CONTINUOUS SPACE LANGUAGE MODELING TECHNIQUES
Ruhi Sarikaya, Ahmad Emami, Mohamed Afify†, and Bhuvana Ramabhadran
IBM T.J. Watson Research Center Yorktown Heights, NY 10598 Cairo, Egypt
{ruhisisaar, ahmademami, mohamedafify}@us.ibm.com mohamed_afify2001@yahoo.com

ABSTRACT
This paper focuses on comparison of two continuous space language modeling techniques, namely Tied-Mixture Language Modeling (TMLM) and Neural Network Based Language Modeling (NNLM). Additionally, we report on using alternative feature representations for words and histories used in TMLM. Besides bigram co-occurrence based features we consider using NNLM based input features for training TMLMs. We also describe how we improve certain steps in building TMLMs. We demonstrate that TMLMs provide significant improvements of over 16% relative and 10% relative in Character Error Rate (CER) for Mandarin speech recognition, over the trigram and NNLM models, respectively in a speech to speech translation task.

Index Terms: Language Modeling, Continuous Space Modeling, Tied-Mixture Modeling, NNLM.

1. Introduction
Despite extensive research over the last three decades, it has been surprisingly difficult to improve upon n−gram language models. N−gram models are still the most widely used technique in all types of natural language processing, speech recognition and machine translation applications. There are several factors contributing to this reality. First, n−gram models are easy to build; all it requires is plain text. Second, the computational overhead to build an n−gram model is virtually negligible given the typical amount of data used in many applications. Last, n−gram models are fast to use during decoding as they do not require any computation other than a table look−up. There are several previous studies that showed improvements over n−gram models by imposing the syntactic and semantic structure onto the text [13, 15]. However, this requires some type of manual work to annotate the training data that the syntactic and semantic parsers are trained on. Moreover, the syntactic and semantic structure tend to be domain/task dependent. As the domain/task shifts from syntactic/semantic parser training data the accuracy of the imposed syntactic/semantic structure tends to degrade and so does the performance of those techniques.

Despite obvious advantages, n−gram models suffer from lack of generalization, lack of discrimination and lack of adaptation. N−gram models can only estimate probabilities for word sequences seen in the training data. They do not provide accurate probabilities for un−seen n−grams, but instead they back off to lower order n−gram models. Moreover, n−gram probabilities are not estimated discriminatively, rather they are based on maximum likelihood estimates. More importantly, n−gram models are inherently difficult to adapt to new domains/tasks, as there is no structure in the model. Each n−gram is a different entity and there is no dependency between n−grams. The adaptation issue is handled to some extent by collecting data in the target domain and building a small language model. The domain specific language model is interpolated with the initial, typically larger, language model.

Some of the weaknesses of the n−gram models have been addressed with limited success in several recent studies. For example, in [16, 17], n−gram parameters are updated in a discriminative fashion, which provides limited improvement in performance. In another set of studies [8, 3, 18] Neural Network based language model (NNLM) is proposed to address both discrimination and generalization. In NNLM, words which are inherently discrete entities, are represented by points in a continuous multi-dimensional feature space and the probability of a sequence of words is computed by means of a neural network. NNLM has achieved some modest level of improvement in performance and helped the generalization by estimating probabilities for any n−gram sequence. However, it is somewhat tedious and time consuming to train NNLM for large amounts of data. Moreover, NNLM did not provide a solution for adaptation. NNLM is interesting in that it trained continuous space representation of the words as part of language model training process.

Recently, we have proposed a new set of continuous space language modeling techniques such as Gaussian Mixture Language Models [9] (GMLMs) and Tied-Mixture Language Models [10] (TMLMs), which have the potential to provide solution to all of listed issues above (i.e. generalization, discrimination, adaptation). TMLM shares the concept of using continuous space representation for words or histories with NNLM, but it is an entirely different model. TMLM is a Hidden Markov Model (HMM) where the model parameters are trained using the training text data. A lingering question we have been receiving and we have been curious about is how TMLM compares to NNLM. In this paper, we compare TMLM and NNLM and also use NNLM based trained input features to train the TMLM. We also describe some of the improvements we made in perfecting TMLM.

