ALTERNATIVE STRUCTURES FOR CHARACTER-LEVEL RNNS

Piotr Bojanowski *
INRIA
Paris, France
piotr.bojanowski@inria.fr

Armand Joulin and Tomáš Mikolov
Facebook AI Research
New York, NY, USA
{tmikolov,ajoulin}@fb.com

ABSTRACT

Recurrent neural networks are convenient and efficient models for learning patterns in sequential data. However, when applied to signals with very low cardinality such as character-level language modeling, they suffer from several problems. In order to successfully model longer-term dependencies, the hidden layer needs to be large, which in turn implies high computational cost. Moreover, the accuracy of these models is significantly lower than that of baseline word-level models. We propose two structural modifications of the classic RNN LM architecture. The first one consists on conditioning the RNN both on the character-level and word-level information. The other one uses the recent history to condition the computation of the output probability. We evaluate the performance of the two proposed modifications on multi-lingual data. The experiments show that both modifications can improve upon the basic RNN architecture, which is even more visible in cases when the input and output signals are represented by single bits. These findings suggest that more research needs to be done to develop general RNN architecture that would perform optimally across wide range of tasks.

1 INTRODUCTION

Modeling sequential data is a fundamental problem in machine learning with many applications, for example in language modeling (Goodman, 2001), speech recognition(Young et al., 1997) and machine translation (Koehn et al., 2007). In particular, for modeling natural language, recurrent neural networks (RNNs) are now widely used and have demonstrated state-of-the-art performance in many standard tasks (Mikolov, 2012).

While RNNs have been shown to outperform the traditional n-gram models and feeedforward neural network language models in numerous experiments, they are usually based on the word level information and thus are oblivious to subword information. For example, RNN language models encode input words such as "build", "building" and "buildings" using 1-of-N coding, which does not capture any similarity of the written form of the words. This can potentially result in poor representation of words that are rarely seen during training. Even worse, the words that appear only in the test data will be not represented at all. This

*Most of the work was done while interning at Facebook AI Research.

problem can become significant when working with languages that have extremely large vocabularies, such as agglutinative languages where words can be created by concatenating morphemes (Finnish and Turkish being well-studied examples). Further, in many real-world applications, typos and spelling mistakes artificially increase the size of the vocabulary by adding several versions of the same word. This requires ad-hoc spell checking approaches that are designed disjointly from the main language modeling task.

To overcome these limitations, we investigate the use of lower-level recurrent neural networks, that model language at the level of characters or even bits. While this type of models has been widely studied in the past (see for example (Mikolov et al., 2011; Sutskever et al., 2011; Graves, 2013)), they lead to both lower accuracy and higher computational cost than word-based models (Mikolov et al., 2012). This drop of performance is unlikely due to the difficulty for character level model to capture longer short term memory, since Longer Short Term Memory (LSTM) recurrent networks also work better with word-based input (Graves, 2013).

Ad-hoc solutions based on the use of sub-word units seem to both deal with new words and offer reasonable accuracy and training speed (Mikolov et al., 2012). However, these approaches have several issues: one has to specify how to create the sub-word units, which can differ from language to language; and a word can have multiple segmentations into the sub-word units.

In this paper, we investigate at first an extension of a standard character-level recurrent neural network (Char-RNN) that includes both word level and character level information. Arguably, such an approach is simpler than the one based on sub-words, and does not have the potential problems mentioned above. Further, we can see that one of the fundamental differences between the word level and character or bit level models is in the number of parameters the RNN has to access during the training and test phase. The smaller is the input and output layer of RNN, the larger needs to be the fully connected hidden layer, which makes the training of the model expensive. Following this observation, we investigate another architecture that does not include the (still somewhat ad-hoc) word level information, and rather attempts to make the computations more sparse. In our experiments, this is achieved by conditioning the probability distribution in the output layer using the recent symbol (character or bit) history. This greatly increases the number of parameters in the model without increasing the size of the hidden layer or the output layer, and thus does not increase the computational complexity.

