Empirical Evaluation and Combination of Advanced Language Modeling Techniques

Tomáš Mikolov[1], *Anoop Deoras*[2], *Stefan Kombrink*[1], *Lukáš Burget*[1], *Jan "Honza" Černocký*[1]

[1]Speech@FIT, Brno University of Technology, Czech Republic
[2] Department of Electrical and Computer Engineering, Johns Hopkins University, USA
{imikolov,kombrink,burget,cernocky}@fit.vutbr.cz, adeoras@jhu.edu

Abstract

We present results obtained with several advanced language modeling techniques, including class based model, cache model, maximum entropy model, structured language model, random forest language model and several types of neural network based language models. We show results obtained after combining all these models by using linear interpolation. We conclude that for both small and moderately sized tasks, we obtain new state of the art results with combination of models, that is significantly better than performance of any individual model. Obtained perplexity reductions against Good-Turing trigram baseline are over 50% and against modified Kneser-Ney smoothed 5-gram over 40%.

Index Terms: language modeling, neural networks, model combination, speech recognition

1. Introduction

In this paper, we will deal with the statistical approaches to language modeling, that are motivated by information theory. This will allow us to fairly compare different techniques. It is supposed that the model that is the best predictor of words given the context, is the closest model to the true model of language. Thus, the measure that we will aim to minimize is the cross entropy of the test data given the language model. The cross entropy is equal to log_2 perplexity (PPL). The per-word perplexity is defined as

$$PPL = \sqrt[K]{\prod_{i=1}^{K} \frac{1}{P(w_i|w_{1\ldots i-1})}} \quad (1)$$

It is important to note that perplexity does not depend just on the quality of the model, but also on the nature of training and test data. For difficult tasks, when small amounts of training data are available and large vocabulary is used (thus the model has to choose between many variants), the perplexity can reach values over 1000, while on easy tasks, it is common to observe values below 100.

Another difficulty that arises when using perplexity as a measure of progress is when improvements are reported as percentual reductions. It can be seen that constant relative reduction of entropy results in variable reduction of perplexity. For example, 20% reduction of perplexity is equal to 4.8% reduction of entropy if the original model had perplexity 100, but only 3.6% if the original model had perplexity 500.

Thus we claim that reporting perplexity reductions by using advanced techniques on different setups, which is a common practice today, can be often more confusing than helpful for evaluating contribution of new techniques. The other problem, which was originally pointed out by Goodman [1], is that new techniques are usually studied in isolation, and are very often compared just to a weak baseline model, such as a Good-Turing smoothed trigram. This does not allow us to estimate complementarity of the new proposed techniques to the already existing ones.

For combination of models, linear interpolation is the most popular technique due to its simplicity. The probability of a word w in the context h is computed as

$$P(w|h) = \sum_{i=1}^{N} \lambda_i P_i(w|h) \quad (2)$$

where N it the number of models. The weights λ_i of individual models are non-negative and sum to 1, i.e. $\sum_{i=1}^{N} \lambda_i = 1$. More complex, but also commonly used is the log-linear interpolation [15].

The rest of this paper is organized as follows. In section 2, we will present results obtained by various basic and advanced techniques on a standard task. Individual performance of models as well as combination of all techniques by using linear interpolation will be presented. In section 3, we will show results obtained with adaptive linear interpolation that uses variable interpolation weights. In section 4, we present automatic speech recognition results obtained with the most promising individual models on a larger task.

2. Penn Treebank corpus experiments

One of the most widely used data sets for evaluating performance of statistical language models is the Penn Treebank portion of the Wall Street Journal corpus (denoted further simply as Penn Treebank corpus). It has been used by numerous researchers, while using exactly the same settings (the same training, development and test data and the same vocabulary limited to 10K words). This is quite rare in the language modeling field, and allowed us to compare directly different techniques and their combinations, as many researchers were kind enough to provide us their results for the following comparison. It should be however noted that in our comparison, we are interested only in the accuracy of the model, and we ignore computational complexity.