The rest of the paper is organized as follows. Section 2 describes the NNLM. A brief description of TMLM is provided in Section 3. Section 4 introduces the speech recognition architecture. Experimental results are pre-
sented in Section 5, followed by the findings and future work in the last section.

2. Neural Network Based Language Modeling

The NNLM architecture is given in Figure 1. The feature vectors of the preceding words make up the input to the neural network, which then produces a probability distribution over a given vocabulary [8, 3, 18]. The neural network is fully connected and contains one hidden layer. The operations of the input is to stack vectors representing words:

$$\vec{f} = (f_1, \ldots, f_{d,m}) = (f(x_1), f(x_2), \ldots, f(x_m))$$  \hspace{1cm} (1)

where $f(x)$ is the $d$-dimensional ($d$ is set to 30 in our experiments) feature vector for word $x$. The hidden layer output $g_k$ is obtained with,

$$g_k = \tanh \sum_j (f_j L_{kj} + B_k^j), \ k = 1, \ldots, p$$  \hspace{1cm} (2)

The weights and biases of the hidden layer are denoted by $L_{kj}$ and $B_k^j$ respectively, and $p$ is the number of hidden units and is set 100. The output layer of the neural network for $k = 1, \ldots, V_o$ with vocabulary $V_o$ is computed as:

$$z_k = \tanh \sum_j (g_j S_{kj} + B_k^j), \ p_k = e^{z_k} \sum_j e^{z_j}$$  \hspace{1cm} (3)

where $S_{kj}$ and $B_k^j$ denote the weights and biases of the output layer, respectively. The softmax layer ensures that the outputs are valid probabilities and provides a suitable framework for learning a probability distribution. The $k^{th}$ output of the neural network, corresponding to the $k^{th}$ item $y_k$ of the output vocabulary, is the desired conditional probability: $p_k = P(y_k | x_1, \ldots, x_m)$. The neural network weights and biases, as well as the input feature vectors, are learned simultaneously using stochastic gradient descent training via back-propagation algorithm. The described architecture for NNLM allows increasing the context size to capture longer dependencies at the input without leading to potentially exponential increase as in the case of $n$-gram models. In fact the increase in the model size is linear.

3. Tied–Mixture Language Modeling

The novelty of TMLM lies not only in its continuous space representation of word histories but also in the modeling framework it introduces to the language modeling. At a high level, TMLM can be considered as an “acoustic modeling” problem [9, 10]. As such, it has the potential for discriminative training and adaptation using such techniques as Minimum Phone Error (MPE) [6] training and Maximum Likelihood Linear Regression (MMLR) [1]. It is essential to recall what we must have to build an acoustic model for any task in any language. We need three elements, i) acoustic data (waveforms), ii) transcriptions of the acoustic data, iii) baseforms (i.e. pronunciation dictionary).

In Fig. 2 we show the similarity between acoustic modeling and TMLM. We literally cast the language modeling problem as that of an acoustic modeling problem. We generate features from the words sequences and also use the word sequence as the corresponding transcriptions that these features belong to. Finally, we write the baseforms where each word has a pronunciation. Currently, we treat each word as a distinct “phone”. As such, none of the words share any common phones and hence any states except for the singletons, which share the same <UNK> state. However, one could cluster the words to share the same pronunciation, something which is not done here. Such a setting would be equivalent to building an acoustic model for a context–independent phone recognizer. Unlike a typical context–independent acoustic model, which has 30–to–60 phones to model, TMLM has as many phones as the vocabulary of the training text. This could be as large as several hundred thousand words depending on the amount of training text and language of interest. However, in principal we can build a more elaborate context–dependent word based TMLM just like we build a context–dependent phone recognizer model. However, this would require very large amounts of training text.

Even though the set of tools developed for acoustic modeling should in principal be used for training TMLM, there are some practical issues that have to be addressed simply because acoustic model training tools are build with the assumption of getting used only for acoustic modeling, with at most hundred or so phones. Handling phone sets of size 50K+ may require changes to the existing tools.

