Bag-of-Words Input for Long History Representation in Neural Network-based Language Models for Speech Recognition

Kazuki Irie[1], *Ralf Schlüter*[1], *Hermann Ney*[1,2]

[1]Human Language Technology and Pattern Recognition,
Computer Science Department, RWTH Aachen University, Aachen, Germany
[2]Spoken Language Processing Group, LIMSI CNRS, Paris, France
{irie,schlueter,ney}@cs.rwth-aachen.de

Abstract

In most of previous works on neural network based language models (NNLMs), the words are represented as 1-of-N encoded feature vectors. In this paper we investigate an alternative encoding of the word history, known as bag-of-words (BOW) representation of a word sequence, and use it as an additional input feature to the NNLM. Both the feedforward neural network (FFNN) and the long short-term memory recurrent neural network (LSTM-RNN) language models (LMs) with additional BOW input are evaluated on an English large vocabulary automatic speech recognition (ASR) task. We show that the BOW features significantly improve both the perplexity (PP) and the word error rate (WER) of a standard FFNN LM. In contrast, the LSTM-RNN LM does not benefit from such an explicit long context feature. Therefore the performance gap between feedforward and recurrent architectures for language modeling is reduced. In addition, we revisit the cache based LM, a seeming analog of the BOW for the count based LM, which was unsuccessful for ASR in the past. Although the cache is able to improve the perplexity, we only observe a very small reduction in WER.

Index Terms: language modeling, speech recognition, bag-of-words, feedforward neural networks, recurrent neural networks, long short-term memory, cache language model

1. Introduction

The LM estimates the probability $p(w_1^N)$ of a sequence of words w_1^N. Such probability appears in the fundamental Bayes' decision rules of many language and speech processing tasks, including handwriting recognition, machine translation and ASR. Its incorporation is crucial for these systems to achieve state-of-the-art results. Since this probability can be factorized by the chain rule as follows,

$$p(w_1^N) = p(w_1) \prod_{n=2}^{N} p(w_n|w_1^{n-1}) \quad (1)$$

the task of language modeling becomes to estimate these conditional probabilities $p(w_n|w_1^{n-1})$. The conventional approach estimates these probabilities based on the relative frequencies of n-gram counts and a smoothing technique is applied to distribute probability to unseen events [1, 2]. More recently, estimation of these quantities by neural networks (NNs) has been shown to produce state-of-the-art performance in many tasks. Previously, two types of NN architectures have been investigated for the LM: feedforward and recurrent NNs (RNNs). Early investigations were interested in RNNs [3], but the experiments were limited to the sequence prediction of stochastic regular grammars and they were not used in a real ASR task. Within a real LM framework, the investigations on the FFNN LM have first shown improvements in PP and WER over the count LM [4, 5, 6], the LM based on RNN was later reported to be more successful [7, 8] and the LM based on the improved architecture of RNN, LSTM-RNN LM [9] has been shown to achieve today's state of the art results in LM when interpolated to the count model. A number of works have compared these two types of architectures and most of them have shown that the RNN LM outperforms FFNN LM [8, 10, 11, 12]. Some exceptions are [13, 14]. However, in most of these works, the context length of FFNN LM is very limited. At most 10-gram [13, 12], often 4 or 5-gram FFNN LMs have been used by analogy to the count model, and have been compared to RNN LMs which are theoretically provided with infinitely long context. A possible reason for the previous absence of attempts to further increase the context size of FFNNs may be the inconvenience of the 1-of-N word representation for long word sequences.

In fact, in order to design a LM within a NN framework, words should be encoded as vectors. Most previous works on NNLM have represented words in the form of 1-of-N coding, also known as one-hot representation of words, which gives an index to each word in the vocabulary and a word is encoded as a vocabulary-sized vector with all coefficients set to 0, except the one at the index of the word to be encoded which is set to 1. A word sequence is therefore encoded as a sequence of 1-of-N vectors. In this paper, we investigate an alternative representation of a word sequence, known as bag-of-words, which is the sum of the 1-of-N vectors of the word sequence. While this representation discards the information about the word order, it provides a compact representation of long word sequences. It has been successfully used in text classification [15] and in computer vision [16]. We investigate the effect of an additional BOW input feature on both FFNN and LSTM-RNN LM, with respect to the PP and WER for a large vocabulary continuous speech recognition (LVCSR) task in English.

