CONTEXT DEPENDENT RECURRENT NEURAL NETWORK LANGUAGE MODEL

Tomas Mikolov [*]

BRNO University of Technology
Czech Republic

Geoffrey Zweig

Microsoft Research
Redmond, WA USA

ABSTRACT

Recurrent neural network language models (RNNLMs) have recently demonstrated state-of-the-art performance across a variety of tasks. In this paper, we improve their performance by providing a contextual real-valued input vector in association with each word. This vector is used to convey contextual information about the sentence being modeled. By performing Latent Dirichlet Allocation using a block of preceding text, we achieve a topic-conditioned RNNLM. This approach has the key advantage of avoiding the data fragmentation associated with building multiple topic models on different data subsets. We report perplexity results on the Penn Treebank data, where we achieve a new state-of-the-art. We further apply the model to the Wall Street Journal speech recognition task, where we observe improvements in word-error-rate.

***Index Terms*—** Recurrent Neural Network, Language Modeling, Topic Models, Latent Dirichlet Allocation

1. INTRODUCTION

Recurrent neural network language models (RNNLMs) [1, 2] have recently been shown to produce state-of-the-art results in perplexity and word error rate across a variety of tasks [3, 4]. These networks differ from classical feed-forward neural network language models [5, 6, 7, 8, 9] in that they maintain a hidden-layer of neurons with recurrent connections to their own previous values. This recurrent property gives a RNNLM the potential to model long span dependencies. However, theoretical analysis [10] indicates that the gradient computation becomes increasingly ill-behaved the farther back in time an error signal must be propagated, and that therefore learning arbitrarily long-span phenomena is difficult.

In the past, a number of techniques have been used to bring long span and contextual information to bear in conventional N-gram language models. Perhaps the simplest of these is the cache language model [11] in which a language model score based on a model trained on the last K words is interpolated with that from a general model. Similar in spirit to the cache based models are the latent semantic analysis (LSA) based approaches of [12, 13]. These methods represent long-span history as a vector in latent semantic space, and base LSA-estimated word probability on the cosine similarity between a hypothesized word and the history. These similarity-based probabilities are then interpolated with N-gram probabilities. Topic-conditioned language models , e.g. [14, 15], most frequently work by partitioning the training data into subsets, with the goal of making subsets containing data on only one topic. Separate language models are then trained, and at runtime the most appropriate one (or combination) is chosen. Khudanpur and Wu [16] proposed the use of topic-features within a maximum-entropy framework, and in a voice-search application [17], long span context was used to enhance the maximum-entropy model of [18, 19] by creating features to indicate when a hypothesized word appeared in a user's history. Finally, in whole-sentence language models [20, 21], trigger features based on the existence of widely separated word pairs also provides long-span information.

In this paper, we study the use of long-span context in RNNLMs. One approach to increasing the effective context is to improve the learning algorithm to avoid the problem of vanishing gradients identified in [10]. This is exemplified by recent work on Hessian-free optimization [22]. Another is to modify the model itself, as in the Long Short-Term Memory neural networks [23], which use gating neurons to ”latch” onto the error signal for multiple timesteps. In contrast to these approaches, we have chosen to explicitly compute a context vector based on the sentence history, and provide it directly to the network as an additional input. This has the advantage of allowing us to bring sophisticated and pre-existing topic modeling techniques to bear with little overhead, specifically Latent Dirichlet Allocation (LDA) [24]. Moreover, it does this in a way that in other applications allows us to use context that is external to the text (e.g. a vector representing user-habits in voice search). This approach is similar in spirit to that of [25, 26], both of which bring side information to bear in a language model. Chu and Mangu [27] also recently used LDA to determine topics, but performed a hard partitioning of the data and built a set of disjoint models.

This paper makes several contributions. First, we suggest the use of context vectors to improve the performance of a RNNLM. Secondly, we demonstrate perplexity improvements over the previous state-of-the-art for the Penn Treebank. Thirdly, we develop an efficient method for computing context vectors when using a sliding window of context. Fi-

[*] Work performed while at Microsoft Research.

nally, we evaluate our models by rescoring N-best lists from a speech recognizer and observe improvements there as well.

The remainder of this paper is structured as follows. Section 2 describes the augmented RNN model we use. Section 3 describes our method of constructing context vectors based on Latent Dirichlet Allocation. Sections 4 and 5 present perplexity results on the Penn Treebank and word error rates on the Wall Street Journal speech recognition task. We provide some future directions and concluding remarks in Section 6.

