Improving Language Model Adaptation using Automatic Data Selection and Neural Network

Shahab Jalalvand
HLT research unit, FBK, 38123 Povo (TN), Italy
jalalvand@fbk.eu

Abstract

Since language model (LM) is very sensitive to domain mismatch between training and test data, using a group of techniques to adapt a big LM to specific domains is quite helpful. In this paper, we, benefit from salient performance of recurrent neural network to improve domain adapted LM. In this way, we first apply an automatic data selection procedure on a limited amount of in-domain data in order to enrich the training set. After that, we train a domain specific N-gram LM and improve it by using recurrent neural network language model trained on limited in-domain data. Experiments in the framework of EUBRIDGE [1] project on weather forecast dataset show that the automatic data selection procedure improves the word error rate around 2% and RNNLM makes additional improvement over 0.3%.

Keywords: Language model, automatic data selection, neural network language model, speech recognition

1 Introduction

Language models are widely used in different applications such as automatic speech recognition (ASR), machine translation; spell checking, handwriting detection etc. Basically, a language model tries to predict the next word in a sentence by considering a history of previous words. To provide this history, the language model needs to be trained on a large set of texts. Generally, the larger train set the better language model.

A main issue in language modeling arises from data sparseness in training set. It means that in a large training set, many of the n-grams[2] are very rare and, consequently, their probabilities are very small. Katz (1987) tried to overcome this problem by proposing back-off technique. In it, the probabilities of rare n-grams are estimated through linear interpolation of the probabilities of the lower order n-grams.

Discounting methods such as Witten-Bell estimate (Witten and Bell, 1991), absolute discounting (Ney and Essen, 1991), Kneser-Ney method (Kneser and Ney, 1995) and modified Kneser-Ney (Chen and Goodman, 1999) allow estimating back-off coefficients.

Recently, using neural network language model (NNLM) has been become of interest because it results more generalization in comparison to N-gram models. In NNLM, the words are represented in a continuous space. The idea of representing words in a continuous space for language modeling was started by Bengio (2003). It was followed by Schwenk (2007) who applied neural network for language modeling in large scale vocabulary speech recognition and obtained a noticeable improvement in word error rate. Mikolov (2010) pursued this way and used recurrent neural network for language modeling (RNNLM). The advantage of RNNLM on feed forward neural network, which was used by Bengio (2003) and Schwenk (2007) is that RNNLM can consider an arbitrary number of preceding words to estimate the probability of

[1] This work has been partially founded by the European project EUBRIDGE, under the contract FP7-287658

[2] Sequence of *n* words (usually 2, 3 or 4 words). By n-gram (with small "n") we refer to an n-word sequence and by N-gram (with capital "N") we refer to a language model based on n-grams

next word, while, feed forward NNLM can only see a fixed number of preceding words. Thanks to positive performance of RNNLM toolbox developed by Mikolov (2011), we use this, in our specific task which is weather forecast transcription in the framework of EUBRIDGE project.

In addition to data sparseness, the performance of language model is affected by mismatch between training and test data. This leads to reduction of language model accuracy. The problem is that it is not always easy to collect sufficient amount of related data in order to train a specific-domain LM. Therefore, research on LM domain adaptation, as well as automatic selection of auxiliary text data is still of large interest.

There are methods which try to adapt a big language model to a limited amount of in-domain data such as Latent Semantic Analysis (Bellegarda, 1988), Mixture (Foster, 2007), Minimum Discrimination information (Federico, 1999) and Lazy MDI (Ruiz, 2012). Another group of methods try to automatically retrieve auxiliary documents from text resources such as Internet. Among them, we are interested in the ones reported in (Maskey, 2009) and (Falavigna, 2012) which are based on information retrieval measures such as LM perplexity and Term Frequency Inverse Document Frequency (TF-IDF).

This paper aims at transcribing a weather forecast speech corpus consisting of audio recordings that are divided into development and test sets. In addition, a small text corpus of weather forecast has been given within EUBRIDGE project. We use this corpus as in-domain data. In this way, we first utilize an automatic data selection procedure to collect more an auxiliary data set. Then, we train an N-gram language model on the selected data and decode the test audio recording. For each audio, an n-best list is produced which is then processed and re-ranked by means of a neural network language model. We show the N-gram which is trained on the automatically selected data is around 2% (in terms of word error rate) better than the original one and neural network language model improves it up to 0.3%.