In addition, we revisit the cache based LM [17] for ASR in the last section. A number of works have been done to incorporate the long range dependencies to the n-gram count models. The techniques such as triggers [18], skip n-grams [19] and the cache based LM [17] can be viewed as analogs of BOW for count LMs. The cache LM is well known to give large improvements in PP over the n-gram count model by interpolation, though it does not lead to WER improvements: we attempt to bring some clarifications by removing pruning related search errors and by reporting the PP of the cache LM which was exactly used to obtain a specific WER.

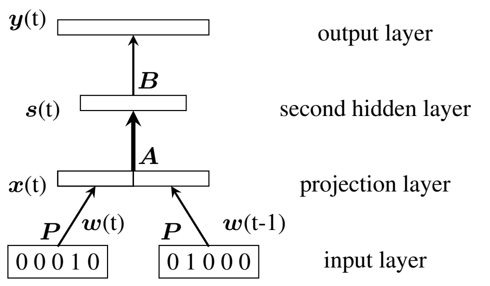

Figure 1: trigram Feedforward NNLM

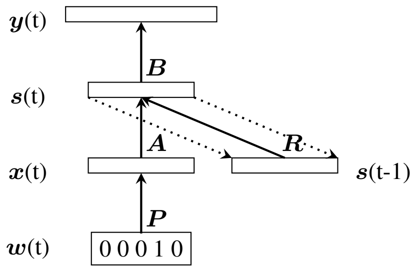

Figure 2: Standard Recurrent NNLM (with projection layer)

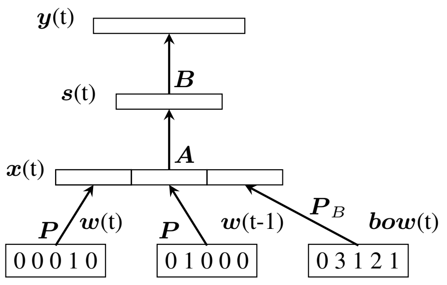

Figure 3: trigram Feedforward NNLM with an additional BOW input

2. Word History Information in NNLMs: Model Architecture Review

2.1. Feedforward Neural Network LM

In a standard FFNN LM, the Markov property is assumed as for the n-gram count LM:

$$p(w_i|w_1^{i-1}) = p(w_i|w_{i-n+1}^{i-1}) \tag{2}$$

FFNN LM takes a fixed number of predecessor words in form of 1-of-N vectors as input. The architecture of a trigram FFNN LM is illustrated in Figure 1. In practice, the output layer of NNLMs is factorized with word classes [20] to reduce the computation: we omit this in the illustration to avoid heavy notations. The forward pass of a trigram FFNN is as follows: given the 1-of-N inputs $\boldsymbol{w}$(t) and $\boldsymbol{w}$(t-1) of two predecessor words w_t and w_{t-1}, it computes

$$\begin{aligned} \boldsymbol{x}(t) &= [\boldsymbol{Pw}(t), \boldsymbol{Pw}(t-1)] & (3)\\ \boldsymbol{s}(t) &= f(\boldsymbol{Ax}(t)) & (4)\\ \boldsymbol{y}(t) &= [p(w|w_t, w_{t-1})]_{w\in V} = g(\boldsymbol{Bs}(t)) & (5) \end{aligned}$$

where V is the vocabulary set, $\boldsymbol{P}$, $\boldsymbol{A}$, $\boldsymbol{B}$ respectively are weight matrices of projection, hidden and output layers. f is an element-wise sigmoid activation and g is the softmax function. We omit the bias terms for the shortcut. The only context information n-gram FFNN LM is provided with is the (n-1) predecessor words and to extend FFNN to larger n-gram context, it requires to increase the size of $\boldsymbol{x}(t)$.

