5-2014

Optimization of Neural Network Language Models for keyword search

Ankur Gandhe
Carnegie Mellon University

Florian Metze
Carnegie Mellon University, fmetze@andrew.cmu.edu

Alex Waibel
Carnegie Mellon University, waibel@andrew.cmu.edu

Ian Lane
Carnegie Mellon University, lane@cmu.edu

Published In

Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 4888-4892.

OPTIMIZATION OF NEURAL NETWORK LANGUAGE MODELS FOR KEYWORD SEARCH

Ankur Gandhe, Florian Metze, Alex Waibel, Ian Lane

Carnegie Mellon University
Language Technology Institute
{ankurgan,fmetze,ahw,lane}@cs.cmu.edu

ABSTRACT

Recent works have shown Neural Network based Language Models (NNLMs) to be an effective modeling technique for Automatic Speech Recognition. Prior works have shown that these models obtain lower perplexity and word error rate (WER) compared to both standard n-gram language models (LMs) and more advanced language models including maximum entropy and random forest LMs. While these results are compelling, prior works were limited to evaluating NNLMs on perplexity and word error rate. Our initial results showed that while NNLMs improved speech recognition accuracy, the improvement in keyword search was negligible. In this paper we propose alternate optimizations of NNLMs for the task of keyword search. We evaluate the performance of the proposed methods for keyword search on the Vietnamese dataset provided in phase one of the BABEL[1] project and demonstrate that by penalizing low frequency words during NNLM training, keyword search metrics such as actual term weighted value (ATWV) can be improved by up to 9.3% compared to the standard training methods.

Index Terms— language modeling, neural networks, keyword search

1. INTRODUCTION

Statistical Language modeling is an important component of many natural language processing applications, including spelling correction, machine translation, automatic speech recognition and keyword search in speech. Modified Kneser-Ney smoothed models [1] have been shown to achieve the best performance[2] within n-gram models. In recent years, however, a variety of novel techniques for language modeling have been proposed, including maximum entropy language models [3], random forest language models [4], and neural network language models ([5],[6]). Of these, neural network language models have been shown to perform the best in automatic speech recognition tasks[7]. These were re-introduced recently in [8] to tackle the curse of dimensionality suffered by n-gram models and allow continuous representation of words. The models were subsequently modified for application to automatic speech recognition in [5],[10], and proved to perform better than n-gram back-off models. More recently, recurrent NNLMs were proposed in [6],[11], and shown to obtain better perplexity and higher speech recognition accuracy than feed-forward NNLMs.

Another advantage of neural network language models is that they are discriminatively trained and minimize an objective function. To our knowledge, no one has reported on using neural network language models for the task of keyword search and possibly optimizing the ff-NNLMs for improving ATWV. In this paper, we perform an empirical study of performance of feed-forward NNLMs, focusing on the problem of keyword search. The remainder of the paper is organized as follows. Section 2 describes the feed-forward architectures in detail. Section 3 , and 4 discuss the experimental setup and results. Concluding remarks are provided in section 5.

2. FEED-FORWARD NEURAL NETWORK LANGUAGE MODEL

In this paper, we compare the performance of feed-forward NNLMs for low-resource languages on WER and ATWV. The following sections describe in detail the architecture and training procedures used within this paper.

2.1. Architecture

In this paper we adopt the techniques introduced in [5] to train the feed-forward NNLMs (ff-NNLMs) and follow the notation used in their paper. Figure 1 shows the architecture of our model. The ff-NNLM, is an n-gram language model where the posterior probability distribution of the following word is computed for a given $n-1$ history using a neural network model. Each word in the model's vocabulary is represented as a sparse vector $S_{1\times N}$, where only the j^{th} column is 1 for the word w_j (1-of-N). The probability of current word is then modeled to depend on the n word history, ie :

$$p(w_j|history) = p(w_j|w_{j-1}, w_{j-2}, ...w_{j-n+1})$$

History is represented in the neural network by multiple sparse vectors at the input layer. Each sparse word vector is