This work was partly supported by Technology Agency of the Czech Republic grant No. TA01011328, Czech Ministry of Education project No. MSM0021630528, Grant Agency of Czech Republic projects Nos. GP102/09/P635 and 102/08/0707, and by BUT FIT grant No. FIT-11-S-2. Anoop Deoras was partly supported by the Human Language Technology, Center of Excellence.

Table 1: *Perplexity of individual models alone and after combination with baseline language models. Results are reported on the Penn Treebank corpus evaluation set.*

Model	Perplexity			Entropy reduction	
	individual	+KN5	+KN5+cache	over KN5	over KN5+cache
3-gram with Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram with Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram with Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram with Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram with Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
Maximum entropy model with 5-gram features	142.1	138.7	124.5	0.4%	0.2%
Random clusterings LM	170.1	126.3	115.6	2.3%	1.7%
Random forest LM	131.9	131.3	117.5	1.5%	1.4%
Structured LM	146.1	125.5	114.4	2.4%	1.9%
Within and across sentence boundary LM	116.6	110.0	108.7	5.0%	3.0%
Log-bilinear LM	144.5	115.2	105.8	4.1%	3.6%
Feedforward neural network LM [9]	140.2	116.7	106.6	3.8%	3.4%
Feedforward neural network LM [18]	141.8	114.8	105.2	4.2%	3.7%
Syntactical neural network LM	131.3	110.0	101.5	5.0%	4.4%
Recurrent neural network LM	124.7	105.7	97.5	5.8%	5.3%
Adaptive RNNLM	123.2	102.7	98.0	6.4%	5.1%
Combination of static RNNLMs	102.1	95.5	89.4	7.9%	7.0%
Combination of adaptive RNNLMs	101.0	92.9	90.0	8.5%	6.9%

The Penn Treebank corpus was divided as follows: sections 0-20 were used as training data (930k tokens), sections 21-22 as validation data (74k tokens) and sections 23-24 as test data (82k tokens). All words outside the 10K vocabulary were mapped to a special token (unknown word).

2.1. Performance of individual models

The performance of all individual models used in our further experiments is shown in Table 1. Results for the n-gram models were obtained by using SRILM toolkit [12] (we have used no count cutoffs). We can observe that the 5-gram model with modified Kneser-Ney smoothing (KN5) is performing the best among n-gram models, and we use it further as a baseline. As several techniques that are presented later exploit longer range dependencies, our second baseline is n-gram model with a unigram cache model.

The Maximum entropy LM was trained by using an extension to SRILM, with the default L1 and L2 regularization parameters [13]. Random clustering LM is a class based model described further in [10] (in our implementation, we used just simple classes for this model). We used 4-gram features for the Random forest language model [6]. We are aware of several implementations of structured language models evaluated on this dataset - in our experiments, we have used the one implemented by Filimonov[1] [7]. The Within and across sentence boundary LM [16] is a combination of several simpler models (including cache-like model, skip n-gram and a class based model).

Next, we present results with different neural network based language models, that have been consistently providing very good performance in wide variety of ASR tasks as reported in [3, 18, 4]. The log-bilinear LM [19] is an alternative to the standard NNLM. In our comparison, we have used the LIMSI implementation of feedforward NNLM [18] that follows closely the original Bengio's model [2]. In our previous work [9], we have presented alternative feedforward model that is based on two neural networks, each having one hidden layer. We have found that both feedforward NNLM architectures work almost the same. Emami has proposed a Syntactical NNLM [8] that aims to incorporate linguistic features into the neural network model. Emami's model was a state of the art model on this dataset, until we have recently shown that Recurrent neural network based language model (RNNLM) is performing better [5].

In [5], we have also proposed several extensions to further improve the RNNLM - by using more models with randomly initialized weights, which is loosely inspired by the Bayesian approach, we can obtain further significant gains. Linear interpolation with uniform weights is used for combination of different RNNLMs. By unsupervised adaptation of the model during the test phase, we can achieve topic adaptation and obtain cache-like information. Interestingly, combination of static and adaptive models leads to further gains, as we will see later on.

