GAUSSIAN MIXTURE LANGUAGE MODELS FOR SPEECH RECOGNITION

Mohamed Afify, Olivier Siohan, and Ruhi Sarikaya

IBM T.J. Watson Research Center
1101 Old Kitchawan Road, Yorktown Heights, NY, 10598

ABSTRACT

We propose a Gaussian mixture language model for speech recognition. Two potential benefits of using this model are smoothing unseen events, and ease of adaptation. It is shown how this model can be used alone or in conjunction with a a conventional N-gram model to calculate word probabilities. An interesting feature of the proposed technique is that many methods developed for acoustic models can be easily ported to GMLM. We developed two implementations of the proposed model for large vocabulary Arabic speech recognition with results comparable to conventional N-gram.

Index Terms: Language model, N-gram, Gaussian mixture model, continuous space.

1. INTRODUCTION

For long time the N-gram [1] has been the dominant technology for language modeling in speech recognition and other related applications. In spite of this success, the N-gram suffers from two major drawbacks that we refer to here as *generalizability* and *adaptability*. These two related issues will be discussed below.

The first problem, *generalizability*, arises because an N-gram which is not explicitly observed in the training corpus is not modeled. The best we could do is to back-off or smooth it with a lower order model. This sacrifices modeling accuracy. For example, the sentences "The dog runs in the bedroom", and "A cat walks in the room"[1] are quite similar in structure, but N-gram models are unconscious to this similarity and are incapable of using it. There are some methods proposed in the N-gram context to make use of these relationships. These techniques, generally speaking, employ word classes and/or parsing techniques to inject structural relationships, as in the previous example, in N-grams. One alternative view to the problem is in a continuous parameter space with a well defined notion of similarity. That is, some words or N-grams are "closer" to each other than other words or N-grams. The second aspect, *adaptability*, is also closely related to the first one. Usually an N-gram model has a huge number of parameters. Thus, it is very difficult to adapt it using a relatively small amount of data. This is in contrast to acoustic models, where a large model can be efficiently adapted using a few utterances. This is primarily achieved by exploiting the inherent structure in the model by techniques like maximum likelihood linear regression (MLLR) [3]. Again moving to a continuous parameter space can help alleviate this problem.

Based on the two problems that are briefly introduced above, we propose to build language models in a continuous parameter space using Gaussian mixtures. The principle of working in a continuous space for language models is not entirely new. Latent semantic analysis [4] employs a continuous representation for semantic clustering of words. In addition, it shows how to integrate this long history information with an N-gram for large vocabulary speech recognition. Also large scale neural network language models (NNLM) were initially proposed in [2]. They were then integrated in state of the art speech recognition system in [5]. The NNLM addresses the first problem raised above by moving to a continuous parameter space. In this space there is a notion of "distance" between different entities. This helps to generalize from seen to unseen events. However, it does not solve the second problem. This is because there is no quick method of adapting large neural networks from limited amount of data. In addition, the NNLM introduces scalability problems both in training and testing. Gaussian mixture models, on the other hand, do not suffer any of these limitations. We know how to build large scale GMMs and also to efficiently adaptat their parameters from acoustic modeling.

The paper is organized as follows. We formulate the Gaussian mixture language model (GMLM) in Section 2. We give a suitable mapping method from the discrete word space to a continuous space and show how to model the resulting continuous vectors. Sections 3 and 4 consider adaptation and parameter estimation for the proposed model. Experiment results for large vocabulary dialectal Arabic speech recognition are given in Section 5. Finally Section 6 summarizes our findings.

2. MODEL FORMULATION

Perhaps a simple way to think about language models in a continuous space is from a classification point of view. Each time the language model is presented with a history, either discrete as in the case of N-gram or continuous as we will discuss in this paper, it has to decide the most likely following word, or equivalently assign a probability to each word in a specified vocabulary. Thus, what we need to construct a continuous space LM is: a mapping from the discrete word space to a continuous representation, and a classifier which decides the next word given the mapped history in the resulting space. What we hope for is that in the new continuous space there is some form of distance or similarity between histories such that histories not observed in the data for some word are smoothed by similar observed histories. This addresses the *generalizability* issue discussed above for N-gram models. Interestingly, that is what exactly happens in the NNLM [2]. There is a linear layer which maps from the word space, an indicator vector with dimension equal to the vocabulary size, to a much smaller continuous parameter space. Concatenations of these continuous

[1] This example is taken from [2].

vectors, corresponding to the history, are then input to a classifier. This classifier is a multi-layer percepton (MLP), and has output nodes equal to the vocabulary size. Hence, for each input history this classifier chooses one of the output words, or alternatively assigns a probability value to each word in the vocabulary. Both the transformation and the classifier are learned together to maximize the likelihood of the data, which is a desirable property. However, NNLM can not be easily adapted, and suffers from scalability problems for typical amounts of data and vocabulary sizes in large vocabulary speech recognition.