We adopted Tied–Mixture based HMM model structure [5, 14] for robust parameter estimation, where a set of Gaussian densities (1024 in our experiments) are shared across all states, which are representing the words in the vocabulary [10]. The HMM model parameters are estimated through an iterative procedure called the Baum-Welch, or forward-backward, algorithm [12]. The algorithm locally maximizes the likelihood function via an iterative procedure. This type of training is identical to training continuous density HMMs except the Gaussians are tied across all states.

In the TLM there is not any restriction in the way the feature vectors representing the history are generated. We have been considering generating the history feature vectors using bi-gram co-occurrence based features. In this study we also consider generating the feature vectors using the NNLM training, which trains the continuous space representation of words as part of the language modeling training process. Next, we briefly describe the bi-gram co-occurrence based history features.
3.1 Bigram Co–occurrence Based Features

The TMLM training process starts with the language model training corpus using each sentence at a time. From the sentences a bigram word co–occurrence matrix is constructed. The bigram co–occurrence matrix is decomposed using Singular Value Decomposition (SVD). Previously, we used to threshold the vocabulary for Singular Value decomposition. One of the improvements we made to the TMLM is not to threshold the vocabulary for SVD. When the vocabulary is thresholded all the words that are below the threshold are represented with a single <UNK> feature vector. This leads to over–generalization where the differences between singletons are lost as far as TMLM is concerned. Not surprisingly, we empirically found out that using all the vocabulary without thresholding is giving better results compared to thresholding the vocabulary.

The columns of the left–singular matrix obtained from SVD is used to map the bigram word histories into a lower dimensional continuous parameter space. The projected word history vectors are stacked together depending on the size of the n-gram. In our experiments here we set the rank of the SVD to 200. For example, for trigram modeling two corresponding bigram history vectors are stacked together as shown in Fig. 2 creating a feature vector of size 200 + 200 = 400. Even though, we have not done so, at this stage one could cluster the word histories for robust parameter estimation. Now, the feature vectors, their corresponding transcriptions and the baseforms are ready to perform the “acoustic model training” for TMLM.

In principal, TMLM is used during decoding just like an n-gram model. Given a hypothesized n-gram the TMLM first extracts the corresponding feature vectors. The feature vectors are used to estimate the likelihood of the word sequence where the last word in the sequence is the word to be predicted (it is also the state in TMLM) using the HMM parameters. Next, we describe the special HMM structure, namely Tied–Mixture models and how TMLM actually estimates the probability.

3.2 TMLM Probability Estimation

The Tied–Mixture based HMM built for TMLM estimates \( p(h | w) \), which is the probability of observing the history vectors \( h \) for a given word \( w \). Note that \( h \) can be obtained either via bigram co–occurrence or NNLM based input features. However, what we need is the posterior probability \( p(w | h) \) of observing \( w \) as the next word given the history, \( h \). This can be obtained using the Bayes rule:

\[
p(w|h) = \frac{p(h|w)p(w)}{p(h)} = \frac{p(h|w)p(w)}{\sum_{v=1}^{V} p(h|v)p(v)}
\]

where \( p(w|h) \) is the unigram probability of the word \( w \). In our TMLM implementation we substitute unigram probabilities with more accurate higher order n-gram probabilities. If this n-gram has an order that is equal to or greater than the one used in defining the continuous contexts \( h \), then the TMLM can be viewed as performing some type of smoothing of the original n-gram model:

\[
P_s(w | h) = \frac{P(w | h)p(h | w)}{\sum_{v=1}^{V} P(v | h)p(h | v)}
\]

where \( P_s(w | h) \) and \( P(w | h) \) are the smoothed and original n–grams. One other improvement we made to the TMLM estimation is to model all but singleton words as a separate state. All the singletons are modeled by the same state, even though they have different feature vectors representing them.

4. Speech Recognition System Architecture

The acoustic models are trained with about 1400 hours of GALE Mandarin dataset [19]. The GALE Mandarin dataset contains broadcast news speech data collected from various TV programs. The speaker independent acoustic model has 10K quinphone states modeled by 300K Gaussian densities. The speaker adaptive model uses a larger tree with 15K states and 500K Gaussians. The details of the acoustic model can be found in [19]. The decoding LM is from Chinese real–time speech–to–speech translation task covering mainly the travel domain. We use 882K/5.56M sentences/words from this domain to build a trigram language model. The trigram language model is built using modified Kneser–Ney smoothing [4]. While the n-gram vocabulary size is 50.3K words, the TMLM and NNLM are built using the top 40K words. The 40K words excludes some of the singletons (thresholding singletons would lead to 36.2K words) in the original 50.3K vocabulary.