2. MODEL STRUCTURE

The simple recurrent neural network language model [1] consists of an input layer, a hidden layer with recurrent connections that propagate time-delayed signals, and an output layer, plus the corresponding weight matrices. The input vector $\mathbf{w}(t)$ represents input word at time t encoded using 1-of-N coding (also called one-hot coding), and the output layer produces a probability distribution over words. The hidden layer maintains a representation of the sentence history. We extend this basic model with an additional *feature layer* $\mathbf{f}(t)$ that is connected to both the hidden and output layers, as shown in Figure 1. The feature layer represents an external input vector that should contain complementary information to the input word vector $\mathbf{w}(t)$. In the rest of this paper, we will be using features that represent topic information. Nevertheless, we note that the external features can be any information source such as part of speech tags, morphological information about the current word $\mathbf{w}(t)$, or speaker-specific information in the context of ASR.

There are several possible advantages to using topic information as additional input to the model, instead of building many separate topic-specific submodels: mainly, the training data will be less fragmented. Also, by providing the topic information directly at the input of the model, we elegantly avoid the long-standing problem of training recurrent neural networks to remember longer-term information (usually referred to as *the vanishing gradient problem*, and addressed in [10, 23]).

The input vector $\mathbf{w}(t)$ and the output vector $\mathbf{y}(t)$ have dimensionality of the vocabulary (later denoted as V). After the network is trained using stochastic gradient descent, the vector $\mathbf{y}(t)$ represents a probability distribution over words from the vocabulary given the previous word $\mathbf{w}(t)$, the context vector $\mathbf{s}(t{-}1)$ and the feature vector $\mathbf{f}(t)$.

The values in the hidden and output layers are computed as follows:

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t{-}1) + \mathbf{F}\mathbf{f}(t)\right) \qquad (1)$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t) + \mathbf{G}\mathbf{f}(t)\right), \qquad (2)$$

where

$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}. \qquad (3)$$

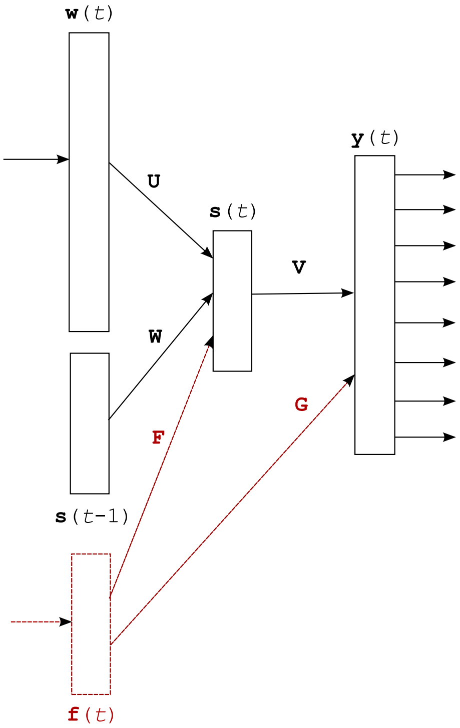

Fig. 1. Recurrent neural network based language model, with the additional feature layer $\mathbf{f}(t)$ and the corresponding weight matrices.

The training of neural network language models consists of finding the weight matrices $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{F}$ and $\mathbf{G}$ such that the likelihood of the training data is maximized. The reader is referred to [5, 28] for further detail.

3. LATENT DIRICHLET ALLOCATION FOR CONTEXT MODELING

We use Latent Dirichlet Allocation (LDA) [24] to achieve a compact vector-space representation of long span context. This procedure maps a bag-of-words representation of a document into a low-dimensional vector which is conventionally interpreted as a topic representation. For our purposes, a document will consist of a sentence or block of contiguous words. Each induced topic has associated with it a unigram distribution over words, and the collection of distributions is denoted β. LDA is a generative model of text, and the generation process of a document goes as follows:

1. Decide on the document length N by sampling from a Poisson distribution: $N \sim$ Poisson(ξ).
2. Decide on a multinomial distribution over topics for the document by sampling from a Dirichlet distribution parameterized by α: $\Theta \sim$ Dir(α).
3. For each of the N words to be generated, first decide on a topic to draw it from, and then on the word itself:

- Choose the topic $z_n \sim$ Multinomial(Θ).
- Choose a word w_n from the unigram distribution associated with the topic: $p(w_n|z_n, \beta)$.