[1]This effort uses the IARPA Babel Program language collection release IARPA-babel107-v0.7

mapped linearly to a continuous word projection of size P by a $N \times P$ weight matrix (F). The resulting layer formed by concatenating the continuous word vectors is called the *projection layer*. The size of the layer increases linearly with the n-gram history. The number of units in the projection matrix determines the number of features used to represent each word. The second layer is a the *hidden layer* that introduces hidden units and a hyperbolic tangent function as a non-linearity. The *output layer* has N units (equal to the vocabulary size of the model). At this layer a softmax function is applied to the activation of each unit to produce posterior probabilities. The probability for each word $p(w_j)$ for a specific history is then given by the activation value of the j^{th} output unit.

Let c_k represent the projections for the history, d_j be the activations of the hidden units and o_i be the outputs. Then, the $p(w_j|h_j)$ is given by:

$$d_j = tanh\left(\sum_l m_{jl} c_l + b_{1_j}\right) \tag{1}$$

$$o_i = \sum_j v_{ij} d_j + b_{2_i} \tag{2}$$

$$p_i = e^{o_i} / \sum_{k=1}^{N} e^{o_k} \tag{3}$$

where $m_{ji}\,, b_{1_j}$ correspond to matrix M and bias B_1 between *projection* and *hidden layer*, and $v_{ij}\,, b_{2_j}$ correspond to the matrix V and bias B_2 between *hidden* and *output layer*. The complexity to calculate a single probability with this ff-NNLM depends on the number of units in the different layers:

$$O \;=\; (n-1) \times P \times H + H + H \times N + N \tag{4}$$

2.2. Optimizing towards WER

It was shown in [12] that word error rate is roughly a linear function of perplexity. Hence, a good choice for the error function to be minimized by the ff-NNLM is the cross entropy of the data, given by:

$$E = \sum_{i=1}^{N} t_i log\, p_i + \beta\big(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2\big) \tag{5}$$

where t_i is the expected activation and p_i is the probability assigned to word i and β is the L2 regularization constant. All the weight matrices in ff-NNLMs are trained using standard back-propagation algorithm with an adaptive learning rate. It can be shown that the outputs of a neural network trained in this manner converge to the posterior probabilities. During training, the expected activation of output layer (t_i) is set to 1 for the next word in the training data and 0 for all others. Hence, the neural network directly minimizes the perplexity.

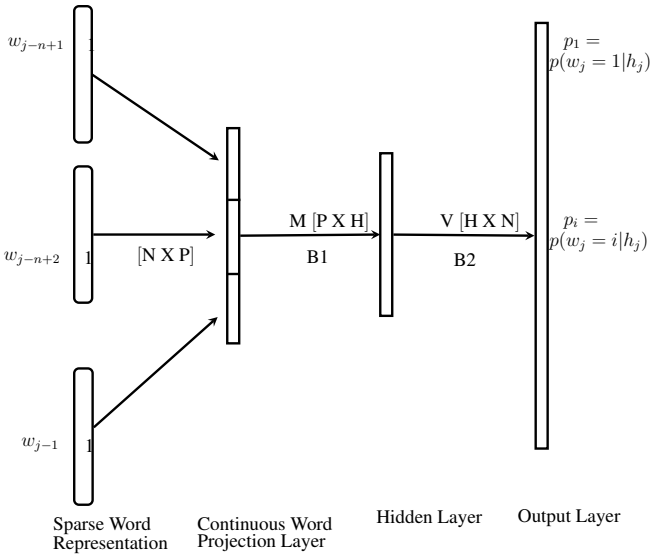

Fig. 1. Graphical Model of feed-forward neural network.

