MULTIPLE PARALLEL HIDDEN LAYERS AND OTHER IMPROVEMENTS TO RECURRENT NEURAL NETWORK LANGUAGE MODELING

Diamantino Caseiro, Andrej Ljolje

AT&T Labs Research, 180 Park Avenue, Florham Park, NJ 07932, USA

ABSTRACT

Recurrent neural network language modeling (RNNLM) have been shown to outperform most other advanced language modeling techniques, however, it suffers from high computational complexity. In this paper, we present techniques for building faster and more accurate RNNLMs. In particular, we show that Brown clustering of the vocabulary is much more effective than other techniques. We also present an algorithm for converting an ensemble of RNNLMs into a single model that can be further tuned or adapted. The resulting models have significantly lower perplexity than single models with the same number of parameters. An error rate reduction of 5.9% was observed on a state of the art multi-pass voice-mail to text ASR system using RNNLMs trained with the proposed algorithm.

***Index Terms*—** Language Modeling, Automatic Speech Recognition, Recurrent Neural Network Language Model, Multiple Parallel Hidden Layers

1. INTRODUCTION

In the last few years, exponential language models such as MaxEnt [1, 2], neural network language models (NNLM) [3, 4], and recurrent neural network language models [5] have proven to be better than traditional n-gram language models, in terms of both perplexity and word accuracy for automatic speech recognition (ASR).

RNNLMs are particularly promising. In a recent study [6] comparing many advanced language modeling techniques, ensembles of RNNLMs were shown to outperform, on their own, all other techniques. Furthermore, they seem to be complementary to n-gram models. One of the main disadvantages of RNNLM is their high computational complexity, both for testing and training. For training, mini-batch techniques used for parallelizing other neural networks, are not as effective due to the recurrent connections and the back propagation through time (BPTT) training algorithm. In this paper, we propose techniques for building more accurate and faster RNNLM.

A RNNLM is a neural network with the architecture shown in figure 1. It's input consists of the previous word $W(t-1)$ represented as a vector using 1 of N encoding, and $S(t-1)$, the activation of the hidden layer in the previous time step. Its output $O(t)$ is a vector containing the probability $P(w_t|h)$ of each word w_t in the vocabulary given the history h, or prefix, of the sentence up to t. The values in layers $S(t)$, $Y(t)$ and $O(t)$ are computed as:

$$s_j(t) = f(\sum_i w_i(t-1)u_{ji} + \sum_i s_i(t-1)r_{ji}) \tag{1}$$

$$f(z) = \frac{1}{1+e^{-z}} \tag{2}$$

$$y_k(t) = \sum_j s_j(t)v_kj \tag{3}$$

$$o_m(t) = \frac{e^{y_m(t)}}{\sum_k e^{y_k(t)}} \tag{4}$$

The network is trained using backpropagation through time.

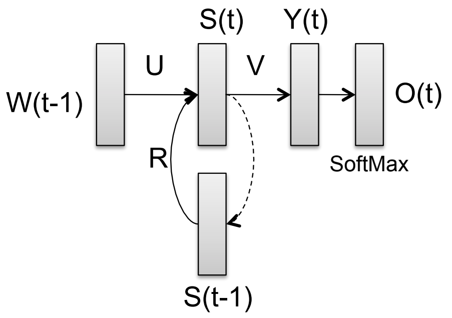

Fig. 1. Recurrent Neural Network Language Model Architecture.

In the next section we describe the corpora used to evaluate our work. In section 3 we show that class decomposition can be improved using Brown clustering. The following section presents a technique for building a large RNNLM from an ensemble of smaller RNNLMs. This technique enables faster and parallel training of large RNNLMs. Section 5 describes ASR experiments. It is followed by a description of related work, and the conclusions.

2. CORPORA

To enable a comparison of our results with those of other researchers, we conducted most experiments on the WSJ subset of the Penn Tree Bank corpus. Following established practice, we used sections 0-20 for model training (930k words), sections 21-22 for tuning and validation (74k words) and sections 23-24 for testing (82k words). The tokenization and the vocabulary (10k different words) is the same as used by other researchers, such as[7]. All results presented were obtained on the test set.

