MIXTURE OF MIXTURE N-GRAM LANGUAGE MODELS

Haşim Sak, Cyril Allauzen, Kaisuke Nakajima, Françoise Beaufays

Google

{hasim,allauzen,kaisuke,fsb}@google.com

ABSTRACT

This paper presents a language model adaptation technique to build a single static language model from a set of language models each trained on a separate text corpus while aiming to maximize the likelihood of an adaptation data set given as a development set of sentences. The proposed model can be considered as a mixture of mixture language models. The mixture model at the top level is a sentence-level mixture model where each sentence is assumed to be drawn from one of a discrete set of topic or task clusters. After selecting a cluster, each n-gram is assumed to be drawn from one of the given n-gram language models. We estimate cluster mixture weights and n-gram language model mixture weights for each cluster using the expectation-maximization (EM) algorithm to seek the parameter estimates maximizing the likelihood of the development sentences. This mixture of mixture models can be represented efficiently as a static n-gram language model using the previously proposed Bayesian language model interpolation technique. We show a significant improvement with this technique (both perplexity and WER) compared to the standard one level interpolation scheme.

Index Terms— language model, adaptation, interpolation, mixture models, bayesian, speech recognition

1. INTRODUCTION

Speech-enabled interfaces on mobile devices allow users to accomplish many different tasks thanks to automatic speech recognition (ASR) systems. Voice search, e-mail and SMS dictation, voice input into any text field, location/business name search, voice actions ("set alarm to 8 a.m.") are some example tasks. The variety of tasks and domains presents some challenges for language modeling in ASR systems. To achieve the best speech recognition accuracy, we need to train and build separate language models each optimized or adapted for a specific task or domain. However, having many language models in the production system and employing the best one for an input utterance given its context is not trivial.

One approach as proposed in [1] is on-demand language model interpolation. In this method, a set of interpolation weights for a number of n-gram language models is determined for each contextual information (such as application id) using a set of development sentences from each context. Since there may be a large number of contexts, the language models are interpolated on-demand, either in a first-pass recognition or second-pass of rescoring lattices. For this purpose, the component n-gram language models are compactly represented as a finite-state transducer, and the transition weights are dynamically computed using the optimized interpolation weights for a given context.

Using a *dynamically-interpolated* LM in the first-pass recognition demands significantly more computation than using a *statically-interpolated* LM due to the overhead of accessing a larger set of weights and combining them on-demand per utterance. Other alternative of using a dynamically-interpolated LM in the second-pass of rescoring lattices recognized with a task-independent LM in the first-pass is quite fast thanks to much reduced lattice search space. However, if a task-independent LM - which is a statically-interpolated LM whose mixture weights are determined to minimize the perplexity of development set from all the tasks - is used in the first-pass and a dynamically-interpolated LM is used in the second-pass, the recognition accuracy improves only 5.1%. In comparison, the dynamically-interpolated LM when used in the first-pass gives 11.2% improvement over the task-independent LM.

Another method - *Bayesian language model interpolation* has been proposed to replace this two-pass strategy which incurs significant additional search errors with a one-pass system using a static task-independent LM that is built to be as close as possible to the dynamically-interpolated LM [2]. It has been shown that using the statically-interpolated LM obtained with the Bayesian LM interpolation method achieves about half of the recognition accuracy improvements that is possible with the dynamically-interpolated LM in the first-pass.

With the advent of one-box interfaces where the user may input information from multiple sources or domains, the distinction of context for an utterance disappeared. For instance, Google Now application on Android and iOS mobile devices allows user to speak queries, ask questions, give commands and trigger actions and dictate e-mail and SMS messages. Therefore, a single system needs to handle all these different tasks.