A key parameter in LDA is α, which controls the shape of the prior distribution over topics for individual documents. As is common, we used a fixed α across topics. When α is less than 1, the prior distribution is peaked, with most topics receiving low probability in any given document. $\alpha = 1$ represents a uniform prior, and $\alpha > 1$ penalizes distributions that assign a high probability to any one topic in a specific document. Blei et al. [24] describe a method based on variational inference for learning the model parameters from a collection of documents, and our experiments use their implementation (`http://www.cs.princeton.edu/~blei/lda-c/`).

The result of LDA is a learned value for α, and the set of topic distributions β. An inference procedure can then be used to obtain the expected number of words accounted for by each topic in any given text, and thus the topic distribution of the text.

In our experiments, we used topic distributions computed from a fixed-length block of words preceding the current word. Thus, it is necessary to update the context vector after each word, which is an expensive process to do exactly. Therefore, as described in the next section, we have developed an efficient alternative method for computing context vectors based on the β matrix output by LDA.

3.1. Fast Approximate Topic Representations

Empirically, it has been observed [24] that the runtime for LDA inference is $O(kN^2)$ where N is the number of words, and k is the number of topics. Computing an LDA representation for each word given its sentence prefix would thus require $O(kN^3)$ time, which is undesirably slow. The same holds for computation over a sliding window of words. Therefore, we developed a more efficient method for computing context. In this computation, we make context vectors directly during the training of the RNN language model, using only the β matrix computed by the LDA. From the β matrix, we extract a continuous-space representation for each word by using the normalized column vectors. Since the topics are about equally represented in the training data, this results in a vector of entries $t_{w_i}^j$ representing $P(t_j|w_i)$.

We found that it is possible to compute a reasonable topic distribution for a block of words $\mathbf{w}$ by multiplying individual distributions over topics for each word from $\mathbf{w}$, and renormalizing the resulting distribution:

$$\mathbf{f}(t) = \frac{1}{Z} \prod_{i=0}^{K} \mathbf{t}_{w(t-i)}, \qquad (4)$$

where $\mathbf{t}_{w(t)}$ is the vector that represents the LDA topic distribution for word $w(t)$. For this approximation to work, it is important to smooth the β matrix by adding a small constant to avoid extremely small probabilities.

As we see in Section 4, the procedure can be further improved by weighting more recent words higher than those in the more distant past. To do this, we introduce features with an exponential decay, where we compute the feature vector as

$$\mathbf{f}(t) = \frac{1}{Z} \mathbf{f}(t-1)^{\gamma} \mathbf{t}_{w(t)}^{(1-\gamma)}, \qquad (5)$$

where γ controls the rate at which the topic vector changes - values close to 1 cause the feature vector to change slowly, while lower values will allow quick adaptation to topics.

While this procedure is not an approximation of the LDA inference procedure, we have found that it nevertheless does a good job representing contextual history, and admits an incremental update, reducing a factor of N^2 from the runtime. An empirical comparison to the use of LDA topic posteriors is found in Section 4.

4. PENN TREEBANK RESULTS

To maintain comparability with the previous research, e.g. [29, 3], we chose to perform experiments on the well-known Penn Treebank (PTB) [30] portion of the Wall Street Journal corpus. This choice also allows the fast evaluation of different models and parameter settings. We used the same standard preprocessing of the data as is common in previous research: all words outside the 10K vocabulary are mapped to the <unk> token; sections 0-20 are used as the training set (930K tokens), sections 21-22 as the validation set (74K tokens) and sections 23-24 as the test set (82K tokens). Extensive comparison of performance of advanced language modeling techniques on this setup is given in [3].

To obtain additional features at every word position, we trained a LDA model using documents consisting of 10 sentence long non-overlapping blocks of text from the PTB training data. We explored several configurations, and found that good results can be obtained with between 10 and 40 topics. Once the LDA model is trained, we can compute the probability distribution over topics for any new document. After some experimentation, we used a sliding window of the previous 50 words to represent the history. While this often goes beyond sentence boundaries, it makes sense because the PTB reproduces news stories, in which adjacent sentences are related to each other. The resulting probability distribution over topics is used directly as an additional feature input for the RNNLM.

Our initial experiments were performed using small RNN models with only 10 neurons. For reduction of computational complexity, we used a factorization of the output layer using 100 classes, as described in [2]. After tuning hyperparameters such as the optimal number of topics and the size of the sliding window, we ran the same experiments with RNN models having 100 neurons. The results are summarized in Table 1. As can be seen, the perplexity is reduced very

Table 1. Perplexities on the Penn Treebank Corpus for RNN models with LDA-based features, using 40 topics and a sliding window of 50 previous words.