Here, we take a different avenue. Assume we have a vocabulary of size V, each word $\{i \;\; 1 \leq i \leq V\}$ can be represented by an indicator vector w_i having one at the i^{th} position and zero in all other $V-1$ positions. This vector is mapped to a lower dimension (M) vector u_i, where M typically ranges from 50 to 150, using a matrix A of dimension $M \times V$. This can be written as

$$u_i = Aw_i \tag{1}$$

Now, each history h consists of a set of words of size $N-1$ for an N-gram[2]. Using the word mapping in Equation (1) the history h can be written as a concatenation of the appropriate mapped words as

$$v_j = u_{\mathcal{I}(h_{N-1})}.....u_{\mathcal{I}(h_1)} \tag{2}$$

where $\{j \;\; 1 \leq j \leq V^{N-1}\}$ is an index of the history, h_n indicates the n^{th} word in history h, and $\mathcal{I}()$ is the word identity of its argument. Vectors v_j are of dimension $M(N-1)$, this for typical 3-gram or 4-gram histories ranges from 150 to 600. Given the above mapping from history h to vector[3] v we can collect the training data histories for each word i and build a Gaussian mixture model. Alternatively one can cluster the words and construct a language model for each word cluster. It is known that modeling is more difficult in large parameter spaces, often referred to as curse of dimensionality. Thus, we can optionally propose another linear mapping B of the histories from the $M(N-1)$ dimension to a smaller dimension L around say 50, and construct the Gaussian mixture models in the new space given by

$$y_j = Bv_j \tag{3}$$

where $\{j \;\; 1 \leq j \leq V^{N-1}\}$ is an index of histroy, and y denotes the new feature space.

The previous paragraph gives a high level description of the proposed Gaussian mixture language modeling (GMLM) approach. We have not yet specified how to calculate the matrices A and B. We will do that in Section 4. Now, we will outline how to use the proposed model to calculate word probabilities.

In the following presentation $P()$ will be used to denote probabilities, while probability densities will be referred to as $p()$. Sometimes we will loosely speaking mention that $p()$ is an approximation to $P()$. In fact the Gaussian mixture model of each word w calculates the probability densities for an observed mapped history y given the word, which can be written as

$$p(y|w) = \sum_{k=1}^{K_w} c_{w,k}\mathcal{N}(y, \mu_{w,k}, \Sigma_{w,k}) \tag{4}$$

where K_w is the number of components in the Gaussian mixture of word w, and $\{c_{w,k}, \mu_{w,k}, \Sigma_{w,k} \;\; 1 \leq k \leq K_w\}$ are the mixture weights, means, and covariances for the k^{th} component respectively. Given the above mapping this can be viewed as an approximation to the probability $P(h|w)$, but what we are really interested in are the probabilities $P(w|h)$ or their approximation $P(w|y)$. This can be calculated using Bayes rule as follows

$$P(w|y) = \frac{P(w)p(y|w)}{p(y)} = \frac{P(w)p(y|w)}{\sum_{v=1}^{V} P(v)p(y|v)} \tag{5}$$

where $P(w)$ can be taken as the usual unigram probability estimate of word w.

Interesting generalizations of Equation (5) can be obtained as follows. First we may note that the unigram probabilities $P(w)$ can be made context dependent by taking them from an already exisitng N-gram. If this N-gram has an order that is equal to or greater than the one used in defining the continuous contexts y, then the Gaussian mixture model can be viewed as performing a kind of smoothing of the original N-gram. This can be written as