First, we describe the standard RNN in the context of character prediction problem in Sec. 2, then we propose two different structural modifications of this model. The first modification, described in Sec. 3, combines two networks, one working with characters at the input, and the other with words. The second approach, described in Sec. 4, attempts to increase capacity of the RNN model by conditioning the softmax output on the recent history.

RELATED WORK

Agglutinative languages such as Finnish or Turkish have very large vocabularies, making word based models impractical (Kurimo et al., 2006). Subword units such as morphemes have been used in statstical models for speech recognition (Vergyri et al., 2004; Hirsimäki et al., 2006; Arisoy et al., 2009; Creutz et al., 2007). In particular, Creutz et al. (2007) show that morph-based N-gram models outperform word based ones on most of the agglutinative languages.

A mix of word and character level input for neural network language models has been investigated by Kang et al. (2011) in the context of Chinese. More recently, Kim et al. (2015) propose a model to predict words given character level inputs, while we predict characters based on a mix of word and character level inputs.

Recurrent networks have been popularized for statistical language modeling by Mikolov et al. (2010). Since then, many authors have investigated the use of subword units in order to deal with Out-Of-Vocabulary

(OOV) words in the context of recurrent networks. Typical choice of subword units are either characters (Mikolov et al., 2011; Sutskever et al., 2011; Graves, 2013) or syllables (Mikolov et al., 2012).

Others have used embedding of words to deal with OOV words (Bilmes & Kirchhoff, 2003; Alexandrescu & Kirchhoff, 2006; Luong et al., 2013). Luong et al. (2013) build word embeddings by applying a recursive neural network over morpheme embeddings, while Bilmes & Kirchhoff (2003) build their embedding by concatenating features built on previously seen words.

2 SIMPLE RECURRENT NETWORK

In this section, we describe the simple RNN model popularized by Elman (1990), in the context of language modeling. We formulate language modeling as a discrete sequence prediction problem. That is, we want to predict the next token in a sequence given its past. We suppose a fixed size dictionary of k words formed from d different characters. We denote by c_t the one-hot encoding of t-th character in the sequence, and w_p the one-hot encoding of the p-th word. Our basic unit is the character.

An RNN consists of an input layer, a hidden layer and an output layer. Its hidden layer has a recurrent connection which allows the propagation through time of information. More precisely, the state of the m hidden units, h_t is updated as a function of its previous state, h_{t-1} and the current character one-hot representation c_t:

$$h_t = \sigma(Ac_t + Rh_{t-1}),$$

where $\sigma(x) \mapsto 1/(1+\exp(-x))$ is the pointwise sigmoid function, A is the $m \times d$ embedding matrix and R the $m \times m$ recurrent matrix. This hidden representation is supposed to act as a memory, and should be able to convey long-term dependencies. With sufficiently high-dimensional hidden representation, it should be *a priori* possible to store the whole history. However, using a big hidden layer implies high computational costs which are prohibitive in practice. Using its hidden representation, the RNN compute a probability distribution y_t over the next character:

$$y_t = f(Uh_t),$$

where U is a $d \times m$ matrix and f is the pointwise softmax function, i.e., $[f(x)]_i = \exp(x_i)/\sum_j \exp(x_j)$.

Optimization. In order to learn the parameters $\theta = (A,\ R,\ U)$ of the model, we minimize the negative log-likelihood (NLL):

$$NLL(\theta) = -\sum_{t=1}^{T} c_{t+1}^{\top} \log y_t, \tag{1}$$

with a stochastic gradient descent method and backpropagation through time (Rumelhart et al., 1985; Werbos, 1988). We clip the gradient in order to avoid gradient explosion. The details of the implementation are given in the experiment section.

Character level RNNs have been shown to perform poorly compared to word level ones (Mikolov et al., 2012). In particular, they require a massive hidden layer in order to obtain results which are on par with word level models, this makes them very expensive to compute. In the following sections, we describe two different structural modification of the char-RNNs in order to add capacity while reducing the overall computational cost.

3 CONDITIONING ON WORDS

In this section, we consider an extension of character-level RNN by conditioning it with word level information. This allows a more direct flow of information from the previous words to the character level prediction.