2.2. Recurrent Neural Network LM

As opposed to the FFNN LMs, RNN LMs only take one 1-of-N input corresponding to the previous word. Instead of taking a fixed length context, RNN stores the value of the hidden layer from the previous position $\boldsymbol{s}$(t-1) as illustrated in Figure 2. The corresponding equations are obtained by replacing the Eqs. (3) and (4) in those of FFNN with:

$$\begin{aligned} \boldsymbol{x}(t) &= [\boldsymbol{Pw}(t)] & (6)\\ \boldsymbol{s}(t) &= f(\boldsymbol{Ax}(t) + \boldsymbol{Rs}(t-1)) & (7) \end{aligned}$$

where $\boldsymbol{R}$ is a weight matrix of the recurrent connections. This structure provides the RNN with a theoretically infinite context. LSTM-RNNs are RNNs with improved architecture which also learn to control access to their internal memory depending on the context. We refer to [12] for details on this structure and the training of NNLMs.

3. Bag-of-Words Representation of Long Word History

The BOW of a word sequence is defined as the sum of 1-of-N representation of all words in the sequence. We investigate the BOW as an alternative vector representation of word sequences.

We use the BOW feature as an additional input to the NNLM as illustrated in Figure 3. The architecture of NNLM with additional features have been previously suggested by [21]. In this framework, the BOW vector of context size L, $\boldsymbol{bow}$(t) is defined at each time step as:

$$\begin{aligned} \boldsymbol{bow}(t) &= \sum_{i=0}^{L-1} \boldsymbol{w}(t-i) & (8)\\ &= \boldsymbol{bow}(t-1) + \boldsymbol{w}(t) - \boldsymbol{w}(t-L) & (9) \end{aligned}$$

The equations of FFNN with BOW input are obtained by replacing Eq (3) by:

$$\boldsymbol{x}(t) = [\boldsymbol{Pw}(t), \boldsymbol{Pw}(t-1), \boldsymbol{P}_B\boldsymbol{bow}(t)] \tag{10}$$

where $\boldsymbol{P_B}$ is the projection matrix for the BOW input. The same applies for RNN.

The BOW representation has one clear advantage over the 1-of-N representation, in particular for the FFNN LM: adding one more word in the context of the FFNN LM implies to increase the projection layer size by the feature size of one word. Even if the computational cost increases only linearly by this process, this is not appealing when considering a long context length, such as a 50-gram. The BOW representation can cover as many predecessor words as one wishes without further modifying the structure of the network.

Also, for the RNN LM, while theoretically the structure learns from the full word history, the question whether the presentation of an explicit long context to the input helps is unknown.

On the other hand, the drawback of the BOW is that it loses the order of the word sequence. A similar problem has been pointed out in [22] for the cache LM. They have introduced an exponential decay to express the distance of predecessor words from the current word. We adopted this approach for the BOW.

Corpus		# Words	VOC
Treebank	Train	930 K	10 K
	Dev	74 K	
	Test	82 K	
Quaero English	Train	50 M / 3.1 B	150 K
	Dev	40 K	
	Test	36 K	

Table 1: Corpora Statistics

Therefore, the BOW vector with decay, $\boldsymbol{bow}_{\text{decay}}(t)$ is defined as follows:

$$\boldsymbol{bow}_{\text{decay}}(t) = \sum_{i=0}^{L-1} \gamma^i \boldsymbol{w}(t-i) \qquad (11)$$

where γ is the decaying factor in [0,1]. The effect of this factor is discussed in the experimental section.

4. Experimental Results

4.1. PP Experiments on Small Corpus: Treebank Corpus

We computed the PP of various NNLMs with an additional BOW input on the Treebank corpus. The statistics of this corpus can be found in Table 1. We trained the baseline 5-gram modified Kneser-Ney LM (mod.KN5) with discounts optimized on the development set [23]. A BOW with context size of 50 words and with decay factor of 0.9 was used. All NNLMs in this paper were trained using rwthlm [24] in which we implemented the BOW features. We trained all NNLMs for this corpus with 300 hidden nodes. For the projection layer, we took a total size of 400 for the 4-gram FFNN with BOW (from which 100 was for the BOW), 399 for the standard 4-gram FFNN and 200 for all LSTM-RNN LMs.