This paper provides a recipe for building single static target n-gram LM by interpolating set of source LMs pretrained on different training corpora/topics. The resulting interpolated model is optimized for the perplexity on a representative development set. The development set is assumed to be a collection of sentences each independently drawn from a set of latent topics (i.e. no topic definition or topic labeling is available for the development and test data; the amount of data for different topics can be very unbalanced). We show that a two level interpolation scheme is appropriate for this scenario combining n-gram level interpolation and sentence level interpolation. N-gram level interpolation, where n-gram probabilities from different source LMs are linearly interpolated, is suitable for synthesizing LMs for the latent topics. Each topic LMs is used to evaluate probability of a test sentence. The resulting probabilities are further linearly interpolated. The resulting interpolation scheme can be seen as *mixture of mixture n-gram language models*. The resulting interpolated LM, which is derived on development data, can be represented as standard n-gram LM using the Bayesian language model interpolation technique [2], which is fixed for testing.

2. RELATED WORK

There are a large number of studies on statistical language model adaptation as reviewed in [3]. Mixture language models as a common adaptation technique linearly interpolate a set of n-gram language models each trained on a separate topic or task domain using a set of mixture weights (interpolation weights) estimated to maximize the likelihood of the adaptation data (development set) [4, 5]. The interpolation can be performed at the n-gram level [4] or sentence level [5]. Other adaptation techniques have been proposed for dialog systems, which use the dialog state to adapt the language models instead of pre-defined topics [6, 7, 8].

This study differs from the related work in the literature in various ways. First, the source corpora for the language models is mostly a collection of sentences or text queries where there is no document structure to model with topic models. Second, there are a large number of tasks in the development set as determined by the applications, where the task domains are very diverse. Third, we use the probabilities assigned to each sentence by a set of language models to cluster the development set into a given number of topic or task clusters rather than clustering the training data. Then, a set of n-gram mixture weights for each cluster are estimated to maximize the likelihood of the sentences in that cluster. The proposed model can be considered as a sentence level mixture model of n-gram level mixture models.

3. MIXTURE OF MIXTURE N-GRAM LANGUAGE MODELS

We assume that we are given a set of language models each trained on a separate corpus possibly from different sources or domains, and a collection of sentences as a development set. The goal is to estimate and build a statically-interpolated language model maximizing the likelihood of the development sentences.

Let $G_1, ..., G_M$ be the set of M given n-gram language models and $p_m(w|h) = p(w|h, m)$ be the probability that word w follows history h as estimated by model G_m. A simple n-gram level linear interpolation of LMs gives us a statically-interpolated language model. This mixture model is simply mixtures of n-gram probabilities from the component LMs and the mixture weights $\lambda_1, ..., \lambda_M$ can be optimized using the EM algorithm to maximize likelihood of development sentences:

$$p(w|h) = \sum_{m=1}^{M} \lambda_m p_m(w|h)$$

In this paper, we show that we can estimate and build a better statically-interpolated LM by using sentence-level cluster mixture models of n-gram level mixture models. In this mixture of mixture models, the probability of a sentence (word sequence) $\boldsymbol{w}$ is defined as follows:

$$\begin{aligned} p(\boldsymbol{w}) &= \sum_{c=1}^{C} \gamma_c p_c(\boldsymbol{w}) \\ &= \sum_{c=1}^{C} \gamma_c \prod_i p_c(w_i|h_i) \\ &= \sum_{c=1}^{C} \gamma_c \prod_i \sum_{m=1}^{M} \lambda_{c,m} p_m(w_i|h_i) \end{aligned}$$

where C is the number of clusters, $\gamma_1, ..., \gamma_C$ is the sentence-level cluster mixture weights, $p_c(.|.)$ is a probability distribution of n-gram level mixtures for the c^{th} cluster. $\lambda_{c,m}$ for $m = 1...M$, $c = 1...C$ is the n-gram level mixture weigth for the m^{th} n-gram language model G_m. As formulated, this interpolated model is a mixture of mixture models with model parameters $\theta = \{\gamma_c, \lambda_{c,m} : 1 \leq c \leq C, 1 \leq m \leq M\}$.