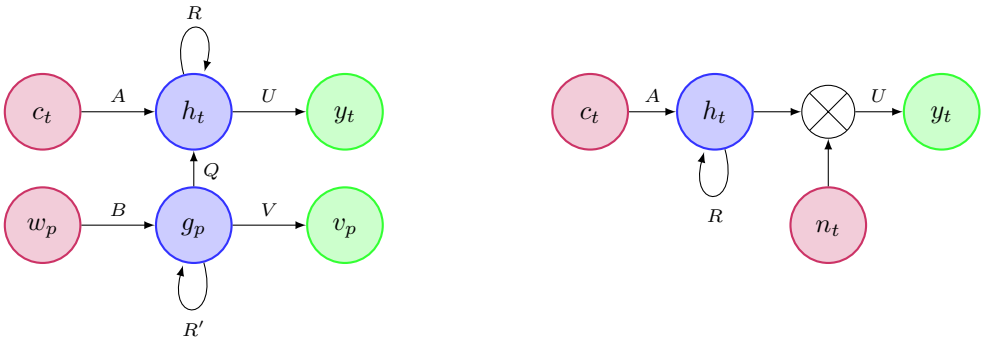

Figure 1: Illustration of the two network modifications that we propose in this paper. **(Left)** mixed model described in Sec. 3. The "faster" character-level network is conditioned on the hidden representations of a "slower", word-level one. **(Right)** conditional model described in Sec. 4. The output at every time step t depends not only on the hidden representation h_t but also on the input history n_t.

We propose to condition the character level on a context vector z_t as follow:

$$h_t = \sigma(Ac_t + Rh_{t-1} + Qz_t), \tag{2}$$

where Q is the conditioning matrix. The context vector z_t is built by gathering information at the word level using a word level RNN, with a similar architecture to the one described in the previous section. Its input for the p-th word is its one-hot representation w_p. The context vector is then simply the state of the hidden layer g_p: if the t-th character belongs to the p-th word, then $z_t = g_{p-1}$. Figure 1 **(Left)** provides an illustration of this hybrid word and character level RNN.

In order to train this model, we combine a loss on characters with a loss on words. However, computing a full softmax at the word level is expensive, in particular when facing large vocabularies. Many solutions have been proposed to reduce the cost of this step, such as hierarchical softmax, or sampling techniques. In this work, we simply restrict the word vocabulary for the output of the word level RNN. We keep the most frequent words (between 3000 and 5000) and associate the other ones with the `<UNK>` token. The loss that we use to train our model can be written as:

$$\text{NLL}(\theta) = -\lambda \sum_{t=1}^{T} c_{t+1}^{\top} \log y_t - (1-\lambda) \sum_{p=1}^{P} w_{p+1}^{\top} \log v_p, \tag{3}$$

where v_p is the prediction made by the word level RNN and $\lambda > 0$ is an interpolation parameter. Note that the restricted vocabulary is only used for the output of the word level RNN, the rest of the model works on the large vocabulary. In the next section, we describe another structural modification that we propose to speed up language modeling on the character level.

4 Conditioning prediction on recent history

In a character based RNNs, the classifier has a very small number of parameters. We propose to condition this classifier on the recent contextual information to increase its capacity while keeping the computational cost constant. This context information is related to simple short-term statistics, such as coocurrence in a language. Explicitly modeling this information in the classifier, removes some of the burden from the hidden layer, allowing to use smaller recurrent matrices while maintaining the performance.

There are many relevant contextual informations on which we can condition the classifier. In particular, simple short term dependencies are easily captured by character n-grams, which are memory expensive but very efficient. Using such cheap information to condition the classifier of the RNN gives a simple way to increase the capacity of the model while encouraging the rest of the RNN to focus on more complex statistical patterns. Conditioning the classifier on n-grams can be written as a bilinear model. More precisely, denoting by n_t the one-hot representation of the t-th n-gram, the prediction of our model is:

$$y_t = f(n_t^\top U h_t),$$

where U is a tensor going from the product space of hidden representation and n-grams to characters. This alternative architecture is depicted in Fig. 1 **(Right)**.