$$P_s(w|h) = \frac{P(w|h)p(y|w)}{\sum_{v=1}^{V} P(v|h)p(y|v)} \tag{6}$$

where $P_s(w|h)$, and $P(w|h)$ are the smoothed and original N-grams. Interestingly, Equation (6) has a close relationship to the history factorization suggested by Bellegarda in [4]. In [4] the word N-grams are modulated, to boost or decrease them, depending on some probabilities derived from the current topic or semantic content of the document. Here, the same modulation effect happens but using Gaussian mixture models constructed from the local N-gram statistics. To better understand this modulation effect, assume we have histories h_1 and h_2 for some word w, and that h_2 is not observed in the training data. In this case, h_2 will be assigned low probability by the conventional N-gram. If the two histories are close in the continuous space, then $p(y_2|w)$ will get a high value, because y_1 is used in constructing the mixture model, and hence will help boost the probability $P_s(w|h_2)$. Another interesting generalization of (6) can be obtained by taking the smoothing issue further and instead of building a Gaussian mixture for each word, we build one for each word cluster, where word clusters can be obtained using any appropriate method. This can be written as

$$P_s(w|h) = \frac{P(w|h)p(y|\mathcal{C}(w))}{\sum_{v=1}^{V} P(v|h)p(y|\mathcal{C}(v))} \tag{7}$$

where $\mathcal{C}(w)$ is the word class of w. In the above equations (5), (6), and (7) it may happen that the dynamic ranges of the N-gram and the Gaussian mixutre probabilities vary. In this case exponents can be used to overcome this difficulty in the same way as done in acoustic models. Also the calculation of the Gaussian mixture models, as in for example Equation (5), for all words or word classes may be computationally intensive. The computational complexity can be significantly reduced using fast Gaussian computation through Gaussian short lists.

So far we presented a Gaussian mixture language model (GMLM) and showed how to use it for calculating word probabilities either on its own as in Equation (5), or to smooth an existing N-gram as in Equations (6) and (7). We also discussed how this language model can address the *generalizability* issue raised in the introduction. In Section 3 we will show first how to adapt the model parameters using a limited amount of adaptation or test data, then in Section 4

[2] We limit histories to N-gram of words for simplicity of presentation but this limitation can be easily relaxed.

[3] Subscript is dropped for convenience.

we will discuss parameter estimation including the projection matrices A, B, and the Gaussian mixture model parameters for each word or word class.

3. MODEL ADAPTATION

One of the most important problems that face N-gram language models is adaptation. Acoustic models are easily adapted from a relatively small number of sentences by utilizing the structure in the model using techniques like maximum likelihood linear regression (MLLR) [3]. Attempts to adapt language models were in general not successful and lead in most situations to modest improvements if any. The main reason is that it is inherently difficult, in the absence of any structure, to adapt a large number of parameters from little adaptation or test data. The introduced Gaussian mixture model can be used to overcome this problem.

First the transcription of adaptation data or equivalently the output of a recognizer is a stream of words which can be transformed into the continuous space using Equations (1), and (3) and the values of the matrices A and B obtained during training. This results in a sequence of vectors which can be used to adapt the parameters of the Gaussian mixture models of the words or word classes in Equation (4). This adaptation can be efficiently achieved using techniques like MLLR as stated above. In the case that multiple transformations are needed the Gaussians can be clustered in a tree structure using well known methods for acoustic models. For example, in the case of MLLR adaptation and using only one transformation matrix Z, we can write each adapted mean $\mu'_{w,k} = Z\mu_{w,k}$, which is used in Equation (4) to calculate the adapted values of $p(y|w)$, which in turn can be used in Equations (5), (6), or (7) to calculate the adapted language model. It should be noted that the transformation matrix Z can be calculated using standard MLLR [3].

4. PARAMETER ESTIMATION

Starting from a text corpus we want to estimate the transformation matrices A, and B, and the Gaussian mixture parameters for each word w which are given by $\{c_{w,k}, \mu_{w,k}, \Sigma_{w,k} \quad 1 \leq k \leq K_w\}$, where K_w is the number of components for word w. We recall from Section 2 that the text corpus can be written as a sequence of indicator vectors each of size equal to the vocabulary size V, and also that each N-gram history (N-1 words) is labeled as belonging to the following word or word class. We will show first how to calculate the matrix A, then move to estimating B, and the Gaussian mixture models.