In Section 2 and 3, we briefly describe Neural Network Language Model (NNLM) and Recurrent NNLM, respectively. Then, in section 4 we describe the process of preparing data and also the experiments which are confirmed by perplexity and WER results. Finally, Section 5 concludes the paper.

2 Neural Network Language Model (NNLM)

In NNLM, a word is represented by a $|V|$-dimensional vector of 0s and 1s. $|V|$ is the size of vocabulary. In $vector_{wi}$ that represents i^{th} word in the vocabulary, all the elements are zero except i^{th} element which is 1 (see Figure 1). For a 4-gram NNLM, three vectors are concatenated and given to the input layer. Thus, the input vector would be $3x|V|$-dimensional and the input layer has the same number of neurons.

Usually there is a projection layer with linear activation function which reduces the dimension of input vectors and maps them into a continuous space. The output of the projection layer is given to a hidden layer with nonlinear activation function (sigmoid, hyperbolic tangent etc). The output of hidden layer is then given to the output layer which has $|V|$ neurons for $|V|$ candidate words. j^{th} neuron in this layer computes the probability of observing j^{th} word after three previous words (in 4-gram NNLM). The activation function that is used in this layer is a softmax function which guarantees that the sum of all probabilities is 1 and each probability is between zero and 1 (Schwenk, 2007).

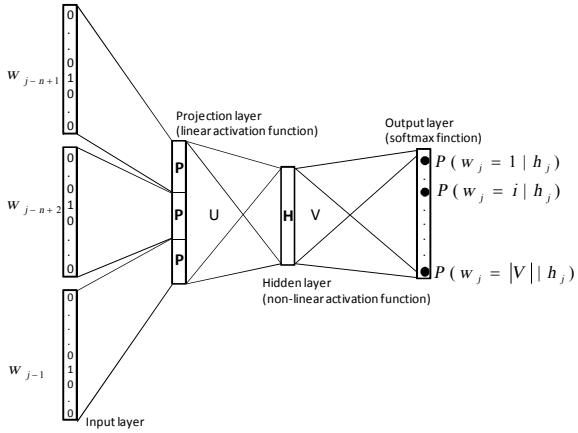

Figure 1: Neural network LM

The computations that are needed for each layer are as follows:

$$d_j = \tanh\left(\sum_l c_l . U_{jl} + b_j\right) \quad \forall j = 1,..., H, \quad (1)$$

$$o_i = \sum_j d_j . V_{ij} + k_i \quad \forall i = 1,..., N, \quad (2)$$

$$p_i = \frac{e^{o_i}}{\sum_{l=1}^{N} e^{o_i}} \quad \forall i = 1,..., N, \quad (3)$$

d_j is the output of j^{th} neuron in projection layer. U and V are the weight matrices from pro-

jection to hidden and from hidden to output layers, respectively. *b* and *k* are the bias vectors of hidden and output layers, respectively. o_i shows the output of i^{th} output neuron. The training procedure is done using a back-propagation algorithm (Schwenk, 2007).

3 Recurrent Neural Network Language Model (RNNLM)

Instead of projection layer, in RNNLM, there are recursive arcs in hidden layer which connect the outputs of hidden neurons to their input and work as a cache memory for neural network.

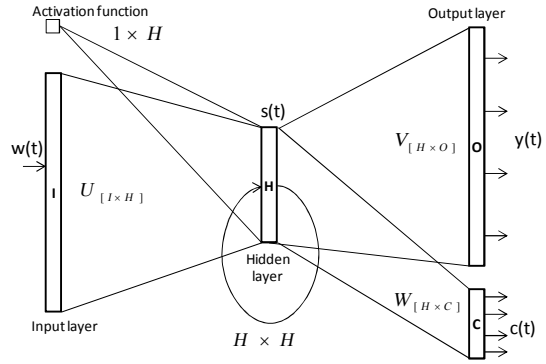

Figure 2: Recurrent Neural Network LM

For a training set with *I* unique words, an input layer with *I* neurons is needed. If the size of hidden layer is $|H|$, then the weight matrix between input and hidden layers (U) will be $I \times |H|$-dimensional. Since the hidden neurons are fully connected by the recursive arcs, there are $|H| \times |H|$ additional weighted connections. Furthermore, we need a $1 \times |H|$-dimensional vector to store the activation function of each hidden neuron.

