a vector of probability values p, which represents the models' prediction of the next word, using the softmax function.

For example, in the LSTM-based language models of (Sundermeyer et al., 2012; Zaremba et al., 2014), for vocabulary of size C, the one-hot encoding is used to represent the input c and $U \in \mathbb{R}^{C \times H}$. An LSTM is then employed, which results in an activation vector h_2 that similarly to $U^\top c$, is also in $\mathbb{R}^H$. In this case, U and V are of exactly the same size.

We call U the input embedding, and V the output embedding. In both matrices, we expect rows that correspond to similar words to be similar: for the input embedding, we would like the network to react similarly to synonyms, while in the output embedding, we would like the scores of words that are interchangeable to be similar (Mnih and Teh, 2012).

While U and V can both serve as word embeddings, in the literature, only the former serves this role. In this paper, we compare the quality of the input embedding to that of the output embedding, and we show that the latter can be used to improve neural network language models. Our main results are as follows: (i) We show that in the word2vec skip-gram model, the output embedding is only slightly inferior to the input embedding. This is shown using metrics that are commonly used in order to measure embedding quality. (ii) In recurrent neural network based language models, the output embedding outperforms the input embedding. (iii) By tying the two embeddings together, i.e., enforcing $U = V$, the joint embedding evolves in a more similar way to the output embedding than to the input embedding of the untied model. (iv) Tying the input and output embeddings leads to an improvement in the perplexity of various language models. This is true both when using dropout or when not using it. (v) When not using dropout, we propose adding an additional projection P before V, and apply regularization to P. (vi) Weight tying in neural translation models can reduce their size (number of parameters) to less than half of their original size without harming their performance.

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 157–163,
Valencia, Spain, April 3-7, 2017.

2 Related Work

Neural network language models (NNLMs) assign probabilities to word sequences. Their resurgence was initiated by (Bengio et al., 2003). Recurrent neural networks were first used for language modeling in (Mikolov et al., 2010) and (Pascanu et al., 2013). The first model that implemented language modeling with LSTMs (Hochreiter and Schmidhuber, 1997) was (Sundermeyer et al., 2012). Following that, (Zaremba et al., 2014) introduced a dropout (Srivastava, 2013) augmented NNLM. (Gal, 2015; Gal and Ghahramani, 2016) proposed a new dropout method, which is referred to as Bayesian Dropout below, that improves on the results of (Zaremba et al., 2014).

The skip-gram word2vec model introduced in (Mikolov et al., 2013a; Mikolov et al., 2013b) learns representations of words. This model learns a representation for each word in its vocabulary, both in an input embedding matrix and in an output embedding matrix. When training is complete, the vectors that are returned are the input embeddings. The output embedding is typically ignored, although (Mitra et al., 2016; Mnih and Kavukcuoglu, 2013) use both the output and input embeddings of words in order to compute word similarity. Recently, (Goldberg and Levy, 2014) argued that the output embedding of the word2vec skip-gram model needs to be different than the input embedding.

As we show, tying the input and the output embeddings is indeed detrimental in word2vec. However, it improves performance in NNLMs.

In neural machine translation (NMT) models (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014), the decoder, which generates the translation of the input sentence in the target language, is a language model that is conditioned on both the previous words of the output sentence and on the source sentence. State of the art results in NMT have recently been achieved by systems that segment the source and target words into subword units (Sennrich et al., 2016a). One such method (Sennrich et al., 2016b) is based on the byte pair encoding (BPE) compression algorithm (Gage, 1994). BPE segments rare words into their more commonly appearing subwords.

Weight tying was previously used in the log-bilinear model of (Mnih and Hinton, 2009), but the decision to use it was not explained, and its effect on the model's performance was not tested. Independently and concurrently with our work (Inan et al., 2016) presented an explanation for weight tying in NNLMs based on (Hinton et al., 2015).

3 Weight Tying

In this work, we employ three different model categories: NNLMs, the word2vec skip-gram model, and NMT models. Weight tying is applied similarly in all models. For translation models, we also present a three-way weight tying method.