The PP results are presented in Table 2. The results show that the standard 4-gram FFNN can be improved significantly by the addition of long context provided in the form of a BOW with decay. The use of decay seems to be important for this corpus. The addition of BOW improves the standard 4-gram FFNN by 8% on the test PP. The interpolation of 4-gram FFNN with BOW with decay, with the mod.KN5 improves the baseline by 27%, achieving the test PP of 102.3, which is only higher by 5% relative than that of LSTM, 97.3. We observed that the incorporation of BOW degrades the LSTM performance.

For a reference, we added the performance of the standard RNN [8]. It is noteworthy that the FFNN with BOW with decay gives a PP close to that of the standard RNN.

	DEV	EVAL
mod.KN5	146.8	140.7
4-gram FFNN	142.5	133.4
+ BOW	140.7	133.3
+ BOW decay	**128.2**	**122.5**
LSTM-RNN	114.9	110.6
+ BOW decay	119.0	115.3
mod.KN5 + 4-gram FFNN	119.8	114.5
+ BOW decay	**106.0**	**102.3**
mod.KN5 + LSTM-RNN	**100.6**	**97.3**
standard RNN [8]	-	123
standard RNN + KN5 [8]	-	106

Table 2: PP results on Treebank corpus for NNLMs with an additional BOW in the input. The PPs of standard RNNs are taken from [8]: the PPs on the development data were not provided.

4.2. English ASR Experiments

We conducted the experiments on the Quaero English LVCSR task[1]. The details of the corpus can be found in the Table 1. For the baseline, a 4-gram count LM with Kneser-Ney smoothing (KN4) [23], was trained. From the 3.1B full corpus, a 50M subset was selected to train the NNLMs, under the criterion of relevance indicated by the interpolation weights of KN4. For comparison, KN4 was trained on both the 50M and the 3.1B corpora. All NNLMs for this corpus have an output with class factorization. 1000 word classes were trained using exchange algorithm under the two-sided bigram criterion [25]. The ASR system used for this experiment is RWTH's best system for the Quaero Evaluation 2013. For the acoustic model, multilingual bottleneck features were trained in sequence discriminative manner with the minimum phone error criterion and used in a tandem approach: we refer to [12] for more details on the ASR system. In order to do experiments on the cache LM presented in the next section, we worked with the manual segmentations. We rescored the lattices with the push-forward algorithm [26] implemented in [24]. We took one beam-pruning threshold and expanded the lattices during the rescoring.

All NNLMs were trained with the projection layer size of 300 for each word. Each type of NNLMs were trained for the hidden layer sizes of 100 and 500. In order to fairly assess the effect of the BOW, for all NNLMs, we chose the BOW's projection layer size such that it has the same size as the feature of one word, while preserving the total size of the projection layer. A BOW with context size of 50 words and with decay of 0.9 was used. We trained bigram, 4-gram, 10-gram FFNNs and LSTM-RNNs with BOW input. Table 3 shows the PP results. It shows again that the BOW does not improve the LSTM-RNN while improving the FFNN. It is noteworthy that the 4-gram FFNN with BOW performed best among the FFNNs and that after interpolation with KN4, the PP of all FFNN LMs with BOW were close including the bigram case (the difference is larger in the case with 500 hidden layers).

The best overall eval PP achieved for a FFNN with BOW was 106.8 which is only 9% relative higher than that for the LSTM-RNN, 97.7. The WER results are presented in Table 4. It shows that with our best FFNN LM with BOW, an absolute improvement in WER of 0.9 from 11.7% to 10.8% over the count model was achieved, while the LSTM-RNN gave 10.2%.