In a class based language model, there are two types of output: probability of classes and probability of words. To implement a class-based RNNLM, two sets of neurons in the output layer are needed: one for computing the probabilities of words and the other for the probabilities of classes. From hidden neurons to word output neurons there are $|H| \times |O|$ connections, which are shown in matrix V and from hidden neurons to class output neurons there are $|H| \times |C|$ connections which are shown in matrix W (the number of classes is equal to $|C|$).

Considering this architecture for neural network language model, the formulation of each layer should be changed as follows:

$$x(t) = \left[w(t)^T \; s(t-1)^T \right]^T \quad (4)$$

$x(t)$, that is the input vector of hidden layer is a $|V|+|H|$-dimensional vector; $w(t)$ is the vector of observed word at time t; $s(t-1)$ is the output of hidden layer at time $t-1$ ($s(0)$ can be initialized by 0.1). The output of the hidden layer is computed by:

$$s_j(t) = \tanh\left(\sum_i x_i(t).U_{ji} \right) \quad \forall j = 1,..., H \quad (5)$$

In which, $s_j(t)$ is the output of j^{th} hidden neuron. $x_i(t)$ is i^{th} element of input vector and U_{ji} indicates the weight of the connection between neuron i and neuron j from input to hidden layer, respectively. The probability over the classes is computed by:

$$c_l(t) = SOFTMAX\left(\sum_j s_j(t).W_{lj} \right) \quad (7)$$

In which $c_l(t)$ is the output of l^{th} output neuron which shows the probability of class l for the word which has been observed at time t. w_{lj} is the weight of the connection between j^{th} neuron of hidden layer and l^{th} neuron of output layer. Using a similar equation just by replacing matrix W by matrix V, we can compute the probability of each word over the classes.

$$y_c(t) = SOFTMAX\left(\sum_j s_j(t).V_{cj} \right) \quad (8)$$

Therefore, the overall probability of a word is computed by:

$$p(w_i \mid history) = p(c_i \mid s(t)) P(w_i \mid c_i, s(t)) \quad (9)$$

where i varies from 1 to the number of vocabulary size. c_i is the class that w_i belongs to that.

Because of the complexity of this model, it is quite hard to use it for huge text corpora. This is why, researchers usually use this model on small training sets or sometimes they partition a huge training set into several small sets and build an RNNLM on each partition and make an interpolation between them.

In the next experiments we train an RNNLM on the small in-domain data and use it to re-score the output of the speech decoder. We show that this approach improves the WER of the decoder up to 0.3%.

4 Experiments

As previously mentioned, we are given a quite small set of in-domain data, consisting of weather forecast texts (around 1 Million words) and a large, out-domain corpus, called GoogleNews that includes around 1.6G words. There are two major challenges:

- First, training a language model on a large domain-independent set is very costly in time and computation and also the resulted model cannot be very efficient in our specific task which is weather forecast transcription.
- Second, the available domain-specific data is to some extent small and the model which is trained on it is not general enough.

Two possible solutions are:

- We can use the available in-domain set to select similar sentences from the huge out-domain set in order to enrich our in-domain training set.
- Or, we can cluster the domain-independent set using word similarity measures. It is expected that the sentences from the same cluster belong to the same domain. Then, we can train a specific language model for each cluster.

We focus on the first solution and utilize it in our experiments. This idea is already proposed by Maskey (2009) for re-sampling an auxiliary data set for language model adaptation in a machine translation task. We use a similar approach to collect in-domain sentences from GoogleNews.

4.1 Text Corpora and Language Models

The source used for generating the documents for training a domain-independent LM is Google-news. Google-news is an aggregator of news provided and operated by Google, that collects news from many different sources, in different languages, and each group of articles consists of similar contents. We download daily news from this site, filter-out useless tags and collect texts. Google-news data is grouped into 7 broad domains (such as economy, sports, science, technology, etc). After cleaning, removing double lines and application of a text normalization procedure, the corpus results into about 5.7M of documents, or a total of about 1.6G of words. The average number of words per document is 272 (refer to (Girardi, 2007) for details about the web document retrieval process applied in this work).