Model	Dev PPL	Test PPL
Kneser-Ney 5-gram, no cutoffs	148.0	141.2
10 neurons, no features	239.2	225.0
10 neurons, exact LDA features	197.3	187.4
10 neurons, approx. LDA features	201.5	191.2
100 neurons, no features	150.1	142.1
100 neurons, exact LDA features	132.3	126.4
100 neurons, approx. LDA features	134.5	128.1

Table 2. Perplexities on the Penn Treebank Corpus with exponentially decaying features.

Model	Dev PPL	Test PPL
10 neurons, no features	239.2	225.0
10 neurons, $\gamma = 0$	220.7	195.7
10 neurons, $\gamma = 0.9$	201.4	190.5
10 neurons, $\gamma = 0.95$	198.5	187.5
10 neurons, $\gamma = 0.98$	199.8	190.1

significantly for small models, and the improvements hold up with larger models. Moreover, the approximate topic features of Section 3.1 work almost as well as the exact LDA features. Thus, in the subsequent experiments on larger data sets we focused on the approximate features. Table 2 shows that for values around $\gamma = 0.95$, the approximate features with exponential decay outperform those computed with an equally weighted window of the last 50 words (Table 1).

4.1. State-of-the-art Comparisons and Model Combination

In this section, we show that the improvements of a context-sensitive RNNLM hold up even in combination with Kneser-Ney 5-grams, cache LMs, and other models. Moreover, in combination with our best previous results we advance the state-of-the-art by 6% relative. Table 3 presents results for a RNN-LDA model with 300 neurons, and a set of previous models, both in isolation and when interpolated with a Kneser-Ney 5-gram model and a cache model. The RNN-LDA model outperforms the other models significantly, even after they are combined with the cache model. The description of the compared models is given in [5, 7, 29, 28].

Table 3. Perplexities on the Penn Treebank Corpus for various neural net models, interpolated with the baseline 5-gram and 5-gram+cache models. The RNN-LDA LM has 300 neurons, and uses 40-dimensional features computed on a 50-words history (sliding window).

Model	Individual	+KN5	+KN5+cache
KN5	141.2	-	-
KN5 + cache	125.7	-	-
Feedforward NNLM	140.2	116.7	106.6
Log-bilinear NNLM	144.5	115.2	105.8
Syntactical NNLM	131.3	110.0	101.5
Recurrent NNLM	124.7	105.7	97.5
RNN-LDA LM	113.7	98.3	92.0

Table 4. Perplexities on the Penn Treebank Corpus for model combination using linear interpolation.

Model	Test PPL
Kneser-Ney 5-gram, no count cutoffs	141.2
Model combination [28]	78.8
Combination of RNN-LDA models	80.1
RNN-LDA models + KN5 + cache	78.1
Combination of All	74.1

Next, we combine the new RNN-LDA model with the previous state-of-the-art model combination on this task. This is important to establish that the RNN LDA model provides truly new information. The previous model combination achieved a perplexity 78.8 by combining many different RNN LMs and other well-known language models such as a random forest LM [31], a structured language model [32], a class-based LM and other models [28]. For these combination experiments, we trained 8 RNN-LDA models with different configurations (up to 400 neurons and 40 dimensional LDA). The results are presented in Table 4. It can be seen that the final combination is significantly better than the best previous result. In addition, when we examine the interpolation weights, the vast majority of weight is assigned to the RNN-LDA models. The simple RNN models are the second most important group, and small contribution still comes from the 5-gram KN model with cache. Other techniques provide insignificant improvements.

4.2. Comparison with Latent Semantic Analysis

Our use of LDA has been motivated by the fact that it is one of the most widely used topic modeling approaches, and a fully generative probabilistic model. We have also performed a set of experiments with Latent Semantic Analysis [33], and found that it performs comparably. Because LSA word representations do not represent probability distributions, we perform the incremental update as

$$\mathbf{f}(t) = \gamma \mathbf{f}(t-1) + (1-\gamma)\mathbf{t}_{w(t)}, \tag{6}$$

The results are summarized in Table 5; the combination with “all” models now includes models using LSA features.

5. WALL STREET JOURNAL ASR RESULTS

In this section, we use a RNN-LDA model for N-best rescoring in a speech recognition setting. We used the medium-

Table 5. Treebank perplexities using LSA.