NNLM models contain an input embedding matrix, two LSTM layers (h_1 and h_2), a third hidden scores/logits layer h_3, and a softmax layer. The loss used during training is the cross entropy loss without any regularization terms.

Following (Zaremba et al., 2014), we employ two models: large and small. The large model employs dropout for regularization. The small model is not regularized. Therefore, we propose the following regularization scheme. A projection matrix $P \in \mathbb{R}^{H \times H}$ is inserted before the output embedding, i.e., $h_3 = VPh_2$. The regularizing term $\lambda \|P\|_2$ is then added to the small model's loss function. In all of our experiments, $\lambda = 0.15$.

Projection regularization allows us to use the same embedding (as both the input/output embedding) with some adaptation that is under regularization. It is, therefore, especially suited for WT.

While training a vanilla **untied NNLM**, at timestep t, with current input word sequence $i_{1:t} = [i_1, ..., i_t]$ and current target output word o_t, the negative log likelihood loss is given by: $\mathcal{L}_t = -\log p_t(o_t|i_{1:t})$, where $p_t(o_t|i_{1:t}) = \frac{\exp(V_{o_t}^\top h_2^{(t)})}{\sum_{x=1}^{C} \exp(V_x^\top h_2^{(t)})}$, U_k (V_k) is the kth row of U (V), which corresponds to word k, and $h_2^{(t)}$ is the vector of activations of the topmost LSTM layer's output at time t. For simplicity, we assume that at each timestep t, $i_t \neq o_t$. Optimization of the model is performed using stochastic gradient descent.

The update for row k of the input embedding is:

$$\frac{\partial \mathcal{L}_t}{\partial U_k} = \begin{cases} (\sum_{x=1}^{C} p_t(x|i_{1:t}) \cdot V_x^\top - V_{o_t}^\top) \frac{\partial h_2^{(t)}}{\partial U_{i_t}} & k = i_t \\ 0 & k \neq i_t \end{cases}$$

For the output embedding, row k's update is:

$$\frac{\partial \mathcal{L}_t}{\partial V_k} = \begin{cases} (p_t(o_t|i_{1:t}) - 1) h_2^{(t)} & k = o_t \\ p_t(k|i_{1:t}) \cdot h_2^{(t)} & k \neq o_t \end{cases}$$

Therefore, in the untied model, at every timestep, the only row that is updated in the input embedding is the row U_{i_t} representing the current input

word. This means that vectors representing rare words are updated only a small number of times. The output embedding updates every row at each timestep.

In **tied NNLMs**, we set $U = V = S$. The update for each row in S is the sum of the updates obtained for the two roles of S as both an input and output embedding.

The update for row $k \neq i_t$ is similar to the update of row k in the untied NNLM's output embedding (the only difference being that U and V are both replaced by a single matrix S). In this case, there is no update from the input embedding role of S.

The update for row $k = i_t$, is made up of a term from the input embedding (case $k = i_t$) and a term from the output embedding (case $k \neq o_t$). The second term grows linearly with $p_t(i_t|i_{1:t})$, which is expected to be close to zero, since words seldom appear twice in a row (the low probability in the network was also verified experimentally). The update that occurs in this case is, therefore, mostly impacted by the update from the input embedding role of S.

To conclude, in the tied NNLM, every row of S is updated during each iteration, and for all rows except one, this update is similar to the update of the output embedding of the untied model. This implies a greater degree of similarity of the tied embedding to the untied model's output embedding than to its input embedding.

The analysis above focuses on NNLMs for brevity. In **word2vec**, the update rules are similar, just that $h_2^{(t)}$ is replaced by the identity function. As argued by (Goldberg and Levy, 2014), in this case weight tying is not appropriate, because if $p_t(i_t|i_{1:t})$ is close to zero then so is the norm of the embedding of i_t. This argument does not hold for NNLMs, since the LSTM layers cause a decoupling of the input and output embedddings.