We have used over 20 different RNN models in the combination, which differ not only in the initialization of weights, but also in the size of the hidden layer. Originally in [5] we reported results with models that had 200 neurons in the hidden layer. In the extended results presented in this paper, we have used models with up to 400 neurons. For the adapted models, we have used fixed learning rate $\alpha = 0.1$ (where α is the standard learning rate parameter for the Backpropagation through time algorithm [20]).

2.2. Model combination

In Table 2, we report results on the Penn Treebank evaluation set after combining all models by using linear interpolation (see eq. 2). The weights of individual models were tuned to minimize perplexity on the evaluation set, as we did not have results with all models on the development set. However this is no serious flaw in our experiments: both the development and evaluation sets are highly coherent, thus the weights tuned on the dev set would be almost the same. Second, as we will show in the next section, we do not need any development data at all

[1] We are aware of slightly better results reported on this dataset with yet another structured LM - in [17], perplexity 118.4 is reported by using SuperARV language model combined with n-gram model.

Table 2: *Results on Penn Treebank corpus (evaluation set) after combining all models. The weight of each model is tuned to minimize perplexity of the final combination.*

Model	Weight	PPL
3-gram with Good-Turing smoothing (GT3)	0	165.2
5-gram with Kneser-Ney smoothing (KN5)	0	141.2
5-gram with Kneser-Ney smoothing + cache	0.0792	125.7
Maximum entropy model	0	142.1
Random clusterings LM	0	170.1
Random forest LM	0.1057	131.9
Structured LM	0.0196	146.1
Within and across sentence boundary LM	0.0838	116.6
Log-bilinear LM	0	144.5
Feedforward NNLM	0	140.2
Syntactical NNLM	0.0828	131.3
Combination of static RNNLMs	0.3231	102.1
Combination of adaptive RNNLMs	0.3058	101.0
ALL	1	83.5

to find optimal weights of individual models by using adaptive weights estimated on the recent history during processing of the test data.

The reported results show that RNN based language models dominate in the final combination, having together weight of almost two thirds. Other notable models that seem to contribute are 5-gram model with cache, Random forest LM, Within and across sentence boundary LM and Syntactical NNLM. In Table 3, we report results when we add models into the combination iteratively, by always adding the one that improves results the most. Somewhat surprisingly, we can see that the standard 5-gram model is the most complementary model to RNN models. Other models provide only small additional improvements.

3. Adaptive linear model combination

We have extended the usual linear combination of models that uses fixed weights to a case when weights of all individual models are variable, and are estimated during processing of the test data. The initial distribution of weights is uniform (every model has the same weight), and as the test data are being processed, we compute optimal weights based on the performance of models on the history of the last K words. In theory, K can be chosen so that all history is used. However we found that it is possible to use multiple model combinations, with different K values. Those that use small K aim to capture short context characteristics that can vary rapidly between individual sentences or paragraphs.

It should be noted that another important motivation for this approach is that combination of adaptive and static RNN models with fixed weights is suboptimal. When the first word in the test data is processed, both static and adaptive models are equal. As more data is processed, the adaptive model is supposed to learn new information, and thus its weight should increase. However if there is a sudden change of topic in the test data, the static model might perform better for several sentences.

Further improvement was motivated by the observation that adaptation of RNN models with the learning rate $\alpha = 0.1$ leads usually to the best individual results, but models in combination are more complementary if some are processed with larger learning rate. The results are summarized in Table 4.