Model	Individual	+KN5	+KN5+cache	+All
RNN-LSA LM	110.3	96.4	90.3	**72.9**

sized Wall Street Journal automatic speech recognition task, with around 37M tokens in the LM training data. To handle the computational complexity associated with applying RNNs to larger data sets, a number of computational efficiencies have been developed [4], which are adopted here. The most useful techniques for complexity reduction are:

- factorization of the output layer using classes to avoid expensive normalization over the full vocabulary
- training the neural net jointly with a maximum entropy model with N-gram features, to avoid huge hidden layers (denoted as RNNME model)

For the following experiments, we used lattices generated with the Kaldi speech recognition toolkit [34]. To allow comparability with previous work, the 100-best lists used in this work are the same as those used in [28]. The triphone GMM acoustic models were trained on the SI-84 data further described in [35]. We tested with the 20k open-vocabulary task. Note that while we do not use advanced acoustic modeling techniques such as SGMMs [36] and speaker adaptation, the baseline system achieves comparable results as is common in the language modeling research [37] and is sufficient for comparison of advanced language modeling techniques.

The ASR system produced lattices using a pruned trigram model with Good-Turing smoothing, from which we generated the 100-best lists that are used in the rescoring. The baseline N-gram language models used for rescoring are a modified Kneser-Ney 5-gram (KN5) with singleton cutoffs, and a KN5 model without cutoffs.

Next, we trained a RNNME model with 10 hidden neurons and 1000M parameters for a concurrent hash-based ME model using 4-gram features. We used 200 classes in the output layer. Next, we trained RNNME models with the same configuration and with additional features that represent the topic information in the current sentence and with exponential decay $\gamma = 0.9$. We reset the feature vector at the beginning of each new sentence, thus the features represent only the topic of the current sentence. This places fewer constraints on the training and test phases (the order of sentences can be random). Results are presented in Table 6, where it can be seen that the topic information is very useful and leads to 0.4% - 0.6% WER reduction.

We trained additional RNNME models with 200 hidden neurons and with 40 LDA features, to see the potential of topic information in larger models. As can be seen in Table 6, we obtain similar reductions in perplexity as with the smaller models; nevertheless, it is harder to obtain improvements in word-error-rate over the large RNNME-200 model. We have explored several scenarios including combination with the KN5 model, always obtaining improvements of 0.1% - 0.2%. Still, the perplexity of the RNNME-200+LDA-40 model is by more than 5% lower than of the RNNME-200 model; thus in this task the improvements are sometimes small but always consistent across metrics and setups.

Table 6. Perplexities, and word error rates for WSJ 100-best list rescoring with RNNME and RNNME-LDA models.

Model	PPLX Dev	WER Dev	WER Test
KN5	121	12.5%	16.6%
KN5, no count cutoffs	113	12.0	16.6
RNNME-10	110	11.7	16.2
RNNME-10+LDA-5	105	11.1	15.9
RNNME-10+LDA-20	103	11.1	15.8
RNNME-200	89	9.9	14.7
RNNME-200+LDA-40	84	9.9	14.6
KN5+RNNME-200	87	9.9	14.6
KN5+RNNME-200+LDA-40	82	9.7	14.5

6. CONCLUSION

In this paper, we introduced the use of context dependent recurrent neural network language models. The main idea is to condition the hidden and output vectors on a continuous space representation of the previous words and sentences. Using a representation based on Latent Dirichlet Allocation, we are able to avoid the data fragmentation associated with the traditional process of building multiple topic-specific language models. We further develop a fast approximate context-updating technique which allows us to efficiently compute context vectors with a sliding window. The use of these models results in the lowest published perplexity on the Penn Treebank data, and in WER improvements for the Wall Street Journal task.

In the future, we are interested in applying our approach to leverage *external* information that is not present in the text. For example, in the machine translation setting, to use a source-language representation to condition the target-side language model, or in a voice-search setting to use a contextual vector representing a users interests.

7. REFERENCES

[1] Tomas Mikolov, Martin Karafiat, Jan Cernocky, and Sanjeev Khudanpur, “Recurrent neural network based language model,” in *Interspeech*, 2010.

[2] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, “Extensions of recurrent neural network based language model,” in *ICASSP*, 2011.

[3] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernocky, “Empirical evaluation and combination of advanced language modeling techniques,” in *Interspeech*, 2011.

[4] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky, "Strategies for Training Large Scale Neural Network Language Models," in *ASRU*, 2011.