2.3. Optimizing towards ATWV

Actual Term Weighted Value(ATWV) is the primary evaluation metric for key word search in speech as defined in [13]. ATWV combines missed detections(P_{miss}) and false alarms(P_{FA}) as follows:

$$ATWV = \frac{1}{N_{terms}} \sum (1 - P_{Miss}(term) - \beta P_{FA}(term)) \tag{6}$$

$$P_{Miss}(term) = 1 - \frac{N_{correct}(term)}{N_{true}(term)} \tag{7}$$

$$P_{FA}(term) = \frac{N_{spurious}(term)}{T_{speech} - N_{true}(term)} \tag{8}$$

Even though all search terms are weighted equally in ATWV calculation, missing a rare term is more expensive than missing a frequent term because of the dependence of p_{Miss} on number of *true* terms. However, the language model metric, perplexity, is agnostic to the frequency of the words and gives equal weight to posterior probabilities of all words. As a result, improving perplexity can reduce p_{Miss} but often increases the p_{FA} for frequent key words leading to a low final ATWV. An intuitive modification to ff-NNLM training error is weighted cross entropy:

$$E_W = \sum_{i=1}^{N} w_i\; t_i log\, p_i + \beta\big(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2\big) \tag{9}$$

where w_i is the weight for the term t_i. When weights are higher for keywords than non-keywords, the estimated probabilities for keywords are better. But often the keywords are not known during model training. In such situations, we can use the knowledge that most keywords have a medium to rare

frequency in the training data and give weight according to word frequencies in the data. We use the following strategy to weight the words in the error function:

for *every training example* **do**
  if $freq_{t,i} < threshold$ **then**
    $w_i = penalty$;
  else
    $w_i = 1$;
  end
end

By weighing the errors in this manner, we boost the probability distributions of the words that fall below a threshold, while still minimizing a variation of perplexity. The meta-parameters $penalty$ and $threshold$ need to be tuned to the task. We report on the empirical values in the experiments section.

3. EXPERIMENTAL SETUP

3.1. Language models

Both standard n-gram and ff-NNLMs estimate the probability of next word based on a recent history of $n-1$ words and *tri-gram* models (history of two) is sufficient for speech recognition[2]. For our baseline experiments, we trained modified Kneser-Ney smoothed tri-gram language models (mKNLM) using the SRILM toolkit [14]. ff-NNLMs were trained using a projection layer of 100 and a hidden layer of 150 units. Speech recognition was performed using the Janus recognition toolkit [15] which can apply n-gram language models, NNLMs and combined mKNLM+NNLM models during both speech recognition decoding and lattice re-scoring. Key word search is performed on confusion networks, as described in [18]. The value of β was hand chosen and set to 999.9.

3.2. Language model training data

We evaluate the performance of the proposed methods on Vietnamese using the FullLP resources made available to participants within phase one the IARPA BABEL project[2]. The FullLP resource comprises of transcripts of 100 hours of training data, as shown in table 1. The average length of each utterance is 3 sec.

	Sentences	Tokens	Vocabulary
FullLP	78.6k	918k	6.2k

Table 1. Training data

[2]http://www.nist.gov/itl/iad/mig/upload/OpenKWS13-evalplan-v4.pdf

4. EXPERIMENT RESULTS

To evaluate the performance of the different language model techniques described earlier in the paper, we report results using following metrics on a 10 hour test set:

1. Perplexity: given by the $2^{\sum_x p(x)}$.
2. Word error rate
3. ATWV

We performed a set of experiments to assess the efficacy of language models as a function of meta-parameters and on the order of ff-NNLM models

4.1. Effect of error penalty

Our modified error function depends on two parameters, the $threshold$ for penalizing errors and the $penalty$ itself. We first experiment with several values of the penalty to find the best parameters. Table 2 shows the effect of varying the penalty value at a particular value of threshold. We observe that while no penalty ($= 1$) gives the best perplexity and WER, a penalty of 5 is best for ATWV. Even at higher penalty, when the perplexity is considerable lower than the best model, there is a 1% absolute improvement in ATWV. These set of results establish that language models should not be optimized towards perplexity for the task of keyword search.