On this data we trained a 4-gram back-off LM using the modified shift beta smoothing method as supplied by the IRSTLM toolkit (Federico, 2008). The LM results into about 1.6M unigrams, 73M bigrams, 120M 3-grams and 195M 4-grams. The LM is used to compile a static Finite State Network (FSN) which includes LM probabilities and lexicon for two ASR decoding passes. In the following we will refer to this LM as GN4gr-ALL.

Within the EUBRIDGE project we were also given a set of in-domain text data, specifically around 1M words related to weather reports published on the BBC web site, that was first used to train a corresponding 4-gram LM (in the following we will call it IN4gr-1MW).Then, with the latter LM we automatically select, using perplexity as similarity measure, from the whole Google-news database an auxiliary corpus of about 100M words. On this corpus we trained a corresponding 4-gram LM and we adapted it to the weather domain using the 1MW in-domain corpus (as adaptation data) and LM-mixture (as adaptation method). The resulting adapted LM contains about 278K unigrams, 9.4M bigrams, 7.9M 3-grams and 9.5M 4-grams. In the following we will refer to it as IN4gr-100MW.

Using the last language model (IN4gr-100MW) and a pre-trained acoustic model which is described in the next subsection we extract the 1000-best list from the decoder and re-score this list using a recurrent neural network language model (RNNLM).

Before that, we need to investigate different types of RNNLM with different configuration in order to find the best one for our specific task. In this way, we trained RNNLMs with 250, 300, 350, 400, 450, 500 hidden neurons and 200, 300, 500, 600, 1000 and 8000 classes on the 1MW in-domain data. Figure 4 compares the perplexity of these models on a development set consisting of 12K words which is completely isolated from the test set.

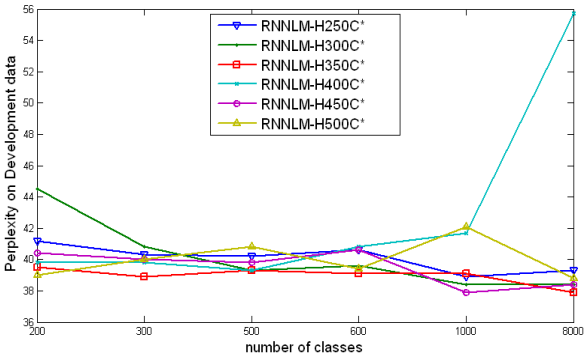

Figure 3. Perplexity of different RNNLMs on development data

As it can be seen from Figure 4, by increasing the number of classes the performance of RNNLM improves. For example, the best three RNNLMs are the ones with: H350C8000,

H450C1000 and H300C1000 (exp. rnnlmH300C1000 is an RNNLM with 300 hidden neurons and 1000 classes).

In accordance with Mikolov (2011), RNNLM works better when it is interpolated with an N-gram. Thus, we train a 4-gram language model based on Kneser-Ney smoothing method using SRI toolkit (Stolcke, 2002) and interpolate it with the best RNNLMs by different weights (lambda). Figure 5 shows the result of these interpolations.

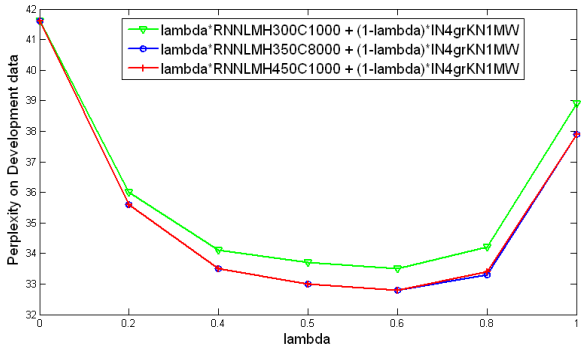

Figure 4. Interpolation of RNNLM scores and 4-gram scores

When lambda is zero, just N-gram score has been considered and when lambda is 1, just the score of RNNLM is used. It is seen that interpolation of N-gram and RNNLM improves the performance of the system. Correspondingly, we see that rnnlmH350C8000 and rnnlmH450C1000 show the highest performance in interpolation with IN4grKN-1MW. In following, we will use the latter to re-score the n-best list obtained from decoder.

4.2 Generation of N-best Lists

As previously mentioned we used the RNNLM, trained on 1MW in-domain set of data, to re-score n-best lists produced during ASR decoding. Details on both acoustic model training and ASR decoding process can be found in (Falavigna, 2012). In short for this work, speech segments to transcribe have been manually detected and labeled in terms of speaker names (i.e. no automatic speech segmentation and speaker diarization procedures have been applied).