[5] Y. Bengio, R. Ducharme, Vincent, P., and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Reseach*, vol. 3, no. 6, 2003.

[6] Holger Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492 – 518, 2007.

[7] Andriy Mnih and Geoffrey Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 641–648.

[8] Hai-Son Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured output layer neural network language model," in *ICASSP*, 2011.

[9] Le Hai Son, Alexandre Allauzen, and Francois Yvon, "Measuring the influence of long range dependencies with neural network language models," in *Proceedings of the Workshop on the Future of Language Modeling for HLT (NAACL/HLT 2012)*, 2012.

[10] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157 –166, 1994.

[11] R. Kuhn and R. de Mori, "A Cache-based Natural Language Model for Speech Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, 1990.

[12] J. Bellegarda, "Exploiting latent semantic information in statistical language modeling," *Proceedings of the IEEE*, vol. 88, no. 8, 2000.

[13] N. Coccaro and D. Jurafsky, "Towards better integration of semantic predictors in statistical language modeling," in *Proceedings, ICSLP*, 1998.

[14] R. Kneser and V. Steinbiss, "On the Dynamic Adaptation of Stochastic LM," in *ICASSP*, 1993.

[15] R.M. Iyer and M. Ostendorf, "Modeling long distance dependence in language: topic mixtures versus dynamic cache models," *Speech and Audio Processing, IEEE Transactions on*, vol. 7, no. 1, pp. 30 –39, 1999.

[16] Sanjeev Khudanpur and Jun Wu, "Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling," *Computer Speech and Language*, vol. 14, no. 4, pp. 355 – 372, 2000.

[17] G. Zweig and S. Chang, "Personalizing Model M for Voicesearch," in *ICSLP 2011*, 2011.

[18] S. Chen, "Performance prediction for exponential language models," in *NAACL-HLT*, 2009.

[19] S. Chen, "Shrinking exponential language models," in *NAACL-HLT*, 2009.

[20] R. Lau, R. Rosenfeld, and S. Roukos, "Trigger-based Language Models: A Maximum Entropy Approach," in *Proc. of ICASSP*, 1993.

[21] R. Rosenfeld, "A whole sentence maximum entropy language model," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, 1997.

[22] J. Martens and I. Sutskever, "Learning Recurrent Neural Networks with Hessian-Free Optimization," in *ICML*, 2011.

[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, , no. 9, 1997.

[24] David M. Blei, Andrew Y. Ng, and Michael I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[25] F. Zamora-Martinez, S. Espana-Boquera, M.J. Castro-Bleda, and R. De-Mori, "Cache neural network language models based on long-distance dependencies for a spoken dialog system," in *Proceeedings of ICASSP*, 2012.

[26] E. Arisoy, M. Saraclar, B. Roark, and I. Shafran, "Discriminative language modeling with linguistic and statistically derived features," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 2, pp. 540 –550, feb. 2012.

[27] S. Chu and L. Mangu, "Improving arabic broadcast transcription using automatic topic clustering," in *Proceeedings of ICASSP*, 2012.

[28] Tomas Mikolov, "Statistical language models based on neural networks," in *PhD Thesis, Brno University of Technology*, 2012.

[29] Ahmad Emami and Frederick Jelinek, "Exact training of a neural syntactic language model," in *ICASSP*, 2004.

[30] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini, "Building a large annotated corpus of english: the penn treebank," *Comput. Linguist.*, vol. 19, no. 2, pp. 313–330, 1993.

[31] Peng Xu and Frederick Jelinek, "Random forests and the data sparseness problem in language modeling," *Computer Speech and Language*, vol. 21, no. 1, pp. 105 – 152, 2007.

[32] Denis Filimonov and Mary Harper, "A joint language model with fine-grain syntactic tags," in *EMNLP*, 2009.

[33] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 96, 1990.

[34] D. Povey and A. Ghoshal et al., "The Kaldi Speech Recognition Toolkit," in *ASRU*, 2011.

[35] W. Reichl and W. Chou, "Robust Decision Tree State Tying for Continuous Speech Recognition," in *IEEE Trans. Speech and Audio Processing*, 2000.

[36] D. Povey and L. Burget et al., "The Subspace Gaussian Mixture Model - a Structured Model for Speech Recognition," in *Computer Speech and Language*, 2011.

[37] Puyang Xu, D. Karakos, and S. Khudanpur, "Self–Supervised Discriminative Training of Statistical Language Models," in *ASRU*, 2009.