Finally, we evaluate the effect of weight tying in **neural translation models**. In this model: $p_t(o_t|i_{1:t}, r) = \frac{exp(V_{o_t}^\top G^{(t)})}{\sum_{x=1}^{C_t} exp(V_x^\top G^{(t)})}$ where $r = (r_1, ..., r_N)$ is the set of words in the source sentence, U and V are the input and output embeddings of the decoder and W is the input embedding of the encoder (in translation models $U, V \in \mathbb{R}^{C_t \times H}$ and $W \in \mathbb{R}^{C_s \times H}$, where C_s / C_t is the size of the vocabulary of the source / target). $G^{(t)}$ is the decoder, which receives the context vector, the embedding of the input word (i_t) in U, and its previous state at each timestep. c_t is the context vector at timestep t, $c_t = \sum_{j \in r} a_{tj} h_j$, where a_{tj} is the weight given to the jth annotation at time t: $a_{tj} = \frac{\exp(e_{tj})}{\sum_{k \in r} \exp(e_{ik})}$, and $e_{tj} = a_t(h_j)$, where a is the alignment model. F is the encoder which produces the sequence of annotations $(h_1, ..., h_N)$.

Language pairs	Subwords only in source	Subwords only in target	Subwords in both
EN→FR	2K	7K	85K
EN→DE	3K	11K	80K

Table 1: Shared BPE subwords between pairs of languages.

The output of the decoder is then projected to a vector of scores using the output embedding: $l_t = VG^{(t)}$. The scores are then converted to probability values using the softmax function.

In our weight tied translation model, we tie the input and output embeddings of the decoder.

We observed that when preprocessing the ACL WMT 2014 EN→FR[1] and WMT 2015 EN→DE[2] datasets using BPE, many of the subwords appeared in the vocabulary of both the source and the target languages. Tab. 1 shows that up to 90% (85%) of BPE subwords between English and French (German) are shared.

Based on this observation, we propose three-way weight tying (TWWT), where the input embedding of the decoder, the output embedding of the decoder and the input embedding of the encoder are all tied. The single source/target vocabulary of this model is the union of both the source and target vocabularies. In this model, both in the encoder and decoder, all subwords are embedded in the same duo-lingual space.

4 Results

Our experiments study the quality of various embeddings, the similarity between them, and the impact of tying them on the word2vec skip-gram model, NNLMs, and NMT models.

4.1 Quality of Obtained Embeddings

In order to compare the various embeddings, we pooled five embedding evaluation methods from the literature. These evaluation methods involve calculating pairwise (cosine) distances between embeddings and correlating these distances with human judgments of the strength of relationships between concepts. We use: Simlex999 (Hill et al.,

[1] http://statmt.org/wmt14/translation-task.html
[2] http://statmt.org/wmt15/translation-task.html

	Input	Output	Tied
Simlex999	0.30	0.29	0.17
Verb-143	0.41	0.34	0.12
MEN	0.66	0.61	0.50
Rare-Word	0.34	0.34	0.23
MTurk-771	0.59	0.54	0.37

Table 2: Comparison of input and output embeddings learned by a word2vec skip-gram model. Results are also shown for the tied word2vec model. Spearman's correlation ρ is reported for five word embedding evaluation benchmarks.

	PTB			text8		
Embedding	In	Out	Tied	In	Out	Tied
Simlex999	0.02	0.13	0.14	0.17	0.27	0.28
Verb143	0.12	0.37	0.32	0.20	0.35	0.42
MEN	0.11	0.21	0.26	0.26	0.50	0.50
Rare-Word	0.28	0.38	0.36	0.14	0.15	0.17
MTurk771	0.17	0.28	0.30	0.26	0.48	0.45

Table 3: Comparison of the input/output embeddings of the small model from (Zaremba et al., 2014) and the embeddings from our weight tied variant. Spearman's correlation ρ is presented.

A	B	$\rho(A,B)$ word2vec	$\rho(A,B)$ NNLM(S)	$\rho(A,B)$ NNLM(L)
In	Out	0.77	0.13	0.16
In	Tied	0.19	0.31	0.45
Out	Tied	0.39	0.65	0.77

Table 4: Spearman's rank correlation ρ of similarity values between all pairs of words evaluated for the different embeddings: input/output embeddings (of the untied model) and the embeddings of our tied model. We show the results for both the word2vec models and the small and large NNLM models from (Zaremba et al., 2014).