We used an internal AT&T voice mail to text system (VMTT) To experiment with the impact of RNNLM techniques on automatic speech recognition. The data used to build the system consists of 2 sets. The first one, `UM` contains transcribed voice mails from an internal AT&T system (2.4M tokens and a vocabulary of 28k words). The second one, `MW` is a larger set of poorly transcribed voice mails from a different application and population of users. This collection contains 49M tokens and has a vocabulary of 126k words. 10% of the `UM` data was set aside to fine tune the models and meta parameters. For evaluation, we used an independent set of 179 `UM` voice mails. All transcriptions were anonymized.

h	classes	speed	Avg PPL	Min PPL
50	200 Freq	26938	156.34	155.19
100	200 Freq	12740	142.93	142.11
200	200 Freq	5178	136.12	135.32
300	200 Freq	2866	135.14	133.99
50	200 Brown	25599	145.33	142.62
100	200 Brown	12087	132.60	131.58
200	200 Brown	4966	127.09	126.23
300	200 Brown	2785	125.06	124.21
50	none	720	145.37	142.79
100	none	355	132.95	131.37
200	none	186	126.55	125.51
300	none	131	125.12	123.46

Table 1. Speed and perplexity on Penn corpus for networks with various hidden layer sizes h using different class decomposition methods.

h	classes	speed	Avg PPL	Min PPL
200	50 Freq	4904	137.35	135.58
200	100 Freq	5620	136.54	135.60
200	200 Freq	5178	136.12	135.32
200	400 Freq	3822	135.96	134.65
200	50 Brown	3402	129.21	128.26
200	100 Brown	4846	128.10	126.69
200	200 Brown	4966	127.09	126.23
200	400 Brown	3786	126.31	125.32
200	none	186	126.55	125.51

Table 2. Speed and perplexity on Penn corpus for various number of classes using different class decomposition methods.

3. OUTPUT LAYER DECOMPOSITION

The computational complexity of RNNLM is very high both in training and testing. Given a vocabulary of size v and h hidden units, the runtime complexity is dominated by $\mathrm{O}(h^2 + hv)$. The most important factor is hv and occurs because the computation of the denominator of $o(t)$, e.g the *partition function*, requires summing over the full vocabulary. A very effective technique to reduce this complexity by a factor of up to $\sqrt{v}$ consists of partitioning the vocabulary in word classes, and decomposing $P(w|h) \approx P(c_w|h)P(w|c_w, h)$. The sub-models $P(c_w|h)$ and $P(w|c_w, h)$ are also exponential models, however the partition functions of each sub-model is much cheaper to compute. This technique has been used, for example, by [1, 8, 7, 9, 2]. In the context of RNNLMs, most published results use a simple frequency binning word to class assignment [7]: words are ranked by their frequency, then, following the rank, they are assigned to each class so that the frequency of each class is close to uniform. This technique is attractive because it is linear in the size of the corpus, and it creates balanced classes, however, it does not take into account any measure of modeling accuracy.

We compare the speed and accuracy of this technique with Brown clustering [10]. This is a computationally much more expensive algorithm, and can result in unbalanced classes, which cause slower RNNLMs for the same number of parameters. Since it clusters words that share similar bigram contexts, we expect it to result in lower perplexity than frequency mining. Tables 1 and 2 show the speed and perplexity obtained with both methods on a variety of configurations. Running speed is presented as words/second computed on a 2.4Ghz Intel Core 2 linux desktop. We show both the minimum and the average perplexity of 9 randomly initialized models. From these results, it is clear that Brown clustering closely approximates the perplexity obtained not using class decomposition at a speed comparable to frequency binning.

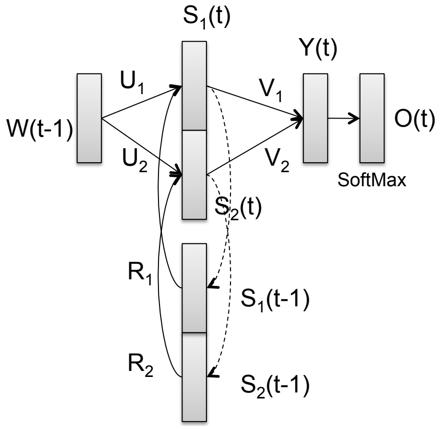

Fig. 2. MPHL combining 2 RNNLMs.