Table 3: *Results on Penn Treebank corpus (evaluation set) when models are added sequentially into the combination. The most contributing models are added first.*

Model	PPL
Combination of adaptive RNNLMs	101.0
+KN5 (with cache)	90.0
+Combination of static RNNLMs	86.2
+Within and across sentence boundary LM	84.8
+Random forest LM	84.0

Table 4: *Results on Penn Treebank corpus (evaluation set) with different linear interpolation (LI) techniques.*

Model	PPL
Static LI of ALL	83.5
Static LI of ALL + Adapted RNNs with $\alpha = 0.5$	80.5
Adaptive LI of ALL + Adapted RNNs with $\alpha = 0.5$	79.4

4. Performance with increasing size of the training data

So far, we have reported results with models trained on small amount of data (930k tokens). It was observed by Goodman [1] that with increasing amount of the training data, improvements provided by many advanced techniques vanish, with a possible conclusion that it might be sufficient to train n-gram models on huge amounts of data.

Thus we extend our results by a study of behaviour of the most successful models so far - RNNLMs and n-gram models with cache - on a larger task. We have decided to extend our previous results reported in [4]. The training data consists of 37M tokens from English Gigaword, NYT section. We also report ASR results after rescoring 100-best lists from DARPA WSJ'92 and WSJ'93 data sets. In Table 5, we can observe that improvements both in entropy and word error rate actually do not vanish with increasing amount of the training data. This is quite optimistic result, as it gives further motivation for future work on language modeling.

For simplicity and due to computational requirements, we only combined two different RNN models[2] trained on all data. The final combination of KN5+cache, static and adaptive RNNs yields perplexity 108 on this task, while Good-Turing smoothed trigram achieves 246 and 5-gram with modified Kneser-Ney smoothing 212. We also report comparison of advanced language modeling techniques on this WSJ setup in Table 6.

[2]All RNN models for WSJ task were trained with factorization of the output layer using 400 classes to reduce computational complexity, as described in [5].

Table 5: *Comparison of results on the WSJ dev set obtained with models based on different amount of the training data.*

# words	PPL		WER		Improvement[%]	
	KN5	+RNN	KN5	+RNN	Entropy	WER
223K	415	333	-	-	3.7	-
675K	390	298	15.6	13.9	4.5	10.9
2233K	331	251	14.9	12.9	4.8	13.4
6.4M	283	200	13.6	11.7	6.1	14.0
37M	212	133	12.2	10.2	8.7	16.4

Table 6: *Comparison of advanced language modeling techniques on the WSJ task with all training data.*

Model	Dev WER[%]	Eval WER[%]
Baseline - KN5	12.2	17.2
Discriminative LM [14]	11.5	16.9
Joint LM [7]	-	16.7
Static RNN	10.5	14.9
Static RNN + KN	10.2	14.6
Adapted RNN	9.8	14.5
Adapted RNN + KN	9.8	14.5
All RNN	**9.7**	**14.4**

5. Conclusion and future work

On the Penn Treebank, we achieved a new state of the art result by using a combination of many advanced language modeling techniques, surpassing previous state of the art by a large margin - the obtained perplexity 79.4 is significantly better than 96 reported in our previous work [5]. Compared to the perplexity of Good-Turing smoothed trigram that is 165.2 on this setup, we have achieved 52% reduction of perplexity, and 14.3% reduction of entropy. Compared to the 5-gram with modified Kneser-Ney smoothing that has perplexity 141.2, we obtained 44% reduction of perplexity and 11.6% reduction of entropy.

On the WSJ task, we have shown that the possible improvements actually increase with more training data. Although we have used just two RNNLMs that were trained on all data, we observed similar gains as on the previous setup. Against Good-Turing smoothed trigram that has perplexity 246, our final result 108 is by more than 56% lower (entropy reduction 15.0%). The 5-gram with modified Kneser-Ney smoothing has on this task perplexity 212, thus our combined result is by 49% lower (entropy reduction 12.6%).

As far as we know, our work is the first attempt to combine many advanced language modeling techniques after the work done by Goodman [1], as usually combination of only two or three techniques is reported. We have found that many techniques are actually redundant and do not contribute significantly to the final combination - it seems that by using Recurrent neural network based language models and a standard n-gram model, we can obtain near-optimal results. However, this should not be interpreted as that further work on other techniques is useless. We are aware of several possibilities how to make better use of individual models - it was reported that log-linear interpolation of models [15] outperforms in some cases significantly the basic linear interpolation. While we have not seen any significant gains when we combined log-linearly individual RNNLMs, for combination of different techniques, this might be an interesting extension of our work in the future. However, it should be noted that log-linear interpolation is computationally very expensive.