Model	Size	Train	Val.	Test
Large (Zaremba et al., 2014)	66M	37.8	82.2	78.4
Large + Weight Tying	51M	48.5	77.7	74.3
Large + BD (Gal, 2015) + WD	66M	24.3	78.1	75.2
Large + BD + WT	51M	28.2	75.8	73.2
RHN (Zilly et al., 2016) + BD	32M	67.4	71.2	68.5
RHN + BD + WT	24M	74.1	68.1	66.0

Table 5: Word level perplexity (lower is better) on PTB and size (number of parameters) of models that use either dropout (baseline model) or Bayesian dropout (BD). WD – weight decay.

2016), Verb-143 (Baker et al., 2014), MEN (Bruni et al., 2014), Rare-Word (Luong et al., 2013) and MTurk-771 (Halawi et al., 2012).

We begin by training both the tied and untied word2vec models on the text8[3] dataset, using a vocabulary consisting only of words that appear at least five times. As can be seen in Tab. 2, the output embedding is almost as good as the input embedding. As expected, the embedding of the tied model is not competitive. The situation is different when training the small NNLM model on either the Penn Treebank (Marcus et al., 1993) or text8 datasets (for PTB, we used the same train/validation/test set split and vocabulary as (Mikolov et al., 2011), while on text8 we used the split/vocabulary from (Mikolov et al., 2014)). These results are presented in Tab. 3. In this case, the input embedding is far inferior to the output embedding. The tied embedding is comparable to the output embedding.

A natural question given these results and the analysis in Sec. 3 is whether the word embedding in the weight tied NNLM model is more similar to the input embedding or to the output embedding of the original model. We, therefore, run the following experiment: First, for each embedding, we compute the cosine distances between each pair of words. We then compute Spearman's rank correlation between these vectors of distances. As can be seen in Tab. 4, the results are consistent with our analysis and the results of Tab. 2 and Tab. 3: for word2vec the input and output embeddings are similar to each other and differ from the tied embedding; for the NNLM models, the output embedding and the tied embeddings are similar, the input embedding is somewhat similar to the tied embedding, and differs considerably from the output embedding.

[3] http://mattmahoney.net/dc/textdata

4.2 Neural Network Language Models

We next study the effect of tying the embeddings on the perplexity obtained by the NNLM models. Following (Zaremba et al., 2014), we study two NNLMs. The two models differ mostly in the size of the LSTM layers. In the small model, both LSTM layers contain 200 units and in the large model, both contain 1500 units. In addition, the large model uses three dropout layers, one placed right before the first LSTM layer, one between h_1 and h_2 and one right after h_2. The dropout probability is 0.65. For both the small and large models, we use the same hyperparameters (i.e. weight initialization, learning rate schedule, batch size) as in (Zaremba et al., 2014).

In addition to training our models on PTB and text8, following (Miyamoto and Cho, 2016), we also compare the performance of the NNLMs on the BBC (Greene and Cunningham, 2006) and IMDB (Maas et al., 2011) datasets, each of which we process and split into a train/validation/test

Model	Size	Train	Val.	Test
KN 5-gram				141
RNN				123
LSTM				117
Stack RNN	8.48M			110
FOFE-FNN				108
Noisy LSTM	4.65M		111.7	108.0
Deep RNN	6.16M			107.5
Small model	4.65M	38.0	120.7	114.5
Small + WT	2.65M	36.4	117.5	112.4
Small + PR	4.69M	50.8	116.0	111.7
Small + WT + PR	2.69M	53.5	104.9	100.9

Table 6: Word level perplexity on PTB and size for models that do not use dropout. The compared models are: KN 5-gram (Mikolov et al., 2011), RNN (Mikolov et al., 2011), LSTM (Graves, 2013), Stack / Deep RNN (Pascanu et al., 2013), FOFE-FNN (Zhang et al., 2015), Noisy LSTM (Gülçehre et al., 2016), and the small model from (Zaremba et al., 2014). The last three models are our models, which extend the small model. PR – projection regularization.