4. MULTIPLE PARALLEL HIDDEN LAYERS

It has been shown that the linear interpolation of many RNNLMs greatly reduces perplexity [7, 6]. These RNNLMs can be trained separately and should be diverse. Diversity can be achieved in a number of ways, for example: using different random initializations, different configurations, different permutations of the training data, and/or different training data subsets. While effective, an ensemble of models allows limited tuning options beyond interpolation weight optimization [11].

Both the linear and the geometric average (or log linear interpolation) model combination methods are well justified. They compute the model with the minimum average Kullback-Leibler divergence to all models in the ensemble [12]. The linear or the geometric averaged models are obtained depending on which model is taken as the prior or the posterior in the asymmetric Kullback-Leibler divergence.

Let's consider the weighted geometric average of two models:

$$P_{12}(w|h) = \frac{P_1(w|h)^\alpha P_2(w|h)^{1-\alpha}}{\sum_v P_1(v|h)^\alpha P_2(v|h)^{1-\alpha}} \quad (5)$$

Replacing P by the definition of exponential model (equation 4), and canceling the normalization terms:

$$P_{12}(w|h) = \frac{\left(e^{y_1(w,h)}\right)^\alpha \left(e^{y_2(w,h)}\right)^{1-\alpha}}{\sum_v \left(e^{y_1(v,h)}\right)^\alpha \left(e^{y_2(v,h)}\right)^{1-\alpha}} \quad (6)$$

Finally:

$$P_{12}(w|h) = \frac{e^{y_1(w,h)\alpha + y_2(w,h)(1-\alpha)}}{\sum_v e^{y_1(v,h)\alpha + y_2(v,h)(1-\alpha)}} \quad (7)$$

Thus the geometric average of exponential models is equivalent to the linear average of the weights to the final layer. This result applies to all exponential models including maximum entropy, deep-networks, NNLM, LSTM-LM and RNNLM, and gives us a principled way to merge models trained separately. Furthermore, it allows the combination and joint training of different techniques. The ME-RNNLM technique of [11] can be seen as the geometric interpolation and tuning of an RNNLM and a maximum entropy model.

The combination of multiple RNNLM in a single RNNLM, in particular, is illustrated in Figure 2 for 2 models. Matrices U_1 and U_2 are concatenated. Matrices V_1 and V_2 are linearly scaled by α or $(1-\alpha)$, and then concatenated. Finally R_1 and R_2 are combined in a (h_1+h_2)x(h_1+h_2) block diagonal matrix. The weights between hidden layer nodes from different sub-models are set to zero. This means that there are fewer independent parameters than in a single large model trained from scratch. An alternative implementation of the combination consists of keeping the hidden layers separate and averaging the last layer before computing the softmax. This was our chosen implementation because it allows for a trivial parallel implementation. We call the result of this model combination technique multiple parallel hidden layers (MPHL) language models. The algorithm consists of the following steps:

1. Train a diverse set of exponential language models
2. Rank models using the following greedy algorithm that iteratively removes the worst (most redundant) model from a set of models:
    (a) Start with the set S of all models
    (b) For each subset R of S such that $|R| = |S|-1$, evaluate the best possible perplexity of the linear interpolation of its component models on a development data set
    (c) Select the subset R^* with lowest perplexity
    (d) $S \leftarrow R^*$
    (e) if $|S| > 1$ go to (2.b)
3. Combine the best k models in a single model implementing geometric interpolation
4. Fine tune by continuing training of the combined model. Optionally retrain only the interpolation weights, or only V (both efficient convex optimization problems)

We use linear interpolation as a proxy for geometric interpolation in step 2.b, because linear interpolation weights can be optimized very efficiently. If desired, we optimize the geometric interpolation weights by averaged stochastic gradient descent (ASGD) [13] once the full network is assembled. Step 3 assumes that all models use the same word class decomposition. When that is not the case, we clear the weights V and retrain it. At first glance, step 4 seems computationally expensive, but training can be parallelized and ASGD converges in very few iterations.

Table 4 compares MPHL with geometric and linear interpolation of different models. Only the last layer of the MPHL network was retrained. MPHL achieves the best perplexity, while the perplexity of the interpolated models is comparable, with geometric interpolation being slightly better that linear interpolation.