In addition, we tested the rectifier linear unit [27] instead of the logistic function as the activation function of the baseline FFNN LM, but we did not observe a better PP result (we only used an $L2$-regularizer without momentum). For all results reported, the logistic activation is therefore used.

5. Revisiting the Cache based LM

In the previous section, the improvements in PP using BOW input were successfully transferred into the WER improvement. While many empirical results exist on the close correlation between PP and WER [28, 29, 12], the cache LM, which can be viewed as an analog to the BOW for the count LM, was unsuccessful in ASR in the past, despite a large PP improvement. We shortly revisit this model.

5.1. Previous Works

The cache based LM has been first proposed in [17] and was used for ASR in [30]. Given the cache size L, the unigram

[1]Quaero Programme: http://www.quaero.org

	DEV		EVAL	
	50M	3.1B Int.	50M	3.1B Int.
KN4	163.0	132.7	160.3	131.2
H=100				
Bigram FFNN	225.2	130.9	217.5	128.8
+BOW	168.3	118.9	165.9	117.8
4-gram FFNN	173.3	125.0	168.0	122.9
+BOW	153.4	**116.5**	151.9	**115.6**
10-gram FFNN	162.3	120.4	160.5	119.1
+BOW	154.0	116.6	152.6	115.6
LSTM-RNN	140.5	**111.7**	140.4	**111.4**
+BOW	146.6	113.0	144.5	112.1
H=500				
Bigram FFNN	212.3	130.5	205.0	128.1
+BOW	140.1	111.4	140.5	111.1
4-gram FFNN	149.6	120.9	145.0	118.7
+BOW	125.9	**107.0**	125.7	**106.8**
10-gram FFNN	138.2	113.8	136.6	112.7
+BOW	128.1	107.6	128.1	107.2
LSTM-RNN	110.5	**98.4**	110.1	**97.7**
+BOW	111.9	98.4	112.5	98.6

Table 3: PP of NNLMs with an additional BOW in input. 50M columns contain the PP of models trained on 50M corpus. 3.1B Int. stands for the linear interpolation with the KN4 trained on 3.1B corpus.

	DEV		EVAL	
	PP	WER	PP	WER
KN4	132.7	13.9	131.2	11.7
Bigram FFNN	130.5	13.8	128.1	11.7
+BOW	111.4	13.2	111.1	11.0
4-gram FFNN	120.9	13.3	118.7	11.2
+BOW	107.0	**13.0**	106.8	**10.8**
10-gram FFNN	113.8	13.1	112.7	11.1
+BOW	107.6	13.1	107.2	10.9
LSTM-RNN	98.4	**12.5**	97.7	**10.2**

Table 4: PP and WER (in %) results for NNLMs (H=500) with an additional BOW in input. PP are those of models interpolated with KN4 trained on 3.1 B.

cache model is defined as:

$$p(w|h) = C(w)/L \tag{12}$$

where $C(w)$ is the count of word w in the last L predecessor words. Then the model is used in linear interpolation with the baseline LM. The cache has been viewed as a simple way to extend the n-gram LM to include longer context and to locally adapt the LM model. Many have reported PP improvements of about 10% over baseline count models, but in most of these cases, the improvement in PP did not carry over to WER improvements (even degradations have been observed) [31, 32]. A detailed analysis of the model has been done by [33]. We revisited this model with a particular care to report the PP of the LMs which was exactly used to obtain the WER (which was not always clear in the literature), and we excluded the search problem related to the LM pruning.

5.2. Experimental Results

We tested unigram and bigram cache. The cache was updated at the beginning of a new speech segment and all previous sentences within the same recording were included in the cache. We report performance of two types of the cache LM: supervised cache LM, which uses the correct transcriptions of predecessor segments, and dynamic cache LM, which caches the ASR output of the previous segments. The basic setups are the same as the ASR experiment in the previous section. The same LM used to generate the lattices during the decoding was used as the baseline LM in rescoring. The interpolation weights used for dynamic cache were those optimized for supervised cases. In addition, we have conducted additional rescoring experiments with the lattice containing the correct transcriptions to exclude the search problem (we did Viterbi decoding for this): WER results remained the same. We also found that a careful tuning of the LM scale was important to avoid degradation in WER.