A straightforward way to think about the estimation of A is using linear discriminant analysis (LDA) [6]. For each word we limit the history to the previous word for the purpose of estimation[4] of A. Using the history indicator vectors we can estimate the within class covariance C, and the between class covariance D, and hence estimate the projection matrix A as consisting of the M eigenvectors corresponding to the M largest eigenvalues of $C^{-1}D$. However, note that the matrices are of size $V \times V$ and hence this decomposition is very expensive for typical vocabulary sizes around 60K words. Thus we use a simpler way. First we form the word co-occurrence matrix E, where e_{ij} is the number of times[5] word i follows word j, this is similar in spirit to latent semantic analysis (LSA) e.g. [4], but using the word co-occurrence matrix instead of the word-document co-occurrence matrix. Starting from a singular value decomposition (SVD) of E we from the projection A from M singular vectors corresponding to the M largest singular values. The matrix E is still of dimension $V \times V$, but it is typically sparse, and our need of only M singular values allows the use of efficient procedures [7] to perform the SVD. It is interesting to note that, for indicator vectors, the co-occurrence matrix E is an approximation of the between class covariance D, and hence decomposing the matrix E is equivalent to performing LDA under the assumption that the within class covariance is the unit matrix.

Once the A is estimated, all the training data can be mapped into vectors of size M, and hence each N-gram history is mapped into a vector of size $M(N-1)$ that is associated to the following word. The matrix B projects from this $M(N-1)$ dimension to a lower dimension L. For example, if $M = 100$, $L = 50$, and a 4-gram history is used the matrix B should project a 300-dimensional vector to a 50-dimensional vector. Thus, the estimation of the projection matrix B and the Gaussian mixture model parameters of different words is similar to that of estimating a projection matrix and the Gaussian mixture model parameters in acoustic modeling. This is typically accomplished using LDA followed by a maximum likelihood linear transform (MLLT) for the projection matrix and the well-known maximum likelihood estimation of Gaussian mixture models. However, it is important to note that in our case the assignment of vectors to words is known, in contrast to acoustic model estimation where a segmentation procedure is carried out for each utterance, and this facilitates the whole estimation process.

To summarize, parameter estimation amounts to the following two steps:

1. Estimate the projection matrix A by SVD of the word co-occurrence matrix E.
2. Once the A is estimated, both the projection matrix B and the Gaussian mixture model parameters can be estimated using standard techniques from acoustic modeling, for example, LDA+MLLT transform, and the well known EM approach for estimating Gaussian mixture models.

5. EXPERIMENTAL RESULTS

We apply the proposed GMLM to dialectal Arabic speech recognition. The system is used as a front-end to a speech-to-speech translation system [8]. The feature space has 40 dimensions, obtained by projecting using (LDA+MLLT) a 216 dimensional space, that consists of concatenating nine 24-dimensional MFCC coefficients. The acoustic model is trained on about 200 hours of speech using maximum-likelihood estimation. 33 graphemes are used as basic phonetic units, each grapheme is represented by 3 states. These states are clustered into about 3K leaves, and each leaf is modeled using a GMM. The total number of Gaussians is about 60K. The language model data has about 400K sentences comprising about 2M words. The vocabulary has 98K words. Trigram Language models are built using Kneser-Ney smoothing. The test data has 2719 sentences spoken by 19 speakers. A Viterbi decoder uses the acoustic and language models to generate 160 alternatives per sentence on average. The baseline LM, the GMLM, and an interpolation of both the baseline LM and GMLM are used to rescore the

[4] This is done for computational reasons, and theoretically larger histories can be used.

[5] Different normalizations can be applied to the raw counts.

N-best list. The interpolated LM uses uniform weights. Because the current implementation of the GMLM is computationally intensive, mainly due to the denominator term in Equation (6), we limited the rescoring to the 5-best utterances in the list. This may limit the potential benefit that could be obtained from the proposed method. We will address this issue in the future by caching the Gaussian scores and using Gaussian short lists.

The GMLM is trained as follows. First note that a GMM is needed for each word, and hence a sufficient number of observations (histories) must exist for the word. For this reason we limited the construction of GMMs for words that occur 100 times or more. From the original 98K vocabulary we ended up in 2950 words. For the rest of the words we had two implementations. In the first, we mapped all the remaining words into one class, a sort of filler or unknown word. In the second, we clustered the remaining words into 200 classes using the SRILM toolkit [9]. We will give results on both implementations below.