Comparing MPHL models and RNNLMs with the same number of hidden units. We observe that MPHL models have significantly better perplexity than single RNNLMs. Furthermore, estimation is faster if we train the sub models in parallel. Comparing MPHL with difference configurations we see that combining fewer but larger models is preferable to combining many smaller ones. The reason is most likely the fact that the later have fewer non-zero recurrent connections.

5. ASR RESULTS

The VMTT speech recognition system used for the Speech Recognition Experiments is a state of the art multi-pass system [14]. Speech

h	MPHL	Linear	Log Lin
50	155.19	155.19	155.19
2x50	140.86	143.06	142.41
4x50	132.35	137.19	136.46
8x50	126.87	134.85	134.27
100	142.11	142.11	142.11
2x100	127.78	129.12	128.55
4x100	121.69	123.17	122.59
8x100	117.62	120.96	120.51
200	135.32	135.32	135.32
2x200	120.16	121.73	121.04
4x200	114.21	115.34	114.77
8x200	112.16	113.06	112.68

Table 3. Perplexity of MPHL, Linear and Log linear interpolation of multiple models on the Penn corpus. All models use 200 frequency classes.

h	MPHL	Linear	Log Lin
50	142.62	142.62	142.62
2x50	129.95	131.75	131.11
4x50	122.73	126.97	126.23
8x50	117.55	125.27	124.76
100	131.58	131.58	131.58
2x100	118.81	119.84	119.23
4x100	112.77	114.47	113.88
8x100	108.55	112.71	112.35
200	126.23	126.23	126.23
2x200	112.58	113.81	113.34
4x200	106.99	107.93	107.51
8x200	105.48	106.05	106.05

Table 4. Perplexity of MPHL, Linear and Log linear interpolation of multiple models on the Penn corpus. All models use 200 Brown classes.

recognition consists of a three-pass decoding approach utilizing MPE-trained baseline, VTLN and CMA trained acoustic models. The models used three-state left-to-right HMMs representing different phonemes for general English, spelled letters and digits. The models had 8.5k states and 26.5k HMMs, trained on just over 5000 hour of poorly transcribed voice mail recordings.The context dependency structure of the models was kept identical. The baseline model was used to generate hypotheses. The hypotheses were used to generate the VTLN warped features which were then used with the VTLN-trained model to generate the improved hypotheses. CMA was applied to the warped features, and the adapted features were recognized in the final pass with the model trained on the CMA-adapted training features. All the passes used the same language model, a linear interpolation of two Knesser-Ney 4-gram language models trained on the `UM` and `MW` data sets respectively. The language model contains 13.4M n-grams.

The recognition system was used to generate lists of 100-best and 1000-best hypotheses for each utterance. Then each hypothesis was rescored by replacing the baseline language model score with the score computed by linearly interpolating the baseline n-gram model with various RNNLM configurations. In particular, we trained 8 RNNLMs with 300 units in the hidden layers on the `UM` data, and 6 RNNLMs with 100 units in the hidden layer on the `MW` data. These networks were then combined using either linear interpolation or MPHL. In all cases the vocabulary was decomposed

System	Word ACC	% of oracle
Baseline 4-gram	78.06	0
Oracle 100 best	80.40	100
8x UM + 6x MW RNNLM linear	79.06	43
8x UM + 6x MW MPHL	79.07	43
Oracle 1000 best	81.28	100
8x UM + 6x MW RNNLM linear	79.23	36
8x UM + 6x MW MPHL	79.36	40

Table 5. Word accuracy of RNN and MPHL LMs in the Voice Mail Task.

in 200 Brown classes. The results are shown in table 5. The table shows the oracle accuracy for each list, we can see that it is only at most 3.2% better than the baseline system. The main reason for this is that these utterances are long, with 120 words on average, thus there is little variety in the list. Even with this constraint, the MPHLs are able to recover 1.3% absolute accuracy, reducing the error rate by 5.9%. MPHL model combination is better than linear interpolation. We also show the percentage of the oracle accuracy that we are able to recover from the n-best list. This percentage is defined as $100\text{x}(accuracy - baseline)/(oracle - baseline)$. The fact that we are able to recover around 40% of the accuracy in a 1000-best list indicates that both MPHL LMs and RNNLMs are constrained by the baseline hypotheses. An alternative rescoring technique, such as proposed by [15], may be able to achieve much better results.