In both first and second decoding passes the system uses continuous density Hidden Markov Models (HMMs) and a static network embedding the probabilities of the baseline LM. A frame synchronous Viterbi beam-search is used to find the most likely word sequence corresponding to each speech segment. In addition, in the second decoding pass the system generates a word graph for each speech segment. To do this, all of the word hypotheses that survive inside the trellis during Viterbi beam search are saved in a word lattice containing the following information: initial word state in the trellis, final word state in the trellis, related time instants and word log-likelihood. From this data structure and given the LM used in the recognition steps, WGs are built with separate acoustic likelihood and LM probabilities associated to word transitions. To increase the recombination of paths inside the trellis and consequently the densities of the WGs, the so called word pair approximation is applied. In this way the resulting graph error rate was estimated to be around 1/3 of the corresponding WER.

The best word sequences generated in the second decoding pass are used to evaluate the baseline performance. Instead, the corresponding word graphs are used to generate lists of 1000 sentences each. To do this a stack decoding algorithm is employed (Hart, 1972), where the score of each partial theory is given by summing the forward score of the theory itself with the total backward score in the final state of the same theory (i.e. the look-ahead function used in the algorithm is the total backward probability associated to the final state of the given theory). Finally, each 1000-best list is re-scored using the RNNLM trained on 1MW in-domain text data set. Note that in this latter decoding step, acoustic probabilities remain unchanged, i.e. the latter decoding step implements a pure linguistic rescoring.

4.3 Speech Recognition Results

An overview of the experiments has been given in Figure 5. The first set of results is obtained by using GN4gr-ALL language model which is trained on whole Google-news data. Then, a small N-gram (IN4gr-100MW) is trained on the in-domain data that is used in the procedure of automatic data selection (see section 4.1). Utilizing the resulted data set, a bigger model (IN4gr-100MW) is trained and adapted to the in-domain data.

Thus, the second and third set of results is obtained by using IN4gr-1MW and IN4gr-100MW along with the decoder. In order to improve the final results, we use rnnlmH450C1000 which is trained on in-domain data to re-score the 1000-best list extracted from previous decoding phase.

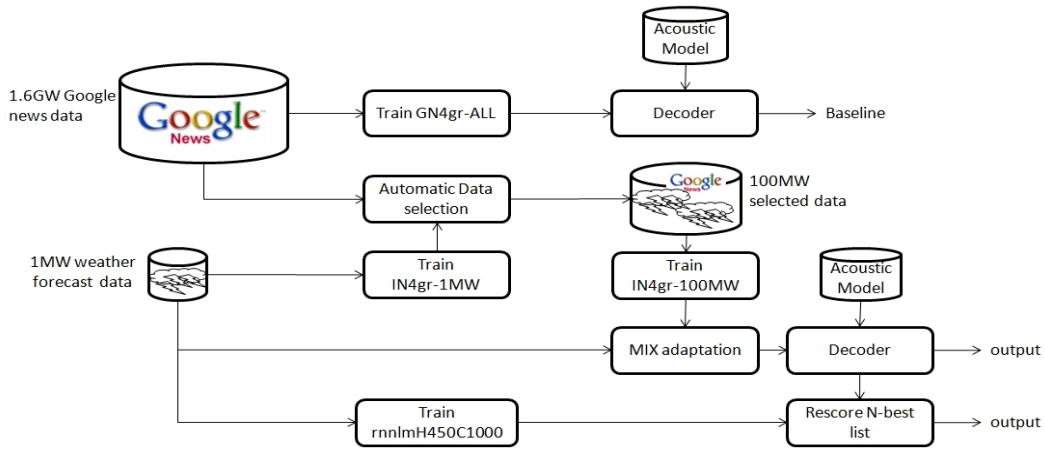

Figure 5. An overview of the speech recognition system

Table 1. compares the WER resulted from using these language models in the decoding phase. It can be seen that the in-domain language model which is trained on the small set of in-domain text is dramatically better than the huge out-domain model. By applying automatic data selection approach and collecting the most useful texts from Google-news we obtained 0.3% improvement and by utilizing RNNLM for re-scoring the n-best lists we reach another 0.3% improvement in word error rate.