In both cases, after selecting the vocabulary the rest of the implementation is the same. First, the word co-occurrence matrix E is constructed, where e_{ij} is the number of times that word i follows word j. The count is then smoothed as $e_{ij}^{'} = \log(1 + e_{ij})$. Singular value decomposition (SVD) is performed on the resulting smoothed count matrix. The subspace projection method provided in the publicly available SVDPACK software is used to compute the 100 highest singular values and their corresponding singular vectors. The resulting singular vectors are used to construct the projection to a 100-dimensional space. To create a trigram, the vectors corresponding to the two words in the history are stacked to form a 200-dimensional vector. Thus, a document can be represented by a sequence of 200-dimensional vectors corresponding to the history of each of its constituent words. Beyond this step training is similar to acoustic model training. First we use an LDA+MLLT projection to reduce the dimensionality to 50 and then build diagonal covariance GMMs. However, it should be noted that in contrast to acoustic model training neither alignment nor decision tree clustering is needed. This makes training simpler.

For the first implementation, in calculating the GMLM score as in Equation (6) the score coming from the Gaussian mixture is divided by 40 to balance its dynamic range with that of the N-gram. Also, because some of the words are not observed in the GMM, the final LM is calculated as

$$P^{'}(w|h) = \begin{cases} \alpha(h)P_s(w|h) & \text{if } w \in \text{GMLM;} \\ P(w|h) & \text{Otherwise.} \end{cases} \quad (8)$$

where the smoothed LM in equation (6) is basically modified as:

$$P_s(w|h) = \frac{P(w|h)p(y|w)}{\sum_{v \in GMLM} P(v|h)p(y|v)} \quad (9)$$

and the normalizing constant $\alpha(h) = \sum_{v \in GMLM} P(v|h)$. While for the second implementation Equation (7) was used where words that have their own GMM are considered as their own class.

Table 1 basically shows that the GMLM alone, for both implementations, is only slightly worse than the N-gram. This makes sense as it is trying to approximate the N-gram by fitting Gaussian mixtures.

6. SUMMARY

In this paper we first point out two drawbacks of the widely used N-gram language model that we refer to as *Generalizability* and *adaptability*. We argue that both these problems can be addressed in a continuous parameter space, and propose to use a Gaussian mixture language model (GMLM) to address these issues. It is shown how this model can be used alone or in conjunction with a a conventional N-gram model to calculate word probabilities. An interesting feature of the proposed technique is that many methods developed for acoustic models can be easily ported to GMLM. We developed two implementations of the proposed model for large vocabulary Arabic speech recognition. Future work will include improving the efficiency of the implementation, developing adaptation strategies, and testing on different ASR tasks.

	0.01	0.02	0.03	0.04	best
N-gram	33.5	33.1	33.9	34.9	33.1
GMLM	33.7	33.4	34.0	34.8	33.4
GMLM+N-gram	33.9	33.5	33.3	33.2	33.2
GMLM-C	33.8	33.6	34.2	35.0	33.6
GMLM-C+N-gram	34.1	33.6	33.3	33.4	33.3

Table 1: Word error rate (%) when rescoring N-best lists with GMLM, N-gram, and their interpolation. The results are for different values of acoustic weight used in rescoring. The best value is given in the last column.

7. REFERENCES

[1] F. Jelinek, Statistical Methods for Speech Recognition, MIT Press, January 1998.

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

[3] C.J. Legetter and P.C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," Computer Speech & Language, vol. 9, pp. 171-185, 1995.

[4] J. Bellegarda, "Large Vocabulary Speech Recognition with Multispan Language Models," in IEEE Transactions on Speech and Audio Processing, vol. 8, No. 1, pp. 76-84, January 2000.

[5] H. Schwenk, and J.L. Gauvain, "Using Continuous Space Language Models for Conversational Telephony Speech Recognition," in IEEE Workshop on Spontaneous Speech Processing and Recognition, Tokyo, Japan, April 2003.

[6] R. Duda, P. Hart, and D. Stork, Pattern Classification (Second Edition). Wiley-Interscience, October 2000.

[7] G. Golub, and C. F. Van Loan, Matrix Computations, Johns Hopkins University Press, June 1996.

[8] Y. Gao, B. Zhou, L. Gu, R. Sarikaya, H.-K. Kuo. A.-V.I. Rosti, M. Afify, W. Zhu, "IBM MASTOR: Multilingual automatic speech-to-speech translator," Proc. ICASSP'06, Tolouse, France, 2006.

[9] A. Stolcke, "SRILM – an extensible language modeling toolkit," Proc. ICSLP'02, Denver, Colorado, Sept., 2002.