As the final combination is dominated by the RNNLM, we believe that future work should focus on its further extension. We observed that combination of different RNNLMs works better than any individual RNNLM. Even if we combine models that are individually suboptimal, as was the case when we used large learning rate during adaptation, we observe further improvements. This points us towards investigating Bayesian neural networks, that consider all possible parameters and hyperparameters. We actually assume that combination of RNNLMs behaves as a crude approximation of a Bayesian neural network.

6. Acknowledgements

We would like to thank to all who have helped us to reach the described results, directly by sharing their results with us or indirectly by providing freely available toolkits[3] : Ahmad Emami (Random Clusterings and Structured NNLM), Hai Son Le and Ilya Oparin (Feedforward NNLM and Log-bilinear model), Yi Su (Random forest LM), Denis Filimonov (Structured LM), Puyang Xu (ME model), Dietrich Klakow (Within and Across Sentence Boundary LM), Tanel Alumäe (ME model), Andreas Stolcke (SRILM).

7. References

[1] Joshua T. Goodman (2001). A bit of progress in language modeling, extended version. Technical report MSR-TR-2001-72.

[2] Yoshua Bengio, Rejean Ducharme and Pascal Vincent. 2003. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155

[3] H. Schwenk. Continuous space language models. Computer Speech and Language, vol. 21, 2007.

[4] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, Sanjeev Khudanpur: Recurrent neural network based language model, In: Proc. INTERSPEECH 2010.

[5] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, Sanjeev Khudanpur: Extensions of recurrent neural network language model. In: Proc. ICASSP 2011.

[6] Peng Xu. Random forests and the data sparseness problem in language modeling, Ph.D. thesis, Johns Hopkins University, 2005.

[7] Denis Filimonov and Mary Harper. A joint language model with fine-grain syntactic tags. EMNLP 2009.

[8] Ahmad Emami, Frederick Jelinek. Exact training of a neural syntactic language model. In: Proc. ICASSP, 2004.

[9] Tomáš Mikolov, Jiří Kopecký, Lukáš Burget, Ondřej Glembek and Jan Černocký: Neural network based language models for highly inflective languages, In: Proc. ICASSP 2009.

[10] A. Emami, F. Jelinek. Random clusterings for language modeling. In: Proc. ICASSP 2005.

[11] S. F. Chen and J. T. Goodman. An empirical study of smoothing techniques for language modeling. In Proceedings of the 34th Annual Meeting of the ACL, 1996.

[12] Andreas Stolcke. SRILM - language modeling toolkit. http://www.speech.sri.com/projects/srilm/

[13] Tanel Alumäe and Mikko Kurimo. Efficient Estimation of Maximum Entropy Language Models with N-gram features: an SRILM extension , In: Proc. INTERSPEECH 2010.

[14] Puyang Xu, D. Karakos, S. Khudanpur. Self-Supervised Discriminative Training of Statistical Language Models. ASRU 2009.

[15] D. Klakow. Log-linear interpolation of language models. In: Proc. Internat. Conf. Speech Language Process., 1998.

[16] S. Momtazi, F. Faubel, D. Klakow. Within and Across Sentence Boundary Language Model. In: Proc. INTERSPEECH 2010.

[17] W. Wang and M. Harper. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In: Proc. EMNLP, 2002.

[18] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, F. Yvon. Structured Output Layer Neural Network Language Model. In: Proc. ICASSP 2011.

[19] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. Proceedings of the 24th international conference on Machine learning, 2007.

[20] D. E. Rumelhart, G. E. Hinton, R. J. Williams. 1986. Learning internal representations by back-propagating errors. Nature, 323, 533-536.

[3] Our own toolkit for training RNNLMs is available at `http://www.fit.vutbr.cz/~imikolov/rnnlm/`