Obviously, the exponential number of n-grams makes this model impossible to learn. Worse, it is impossible to generalize at test time to unseen n-grams. To avoid these problems, we restrict our set of n-grams $\mathcal{N}$ to those that contain less than N symbols and appear frequently enough in the training set. If multiple n-grams can be used at the t-th character of the text, we fix n_t to be the longest one appearing in $\mathcal{N}$. This insures that each n-gram n_t is associated with enough examples to learn a statistically meaningful tensor U. The great thing with this solution is the fact that apart from characters that don't appear in the training set, we always have a non-trivial output model at test time. In the worst case, this procedure will select the model corresponding to the last unigram. The n-gram cut-off frequency allows to control the overfitting of the model.

5 Experimental evaluation

We evaluate the two proposed models on the Penn Treebank corpus and on a subset of the Europarl dataset. These experiments are described in Sec. 5.1 and Sec. 5.2. As a proof of concept, we illustrate the use of the conditioned model on the binary representation of Penn Treebank. We evaluate our model on the bit prediction task in the experiments described in Sec. 5.3.

For the sake of simplicity, we compare our models to a plain RNN but all the modifications that we propose can be applied to more complex units (LSTM *etc.*). We train our model using stochastic gradient descent and select hyper parameters on a validation set. We set a constant learning rate γ and when the validation entropy starts to increase, we divide it by a factor α after every epoch (values arround $\gamma = 0.1$ and $\alpha = 1.5$ work well in practice). Our implementation is a single threaded CPU code which could easily be parallelized. Code for training both models is publicly available[1].

We evaluate our method using entropy in bits per character. It is defined as the empirical estimate of the cross-entropy between the target distribution and the model output in base 2. This corresponds to the negative log likelihood that we use to train our model up to a multiplicative factor: $BPC(\theta) = \frac{1}{T \log(2)} NLL(\theta)$.

5.1 Experiments on Penn Treebank

We first carry out experiments on the Penn Treebank corpus (Marcus et al., 1993). This is a dataset with a training set composed of 930k normalized words, yielding a total of 5017k characters. All characters are in the ASCII format which leads to a limited size of character vocabulary C. The text was normalized and the word dictionary limited to 10000 most frequent words of the training set. The other words were replaced by a `<UNK>` token in the training, validation and testing sets.

We evaluate both models described in this paper on the Penn Treebank dataset. For the mixed model from Sec. 3 (Mixed), we fix size of the word-level hidden representations to 200. For the conditional model

[1] https://github.com/facebook/Conditional-character-based-RNN

Table 1: Performance of the proposed models as compared to a classical Char-RNN (CRNN) on Penn Treebank. For all methods we report the entropy in bits per character on the validation set and the corresponding training time for one epoch.

	val BPC			training time (s / epoch)		
m	CRNN	Mixed	Cond.	CRNN	Mixed	Cond.
100	1.86	1.73	1.51	166	613	250
200	1.63	1.50	1.48	527	1152	707
300	1.53	1.43	1.47	1061	1838	1360
500	1.46	1.40	1.46	2645	3722	3237
1000	1.42	N/A	N/A	9752	N/A	N/A

presented in Sec. 4 (Cond.), we choose the optimal N on the validation set. We compare these models with our own implementation of a character-level RNN as it allows us to fairly compare run times (a significant part of the code is shared). All models are trained for various sizes of the hidden layer m. We report entropy in bits per character on the validation set and the training time per epoch in Table 1.

The character-level performance we obtain for the "vanilla" RNN is coherent with numbers published in the past on this dataset (Mikolov et al., 2012). We observe three important things: **(a)**, for any size of character hidden layer, both proposed models perform better than the plain one. This of course comes at the expense of some additional computational cost and the benefits seem to decrease when the size of h_t grows. **(b)** using this model, we manage to obtain a comparable performance to the heavy 1000-dimensional character RNN with a hidden representation of only 300. This corresponds to an important reduction in the number of recurent parameters and to a five times speedup per epoch at training time. **(c)** when the hidden representation is small, the best working model is the conditional one. However, when m gets larger, the mixed one seems to work best, and provides competitive entropy for a reasonable runtime.