	PPL	WER	ATWV
mKN-3g	125.4	50.2	0.273
ff-NN+mKN (penalty=1)	**109**	**49.6**	0.269
ff-NN+mKN (penalty=2)	109.3	49.8	0.270
ff-NN+mKN (penalty=5)	112	49.9	**0.291**
ff-NN+mKN (penalty=8)	115.9	50.0	0.282

Table 2. Performance with different penalty values for ff-NNLM error at threshold frequency= 50

4.2. Effect of threshold

Thresholding the words based on frequency is an important part of the optimization algorithm. If the keywords are very rare, the threshold value should be low. However, if the keywords are more distributed along the word-frequency curve, then we might need a high frequency. We vary the threshold value of frequency to find the optimal value. Table 3 shows the effect of threshold on the performance of language model. We observe that a penalizing words with frequency upto 25 yields the best ATWV. Intuitively, this penalizes the low and medium-low words in the training data.

However, the threshold also depends heavily on the training data. For some languages,especially the ones with morphology, the words may not follow zipf's law and we can not fix

	PPL	WER	ATWV
mKN-3g	125.4	50.2	0.273
ff-NN+mKN (thresh =5)	109.1	49.7	0.271
ff-NN+mKN (thresh =10)	109.7	49.7	0.279
ff-NN+mKN (thresh =25)	110.7	49.8	**0.292**
ff-NN+mKN (thresh=50)	112	49.9	0.291

Table 3. Performance with different word frequency based thresholds for penalizing ff-NNLM error at penalty = 5

the frequency. Hence, a better way is thresholding based on number of training samples being penalized. For example, when we penalize $\approx 1\%$ of samples, the frequency is thresholded at 8. Table 4 shows the result of varying the threshold based on number of training examples. Indeed, searching the threshold this way gives us a better final ATWV of 0.295.

	PPL	WER	ATWV
mKN-3g	125.4	50.2	0.273
ff-NN+mKN (1%,thresh =8)	110	49.7	0.280
ff-NN+mKN (2%,thresh =18)	110.4	49.7	0.288
ff-NN+mKN (3%,thresh =30)	111.1	49.8	**0.295**
ff-NN+mKN (4%,thresh=50)	112	49.9	0.291

Table 4. Performance with different thresholds based on amount if samples penalized ff-NNLM error at penalty = 5

4.3. Lower Order ff-NNLM

The complexity of forward calculation of ff-NNLM depends on the number of words used in the history. Many applications require faster computations and in such a case, having a lower order ff-NNLM is desirable. Additionally, a bigram ff-NNLM with a low vocabulary size can be converted into ARPA format and used by most ASR systems. We perform experiments using bigram language models and see the effect of ATWV optimization. Table 5 shows the performance of ff-NNLM on various metrics. Even though higher order mKN-LM has better perplexity, the ATWV at $penalty = 5$ is better 1% than the baseline system.

	PPL	WER	ATWV
mKN-2g	132.3	50.8	0.214
ff-NN-2g+mKN-2g (penalty=1)	127.1	50.6	0.268
ff-NN-2g+mKN-2g (penalty=5)	130.1	50.6	**0.281**
mKN-3g	**125.4**	**50.2**	0.273

Table 5. Performance with bigram ff-NNLM at penalty = 5 and thresholded at frequency = 30

4.4. Analysis

To analyze the cause of improved ATWV, we compare the baseline system, ff-NNLM system with best perplexity and the two ff-NNLM models with best ATWV on several metrics. Table 6 lists the performance of the 3 systems on them.