5. Experimental Results

The language model rescoring experiments are performed using two test sets. TestA is from the travel domain TestB is from the medical domain, which is different than the language model training data. TestA consists of about 900 sentences spoken by 9 speakers with about 100 sentences/speaker and TestB consists of about 3900 utterances spoken by 26 speakers with 150 sentences/speaker. In order to evaluate the performance
of the continuous space language models, a lattice with a low oracle error rate was generated by a Viterbi decoder using the word trigram model (Word-3gr) model. From the lattice at most 50 (N=50) sentences are extracted for each utterance to form an N-best list. These utterances are rescored using TMLM and NNLM. The results are presented in Table 1. TMLM built using bigram co-occurrence based features are denoted as “TMLM–CO” and TMLM built using Neural Network based input features are denoted as “TMLM–NN”.

The language model performances are compared using Character Error Rate (CER), which is widely accepted to measure Chinese speech recognition performance. The N–best oracle error rates for TestA and TestB are 5.49% and 6.71%, respectively. The CER figures obtained for TestA are obtained using TestB as the development data to tune the language model weights for NNLM and TMLM and their interpolation with the Word–3gr model. The same process is repeated to obtain the CER for TestB, where this time TestA is used as the development data. Using NNLM by itself did not improve the CER on both test sets. Log–linear interpolation of NNLM with Word–3gr reduced the CER by 1.2% for TestB but has not improved the performance for TestA. TMLM–NN improved the CER by 0.6% and 0.9% over the baseline model. Interpolating TMLM–NN with Word–3gr further reduced the CER by an additional 0.9% and 1.1%. It is worth noting these numbers as they present a data point comparing the modeling power of NNLM and TMLM given the same set of features. Even though the features trained as part of the NNLM training, using them within the TMLM based modeling technique results in better performance. Using the TMLM–CO achieves a CER of 11.7% and 13.1% for TestA and TestB, respectively. Interpolating TMLM–CO with Word–3gr provides some additional small improvements. With TMLM–CO+Word–3gr improvements of 2.4% and 2.6% over Word–3gr and 2.4% and 1.4% over NNLM+Word–3gr are achieved for TestA and TestB, respectively. We observed that bigram co-occurrence based features provide better results than NNLM based input features for TMLM.

### 6. Conclusions and Future Work

We compared continuous space language models, namely Neural Network based Language Model (NNLM) and Tied–Mixture Language Model (TMLM). We considered using NNLM based continuous space representation of words as input to TMLM. We also described how we improved certain steps in building TMLMs. Then we evaluated these language models on a Chinese real–time translation services speech recognition task. TMLM with NNLM based input features (TMLM–NN) outperformed NNLM. TMLM with bigram co-occurrence based features (TMLM–CO) provided further improvement over TMLM–NN. These results are important and are motivating us to extend the experiments to various other tasks and languages as part of our future work.

### Table 1: Speech Recognition experiments for the comparison of continuous space language models.

<table>
<thead>
<tr>
<th>LM</th>
<th>TestA (Travel)</th>
<th>TestB (Medical)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CER (%)</td>
<td>CER (%)</td>
</tr>
<tr>
<td>N–best</td>
<td>5.49</td>
<td>6.71</td>
</tr>
<tr>
<td>Word-3gr</td>
<td>13.8</td>
<td>15.6</td>
</tr>
<tr>
<td>NNLM</td>
<td>15.7</td>
<td>16.6</td>
</tr>
<tr>
<td>TMLM–NN</td>
<td>12.3</td>
<td>14.4</td>
</tr>
<tr>
<td>TMLM–NN+Word-3gr</td>
<td>12.1</td>
<td>14.6</td>
</tr>
<tr>
<td>TMLM–CO</td>
<td>14.7</td>
<td>15.1</td>
</tr>
<tr>
<td>TMLM–CO+Word-3gr</td>
<td>14.4</td>
<td>14.0</td>
</tr>
</tbody>
</table>

### References