The results are summarized in Table 5. In all cases, we obtained PP improvements. With surprise, we found that on the evaluation data, the dynamic cache resulted in better PP than the supervised cache (this is not due to the favorable interpolation weights). We only get degradation in the case of supervised unigram but overall, the gain in WER is very small (only 1% relative improvement in WER for 10% in PP, in the best case). This is contrastive to the results we get with the BOW, which shows that the long dependencies a NN learns from a BOW is more than a simple cache as in a count LM which ignores the identity of all words in the cache but the current one.

		DEV		EVAL	
		PP	WER	PP	WER
	KN4	132.7	14.1	131.2	11.9
Supervised	Unigram	124.0	13.8	127.6	12.0
	Bigram	117.3	13.6	125.2	11.9
Dynamic	Unigram	125.6	14.1	124.4	11.8
	Bigram	120.4	13.9	120.8	11.8

Table 5: PP and WER of cache LMs, interpolated with KN4 on Quaero English. Viterbi decoding.

6. Conclusions

In this paper, we showed that the BOW representation offers a better alternative to the 1-of-N encoding of words to provide a FFNN with a long context. The additional BOW input feature consistently improved the standard FFNN LMs and reduced the performance gap between FFNN and RNN in language modeling: we showed that the previous comparisons between FFNNs and RNNs were not fair in the sense that FFNNs were not provided with long enough context. This result reminds us that RNNs are originally FFNNs but with deep structure in time. At the same time, BOW was shown to reduce the LSTM-RNN LM performance, which shows that LSTM-RNN might store the long context in a better way through its implicit deep structure. The experiment with the cache LM was again not conclusive in itself. Still, it showed that NNLMs with BOW input do not seem to be analogous to cache LMs but learn more complex long dependencies which is useful for ASR. As future work, we want to investigate how the BOW feature behaves when the deep structure is provided. Also, we did not investigate the trade-off between including a context longer than the one we considered in this paper, and the decay factor, which could further improve the model with BOW.

7. Acknowledgements

H. Ney was partially supported by a senior chair award from DIGITEO, a French research cluster in Île-de-France.

8. References

[1] S. F. Chen and J. Goodman, “An empirical study of smoothing techniques for language modeling,” *Computer Speech & Language*, vol. 13, no. 4, pp. 359–393, 1999.

[2] J. T. Goodman, “A bit of progress in language modeling,” *Computer Speech & Language*, vol. 15, no. 4, pp. 403–434, 2001.

[3] M. Castaño, E. Vidal, and F. Casacuberta, “Inference of stochastic regular languages through simple recurrent networks,” in *IEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives*, Colchester, England, Apr. 1993, pp. 16/1–16/6.

[4] Y. Bengio, R. Ducharme, and P. Vincent, “A neural probabilistic language model,” in *Advances in Neural Information Processing Systems (NIPS)*, vol. 13, Denver, CO, USA, 2000, pp. 932–938.

[5] H. Schwenk and J.-L. Gauvain, “Connectionist language modeling for large vocabulary continuous speech recognition,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Orlando, FL, USA, 2002, pp. 762–765.

[6] H. Schwenk, “Continuous space language models,” *Computer Speech & Language*, vol. 21, no. 3, pp. 492–518, 2007.

[7] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, “Recurrent neural network based language model.” in *Proc. Interspeech*, Makuhari, Japan, 2010, pp. 1045–1048.

[8] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 5528–5531.

[9] M. Sundermeyer, R. Schlüter, and H. Ney, “LSTM neural networks for language modeling.” in *Proc. Interspeech*, Portland, OR, USA, Sep. 2012, pp. 194–197.

[10] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, “Deep neural network language models,” in *Proc. of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Montréal, Canada, Jun. 2012, pp. 20–28.