6. RELATED WORK

The vocabulary clustering using Brown classes has been widely used in language modeling, for example, is has been used with n-gram models [10], and in both NNLMs [16] and MaxEnt models [1, 2]. Log Linear interpolation has also been used by many researchers [17]. Combining linear exponential models trained on different data sets by linearly averaging their parameters is a common technique for distributed training of MaxEnt models [18], however we are not aware of any previous work that uses it to build a single non-linear language model.

7. CONCLUSIONS

We have shown that the word clustering algorithm used to decompose the output layer of a RNNLM is critical, and that using algorithms such as Brown clustering that take into account the context of words leads to substantial reductions in perplexity relative to simpler word frequency binning approaches. We have also presented the relation between the ensemble combination technique of geometric averaging and the structure of exponential models. This enables us to combine separately trained models in a single model that can be further tuned. The combined model has substantially lower perplexity than a single model with the same number of parameter trained from scratch using BPTT. When evaluated on a state of the art ASR system, this technique proved to be superior to linear interpolation, and achieved a 5.9% error rate reduction relative to the n-gram baseline.

8. REFERENCES

[1] J. Goodman, “Classes for fast maximum entropy training transform,” in *Proceedings of ICASSP*, May 2001.

[2] Stanley F. Chen, “Shrinking exponential language models,” in *Proceedings of HLT-NAACL*, 2009.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin, “A neural probabilistic language model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003.

[4] H.-S. Le, I. Oparin, A.Allauzen, J.-L. Gauvain, and F.Yvon, “Structured output layer neural network language models for speech recognition,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 1, pp. 197–206, 2012.

[5] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, “Recurrent neural network based language model,” in *Proceedings of INTERSPEECH*, 2010, vol. 2010, pp. 1045–1048.

[6] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernocky, “Empirical evaluation and combination of advanced language modeling techniques,” in *Proceedings of Interspeech*, 2011, vol. 2011, pp. 605–608.

[7] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, “Extensions of recurrent neural network language model,” in *Proceedings of ICASSP*, 2011, pp. 5528–5531.

[8] Frederic Morin and Yoshua Bengio, “Hierarchical probabilistic neural network language model,” in *AISTATS05*, 2005, pp. 246–252.

[9] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, “Structured output layer neural network language model,” in *Proceedings of ICASSP*, 2011, pp. 5524–5527.

[10] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai, “Class-based n-gram models of natural language,” *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, Dec. 1992.

[11] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky, “Strategies for training large scale neural network language models,” in *Proceedings of ASRU*, 2011, pp. 196–201.

[12] D.J. Miller and Lian Yan, “Critic-driven ensemble classification,” *Trans. Sig. Proc.*, vol. 47, no. 10, pp. 2833–2844, Oct. 1999.

[13] Wei Xu, “Towards optimal one pass large scale learning with averaged stochastic gradient descent,” *CoRR*, vol. abs/1107.2490, 2011.

[14] A. Ljolje, V. Goffin, D. Caseiro, T. Mishra, and M. Gilbert, “Visual voice mail to text on the iphone/ipad,” in *INTERSPEECH*. 2011, pp. 3337–3338, ISCA.

[15] Anoop Deoras, Tom Mikolov, and Kenneth Church, “A fast re-scoring strategy to capture long-distance dependencies,” in *Proceedings of EMNLP*, 2011, pp. 1116–1127.

[16] Andrei Alexandrescu and Katrin Kirchhoff, “Factored neural language models,” in *Proceedings of NAACL*, Stroudsburg, PA, USA, 2006, NAACL-Short ’06, pp. 1–4.

[17] Dietrich Klakow, “Log-linear interpolation of language models,” in *Proceedings of ICSLP*, 1998.

[18] Gideon Mann, Ryan McDonald, Mehryar Mohri, Nathan Silberman, and Dan Walker, “Efficient large-scale distributed training of conditional maximum entropy models,” in *Advances in Neural Information Processing Systems 22*, pp. 1231–1239. 2009.