For the conditional model, we observed in our experiments, that there seems to be a clear trade off in the choice of N, reached on Penn Treebank at roughly $N = 1000$. Indeed, in the limit case, when N is very large, we only have one output model, and the network is exactly equivalent to a RNN. On the other hand, when N is small, we keep a separate model for any sequence and therefore overfits to the training set.

5.2 The multilingual Europarl dataset

We perform another set of experiments on the Europarl dataset (Koehn, 2005). It is a corpus for machine translation with sentences from 20 different languages aligned with their English correspondence. For almost every language, there is more than 500k sentences, composed of more that 10M words. Because of its size, we restrict our experiments to a subset of sentences for each language. We randomly permute lines of the transcriptions, select 60k sentences for training, 10k for validation and 10k for testing. The permutation we use will be made publicly available upon publication.

In this experiment, as in the one described in Sec. 5.1, we compare our models to a character-level RNN. We train our mixed model with a word hidden of 200 and a character hidden of 300. For the conditional one, we fix the hidden representation and select the optimal N on the validation set. As a baseline, we train a character-level RNN for two sizes of hidden layers: 200 and 500. These results are summarized in Table 2, where we group "light" and "heavy" models together. We also report the word dictionary size (k) and out of vocabulary rate (OOVR) for every language.

The CRNN baseline as well as the proposed models still are quite far from the performance of a word-level RNN. As we see in Table 2, the average performance of both proposed models give a per-character entropy

Table 2: Results on the Europarl dataset. For all languages, we report the word vocabulary size and the out-of-vocabulary rate on the validation set. We report the performance of the mixed model (**Mixed**) and the conditional model (**Cond.**). We compare to a character-level RNN (**CRNN**) with hidden representations of size 200 and 500.

			large models		light models	
language	Vocab. size	OOVR	CRNN_{500}	Mixed	CRNN_{200}	Cond.
Bulgarian	109 k	1.87	1.28	1.26	1.52	1.27
Czech	144 k	3.02	1.54	1.53	1.79	1.52
Danish	128 k	2.78	1.37	1.36	1.62	1.37
German	136 k	2.78	1.32	1.31	1.53	1.30
Greek	132 k	2.24	1.28	1.27	1.55	1.27
Spanish	105 k	1.71	1.28	1.26	1.50	1.27
Estonian	190 k	5.29	1.48	1.50	1.72	1.47
Finnish	227 k	6.91	1.39	1.43	1.63	1.38
French	105 k	1.67	1.24	1.23	1.48	1.24
Hungarian	208 k	5.05	1.39	1.42	1.65	1.36
Italian	115 k	1.95	1.30	1.29	1.52	1.29
Lithuanian	163 k	4.25	1.45	1.46	1.69	1.45
Latvian	138 k	3.06	1.42	1.41	1.65	1.42
Dutch	109 k	2.09	1.33	1.32	1.56	1.31
Polish	153 k	3.13	1.41	1.39	1.66	1.39
Portuguese	110 k	1.88	1.33	1.30	1.54	1.32
Romanian	104 k	1.66	1.29	1.26	1.54	1.27
Slovak	145 k	2.95	1.47	1.45	1.74	1.46
Slovene	132 k	2.66	1.47	1.44	1.70	1.47
Swedish	130 k	2.85	1.40	1.39	1.64	1.38
Average			1.37	1.36	1.61	1.36

of 1.36. The proposed structural modifications allow us to achieve similar performance to a large character-level RNN with a reduced computational cost. For languages such as Finnish and Hungarian, the conditional model (**Cond.**) yields best performance.

For reference, we computed a word-level RNN baseline using a modified version of SCRNN[2]. If we assign 4 times the average entropy to OOV words, it gives us an entropy of 1.27 BPC. The proposed models allow us to efficiently tackle the problem of learning small vocabulary sequences. However, the gap between the word and character-level models is far from being closed.