	Model	Small	S + WT	S + PR	S + WT + PR
text8	Train	90.4	95.6	92.6	95.3
	Val.	-	-	-	-
	Test	195.3	187.1	199.0	183.2
IMDB	Train	71.3	75.4	72.0	72.9
	Val.	94.1	94.6	94.0	91.2
	Test	94.3	94.8	94.4	91.5
BBC	Train	28.6	30.1	42.5	45.7
	Val.	103.6	99.4	104.9	96.4
	Test	110.8	106.8	108.7	98.9

Table 7: Word level perplexity on the text8, IMDB and BBC datasets. The last three models are our models, which extend the small model (S) of (Zaremba et al., 2014).

split (we use the same vocabularies as (Miyamoto and Cho, 2016)).

In the first experiment, which was conducted on the PTB dataset, we compare the perplexity obtained by the large NNLM model and our version in which the input and output embeddings are tied. As can be seen in Tab. 5, weight tying significantly reduces perplexity on both the validation set and the test set, but not on the training set. This indicates less overfitting, as expected due to the reduction in the number of parameters. Recently, (Gal and Ghahramani, 2016), proposed a modified model that uses Bayesian dropout and weight decay. They obtained improved performance. When the embeddings of this model are tied, a similar amount of improvement is gained. We tried this with and without weight decay and got similar results in both cases, with slight improvement in the latter model. Finally, by replacing the LSTM with a recurrent highway network (Zilly et al., 2016), state of the art results are achieved when applying weight tying. The contribution of WT is also significant in this model.

		Size	Validation	Test
EN→FR	Baseline	168M	29.49	33.13
	Decoder WT	122M	29.47	33.26
	TWWT	80M	29.43	33.46
EN→DE	Baseline	165M	20.96	16.79
	Decoder WT	119M	21.09	16.54
	TWWT	79M	21.02	17.15

Table 8: Size (number of parameters) and BLEU score of various translation models. TWWT – three-way weight tying.

Perplexity results are often reported separately for models with and without dropout. In Tab. 6, we report the results of the small NNLM model, that does not utilize dropout, on PTB. As can be seen, both WT and projection regularization (PR) improve the results. When combining both methods together, state of the art results are obtained. An analog table for text8, IMDB and BBC is Tab. 7, which shows a significant reduction in perplexity across these datasets when both PR and WT are used. PR does not help the large models, which employ dropout for regularization.

4.3 Neural Machine Translation

Finally, we study the impact of weight tying in attention based NMT models, using the DL4MT[4] implementation. We train our EN→FR models on the parallel corpora provided by ACL WMT 2014. We use the data as processed by (Cho et al., 2014) using the data selection method of (Axelrod et al., 2011). For EN→DE we train on data from the translation task of WMT 2015, validate on newstest2013 and test on newstest2014 and newstest2015. Following (Sennrich et al., 2016b) we learn the BPE segmentation on the union of the vocabularies that we are translating from and to (we use BPE with 89500 merge operations). All models were trained using Adadelta (Zeiler, 2012) for 300K updates, have a hidden layer size of 1000 and all embedding layers are of size 500.

Tab. 8 shows that even though the weight tied models have about 28% fewer parameters than the baseline models, their performance is similar. This is also the case for the three-way weight tied models, even though they have about 52% fewer parameters than their untied counterparts.

[4] https://github.com/nyu-dl/dl4mt-tutorial

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 278–289. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(1-47).

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*.

Yarin Gal. 2015. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *arXiv preprint arXiv:1512.05287*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Çaglar Gülçehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. 2016. Noisy activation functions. *arXiv preprint arXiv:1603.00391*.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

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

Thang Luong, Richard Socher, and Christopher Manning, 2013. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, chapter Better Word Representations with Recursive Neural Networks for Morphology, pages 104–113. Association for Computational Linguistics.

L. Andrew Maas, E. Raymond Daly, T. Peter Pham, Dan Huang, Y. Andrew Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1992–1997. Association for Computational Linguistics.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL*.

Nitish Srivastava. 2013. Improving Neural Networks with Dropout. Master's thesis, University of Toronto, Toronto, Canada, January.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, Portland, OR, USA, September.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Li-Rong Dai. 2015. A fixed-size encoding method for variable-length sequences with its application to neural network language models. *arXiv preprint arXiv:1505.01504*.

Julian G. Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.