[11] M. Sundermeyer, I. Oparin, J.-L. Gauvain, B. Freiberg, R. Schluter, and H. Ney, “Comparison of feedforward and recurrent neural network language models,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013, pp. 8430–8434.

[12] M. Sundermeyer, H. Ney, and R. Schlüter, “From feedforward to recurrent lstm neural networks for language modeling,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[13] L. Hai Son, A. Allauzen, and F. Yvon, “Measuring the influence of long range dependencies with neural network language models,” in *Proc. of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Montréal, Canada, Jun. 2012, pp. 1–10.

[14] A. Gandhe, F. Metze, and I. Lane, “Neural network language models for low resource languages,” in *Proc. Interspeech*, Singapore, Sep. 2014.

[15] A. McCallum, K. Nigam *et al.*, “A comparison of event models for naive bayes text classification,” in *AAAI-98 workshop on learning for text categorization*, vol. 752, Madison, WI, USA, 1998, pp. 41–48.

[16] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *Workshop on Statistical Learning in Computer Vision, European Conference on Computer Vision (ECCV)*, vol. 1, no. 1–22, Prague, Czech Republic, May 2004.

[17] R. Kuhn and R. De Mori, “A cache-based natural language model for speech recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 570–583, 1990.

[18] R. Rosenfeld, “A maximum entropy approach to adaptive statistical language modelling,” *Computer Speech & Language*, vol. 10, no. 3, pp. 187–228, 1996.

[19] D. Klakow *et al.*, “Log-linear interpolation of language models.” in *Proc. of International Conference on Spoken Language Processing (ICSLP)*, Sydney, Australia, 1998.

[20] F. Morin and Y. Bengio, “Hierarchical probabilistic neural network language model,” in *Proc. of the international workshop on artificial intelligence and statistics (AISTATS)*, vol. 5, Barbados, Jan. 2005, pp. 246–252.

[21] T. Mikolov and G. Zweig, “Context dependent recurrent neural network language model.” in *Proc. of Spoken Language Technologies (SLT)*, Miami, FL, USA, 2012, pp. 234–239.

[22] P. R. Clarkson and A. J. Robinson, “Language model adaptation using mixtures and an exponentially decaying cache,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, Munich, Germany, Apr. 1997, pp. 799–802.

[23] M. Sundermeyer, R. Schlüter, and H. Ney, “On the estimation of discount parameters for language model smoothing,” in *Proc. Interspeech*, Florence, Italy, Aug. 2011, pp. 1433–1436.

[24] ——, “rwthlm the RWTH Aachen University neural network language modeling toolkit,” in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 2093–2097.

[25] S. Martin, J. Liermann, and H. Ney, “Algorithms for bigram and trigram word clustering,” *Speech communication*, vol. 24, no. 1, pp. 19–37, 1998.

[26] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, “Lattice decoding and rescoring with long-span neural network language models,” in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 661–665.

[27] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proc. of the 27th Int. Conf. on Machine Learning*, Haifa, Israel, Jun. 2010, pp. 807–814.

[28] S. F. Chen, D. Beeferman, and R. Rosenfield, “Evaluation metrics for language models,” *in Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, pp. 275–280, 1998.

[29] D. Klakow and J. Peters, “Testing the correlation of word error rate and perplexity,” *Speech Communication*, vol. 38, no. 1, pp. 19–28, 2002.

[30] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss, “A dynamic language model for speech recognition.” in *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, Feb. 1991, pp. 293–295.

[31] S. C. Martin, J. Liermann, and H. Ney, “Adaptive topic-dependent language modelling using word-based varigrams,” in *Proc. Eurospeech*, Rhodes, Greece, Sep. 1997.

[32] S. Khudanpur and J. Wu, “Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling,” *Computer Speech & Language*, vol. 14, no. 4, pp. 355–372, 2000.

[33] P. Clarkson and T. Robinson, “The applicability of adaptive language modelling for the broadcast news task.” in *Proc. of International Conference on Spoken Language Processing (ICSLP)*, Sydney, Australia, 1998.