5.3 Binary Penn Treebank

We aditionnaly carry out some experiments on the binary representation of Penn Treebank. We train the network for the task of bit prediction: given a sequence of t bits, the goal is to predicth the $t + 1$-th. As mentionned in the introduction, we want to develop models that are independent of the representation in use. Working with binary representation would allow to have models of sequential data that would be agnostic to the nature of the sequence. This could straightforwadly be applied to language modelling but also speech recognition directly from wave files *etc.*We run the conditional model and a baseline bit-level RNN, both with a hidden representation of 100. For the conditional model, we select the optimal n-gram history N by choosing it on the validation set ($N = 2000$). We evaluate both models by computing the entropy per bit

[2]https://github.com/facebook/SCRNNs

Table 3: Comparison of the conditional model to a RNN for the binary representation of Penn Treebank. We provide results in Bits Per Bit and Bits per character. The conditional model outperforms the plain RNN by a large margin for a fixed size of hidden representation.

	BPB		BPC	
model	val	test	val	test
CRNN	0.287	0.282	2.29	2.25
Cond.	0.222	0.216	1.78	1.73

and per character and report performance on the validation and test sets. The results for this experiment are presented in Table 3.

This setting corresponds to the extreme case where the dictionary is as small as it could be. When the input and output dictionaries are so small, the number of parameters is also small and mostly depend on m, the size of the hidden representation. In the case of a classical bit-level RNN, the input and output model only have $2 \times m$ parameters which can be a serious limitation for RNNs. We empirically observe that to make these models work, one needs to use a very large hidden layer. As we see in Table 3, the conditional model works much better as it compensates this small output model by storing several ones instead.

CONCLUSION

In this work we investigated modifications of RNNs for general discrete sequence prediction when the number of symbols is very small, such as in char-RNNLM. We found that with certain tricks, one can train the model much faster, and overall we observed that the fully connected RNN architecture has its weaknesses, especially related to the excessive computational complexity. We believe more research is needed to develop general mechanisms that would allow us to train RNNs with richer internal structure. We expect such research can greatly simplify many pipelines, for example in the NLP applications where we could avoid having separate systems that perform spell checking, text normalization, and modeling of the language disjointly.

We hope that this work will open up new research paths for modeling sequences with small vocabularies. Initial yet promising results on binary representations of Penn Treebank show that RNNs can be trained on that kind of data. We believe that this allows us to define models for sequence modeling that would be agnostic to the nature of the input. Bit-level models could be used on any sequential data, for example speech signal in binary .wav form.

REFERENCES

Andrei Alexandrescu and Katrin Kirchhoff. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 1–4. Association for Computational Linguistics, 2006.

Ebru Arisoy, Doğan Can, Siddika Parlak, Haşim Sak, and Murat Saraçlar. Turkish broadcast news transcription and retrieval. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(5):874–883, 2009.

Jeff A Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguis-*

tics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2, pp. 4–6. Association for Computational Linguistics, 2003.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(1):3, 2007.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Joshua T Goodman. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434, 2001.

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

Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. Unlimited vocabulary speech recognition with morph language models applied to finnish. *Computer Speech & Language*, 20(4):515–541, 2006.

Moonyoung Kang, Tim Ng, and Long Nguyen. Mandarin word-character hybrid-input neural network language model. In *INTERSPEECH*, pp. 625–628, 2011.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*, 2015.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pp. 79–86, 2005.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pp. 177–180. Association for Computational Linguistics, 2007.

Mikko Kurimo, Antti Puurula, Ebru Arisoy, Vesa Siivola, Teemu Hirsimäki, Janne Pylkkönen, Tanel Alumäe, and Murat Saraclar. Unlimited vocabulary speech recognition for agglutinative languages. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pp. 487–494. Association for Computational Linguistics, 2006.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104, 2013.

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

Tomas Mikolov. *Statistical Language Models based on Neural Networks*. PhD thesis, PhD thesis, Brno University of Technology., 2012.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 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*, pp. 1045–1048, 2010.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. Rnnlm-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pp. 196–201, 2011.

Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and J Cernocky. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 2012.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.

Dimitra Vergyri, Katrin Kirchhoff, Kevin Duh, and Andreas Stolcke. Morphology-based language modeling for arabic speech recognition. In *INTERSPEECH*, volume 4, pp. 2245–2248, 2004.

Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. *The HTK book*, volume 2. Entropic Cambridge Research Laboratory Cambridge, 1997.