Table 1. %WER with different language models (Oracle Error Rate is 9.7%)

Language model	Development set	Test set
GN4gr-ALL	16.2	15.1
IN4gr-1MW	14.3	12.8
IN4gr-100MW	14.0	12.6
0.5*IN4gr-100MW + 0.5*rnnlmH450C1000	**13.7**	**12.3**

Although it's not a salient improvement from the third to fourth row of the table, we should notice that the RNNLM model has re-scored an N-best list, which in the best conditions, it gives 9.7% WER. That is, if we ideally select the best sentences from these n-best lists we cannot reach better result than 9.7%.

5 Conclusion

Given a small set of in-domain data and a huge out-domain corpus, we proposed a thorough system which applies an automatic data selection approach to train a general in-domain language model. In addition, we used a continuous space language model to improve the generality of the model and consequently to improve the accuracy of ASR.

In future, we will benefit from RNNLM in the procedure of data selection. That is, instead of evaluation of candidate sentences using N-gram, we will rank them using RNNLM.

Moreover, it would be worthwhile to explore the performance of a group of small RNNLM on the selected data rather than a single N-gram LM.

Acknowledgments

I would like to express my special thanks of gratitude to my supervisor, Daniele Falavigna, who kindly helped me to do the experiments and write this paper.

References

Andreas Stolcke. 2002. *SRILM - An Extensible Language Modeling Toolkit.* In Proceedings of the International Conference on Statistical Language Processing, Denver, Colorado.

Christian Girardi. 2007. *Htmcleaner: Extracting Relevant Text from Web. 3rd Web as Corpus workshop (WAC3),* Presses Universitaires de Louvain, pp. 141-143.

Daniele Falavigna, Roberto Gretter, Fabio Brugnara, and Diego Giuliani. 2012. *Fbk @ iwslt 2012 - ASR Track.* in Proc. of the International Workshop on Spoken Language Translation, Hong Kong, HK.

George Foster and Roland Kuhn. 2007. *Mixture Model Adaptation for SMT.* In Proceedings of the Second Workshop on Statistical Machine Translation, StatMT '07, pages 128–135, Stroudsburg, PA, USA. association for Computational Linguistics.

Hermann Ney, Ute Essen. 1991. *On Smoothing Techniques for Bigram-based Natural Language Modelling.* Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing '91, volume 2, pp. 825–829.

Holger Schwenk. 2007. *Continuous Space Language Models*. in Computer Speech and Language, volume 21, pp. 492-518.

Ian H. Witten and Timothy C. Bell. 1991. *The Zero-frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression*. IEEE Transactions on Information Theory, 37, pp. 1085–1094.

Jerome R. Bellegarda. 1998. *A Multispan Language Modeling Frame-work for Large Vocabulary Speech Recognition*. IEEE Transactions on Speech and Audio Processing, vol. 6, no. 5, pp. 456–467.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. *IRSTLM: an Open Source Toolkit for Handling Large Scale Language Model*. in Proc. Of INTERSPEECH, Brisbane, Australia, pp. 1618–1621

Marcello Federico. 1999. *Efficient Language Model Adaptation Through MDI Estimation*. In Proceedings of the 6th European Conference on Speech Communication and Technology, vol. 4, Budapest, Hungary, pp. 1583–1586.

Nick Ruiz and Marcello Federico. 2012. *MDI Adaptation for the Lazy: Avoiding Normalization in LM Adaptation for Lecture Translation*. In Proceedings of the International Workshop on Spoken Language Translation, Hong Kong, China.

Peter E. Hart. Nils J. Nilsson. Bertram Raphael. 1972. *Correction to A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. SIGART Newsletter 37: 28–29

Sameer Maskey, Abhinav Sethy. 2009. *Resampling Auxiliary Data for Language Model Adaptation in Machine Translation for Speech*. in ICASSP 2009, Taiwan

Stanley F. Chen and Jushua Goodman. 1999. *An Empirical Study of Smoothing Techniques for Language Modeling*. Computer Science and Language, 4(13), pp. 359-393.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, Jan Cernocky. 2011. *Empirical Evaluation and Combination of Advanced Language Modeling Techniques*. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011). Florence, IT.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Honza Cernocky, Sanjeev Khudanpur. 2010. *Recurrent Neural Network Based Language Model*. In Proc. INTERSPEECH2010. pp. 1045–1048

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. *A Neural Probabilistic Language Model*. In journal of machine learning research 3, pp. 1137-1155.