	ATWV	P_{Miss1}	P_{FA}
mKN-3g	0.273	0.682	0.0217
ff-NN+mKN (ppl)	0.269	0.67	0.0231
ff-NN+mKN (atwv1)	0.292	0.64	0.0235
ff-NN+mKN (atwv2)	0.295	0.65	0.0232

Table 6. Performance of baseline model and best model on different metrics

On analyzing the improvements based on keyword frequency, we observed that indeed the most improvement was obtained in rare and medium-frequent (4-15) words. This confirms our hypothesis and motivation for choosing the optimization algorithm. The false alarm probability for both ff-NNLM systems is higher than the baseline system. However, while the ATWV-optimized ff-NNLM system reduces the P_{Miss1} sufficiently enough to improve the final ATWV, the same does not happen in the standard ff-NNLM. It might be possible to combine the reduction of P_{Miss1} by the first ATWV-optimized ff-NNLMs with the low P_{FA} of the second ff-NNLM, but this is left as a future work.

5. CONCLUSION AND FUTURE WORK

We showed that by training the neural network language models towards a function other than perplexity, we can optimize towards the keyword search metric, ATWV and get improvements as high as 9.3%. The improvement in ATWV is at a small cost of degradation in WER, but it is still better than the baseline system. In future, we plan to combine the different language models for producing lattices and re-scoring lattices to improve ATWV further. Additionally, we observed most improvement in medium frequent and rare words. Most keywords are typically bi-grams and tri-grams and we can also create similar optimization algorithms based on bi-gram and tri-gram frequencies in the training data. We will also verify the performance on other languages to validate our findings.

6. ACKNOWLEDGEMENTS

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

7. REFERENCES

[1] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling", *Proc. ICASSP* vol 1. pp. 181-184, 1995.

[2] S. F. Chen and J. T. Goodman, "An empirical study of smoothing techniques for language modeling", *Proc. Computer Speech and Language*, pp. 359-394, 1999.

[3] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling", *Proc. Computer Speech and Language*, pp. 187228, 1996.

[4] Peng Xu and F. Jelinek, "Random forest in language modeling", *Proc. EMNLP*, pp. 325332, 2004.

[5] H. Schwenk and J. L. Gauvain, "Neural Network Language Models for Conversational Speech Recognition", *Proc. ICSLP*, pp. 1215-1218, 2004.

[6] T. Mikolov, M. Karafiát, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model, *Proc.Interspeech*, pp. 1045-1048, 2010.

[7] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Cernocký, "Empirical Evaluation and Combination of Advanced Language Modeling Techniques" , *Proc. Interspeech*, pp. 605-608, 2011.

[8] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A Neural Probabilistic Language Model", *Proc. J. Machine Learning Research*, vol 3, pp. 1137-1155, 2003.

[9] R. Miikkulainen, and M. G. Dyer, "Natural Language Processing with Modular PDP Networks and Distributed Lexicon", *Proc. Cognitive Science*, vol. 15, No. 3, pp 343-399, 1991.

[10] H. Schwenk, "Continuous space language models", *Proc. Computer Speech and Language*, pp. 492-518, 2007.

[11] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur, "Extensions of recurrent neural network language model", *Proc. ICASSP*, pp. 5528-5531, 2011.

[12] S. F. Chen, D. Beeferman and R. Rosenfeld, "Evaluation Metrics For Language Models" *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pp. 275-280,1998

[13] NIST Spoken Term Detection (STD) 2006 Evaluation Plan, http://www.nist.gov/speech/tests/std/docs/-std06-evalplan-v10.pdf

[14] A. Stolcke, "SRILM - an extensible language modeling toolkit", *Proc. ICSLP*, vol 2, pp. 901-904, 2002.

[15] H. Soltau, F. Metze, C. Fgen, and A. Waibel, "A one-pass decoder based on polymorphic linguistic context assignment", *Proc. ASRU*, pp. 214-217, 2001.

[16] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocký, " RNNLM - Recurrent Neural Network Language Modeling Toolkit" , *Proc. ASRU Demo session*, 2011.

[17] J. Gehring, Y. Miao, F. Metze and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders", *Proc. ICASSP* pp. 3377 - 3381, 2013.

[18] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary Independent Spoken Term Detection", *Proc. SIGIR*, pp. 615-622, 2007