Given a set of sentences $\boldsymbol{W}$, we can find the maximum likelihood estimate of these unknown mixture parameters using the EM algorithm. We assume that each sentence is drawn from an unobserved latent cluster and each n-gram is drawn from a latent n-gram language model. The EM algorithm is an iterative approach to find the unknown parameters $\boldsymbol{\theta}$ that maximizes the likelihood of the observed data $\boldsymbol{W}$:

$$\begin{aligned} \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \boldsymbol{W}) &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{W}|\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{\theta}) \end{aligned}$$

3.1. Estimation of Mixture Parameters

In the proposed model, we assume there is a latent unobserved cluster variable corresponding to each sentence. We can consider this as a clustering problem where each sentence in the development set is assigned to one of C clusters. In the formulation of this clustering problem, we can apply two types of EM algorithm. In the (soft-)EM type algorithm, we estimate the probability of each cluster assignment for each sentence and use the probabilities associated with a particular cluster assignment to compute a weighted estimation of n-gram mixture parameters for each cluster. In the hard-EM type algorithm (which is an approximation to EM algorithm), we make a hard choice for the cluster assignments of the sentences where each sentence is assigned to a single cluster given the current model parameters, and use the current assignments to estimate n-gram mixture parameters for each cluster.

In the *hard-EM algorithm*, we first randomly select C sentences from the development set $\boldsymbol{W}$ and estimate the mixture coefficients for each sentence that locally maximizes the likelihood of that sentence using the EM algorithm. For each of the C sentences, we create a corresponding cluster whose initial mixture weights are those estimated for the sentence. We then iterate between two steps:

- Assign each sentence $\boldsymbol{w}$ in $\boldsymbol{W}$ to the cluster whose mixture weights maximizes the likelihood of the sentence. This gives a clustering as a set of sentences S_c for each cluster c:

$$S_c = \{\boldsymbol{w} : p_c(\boldsymbol{w}) \geq p_i(\boldsymbol{w}), \forall\, 1 \leq i \leq C\}$$

- Calculate the new mixture weights for each cluster to be the mixture weights optimizing the likelihood of the sentences in each cluster using the EM algorithm. The mixture weights for n-gram language models for each cluster can be found using the following iterative parameter update EM solution:

$$\begin{aligned} \lambda'_{c,m} &= \frac{1}{||S_c||} \sum_{\boldsymbol{w} \in S_c} \sum_{i=1}^{|\boldsymbol{w}|} p(m|w_i, h_i, c) \\ &= \frac{1}{||S_c||} \sum_{\boldsymbol{w} \in S_c} \sum_{i=1}^{|\boldsymbol{w}|} \frac{\lambda_{c,m} p_m(w_i|h_i)}{\sum_{j=1}^{M} \lambda_{c,j} p_j(w_i|h_i)} \end{aligned}$$

  where $||S_c|| = \sum_{\boldsymbol{w} \in S_c} |\boldsymbol{w}|$ is the number of all words in the cluster c.

We can stop this iterative process after a fixed number of iterations or when the total likelihood of the development sentences given the current parameters does not improve. Finally, the mixture weights for each cluster is calculated as $\gamma_c = \frac{|S_c|}{|W|}$.

In the *(soft-)EM algorithm*, we initialize the cluster mixture weights γ_c and n-gram mixture model weights $\lambda_{c,m}$ for each cluster randomly. We then iteratively update the model parameters as follows:

$$\begin{aligned} \lambda'_{c,m} &= \frac{1}{N_c} \sum_{\boldsymbol{w}} p(c|\boldsymbol{w}) \sum_{i=1}^{|\boldsymbol{w}|} p(m|w_i, h_i, c) \\ &= \frac{1}{N_c} \sum_{\boldsymbol{w}} p(c|\boldsymbol{w}) \sum_{i=1}^{|\boldsymbol{w}|} \frac{\lambda_{c,m} p_m(w_i|h_i)}{\sum_{j=1}^{M} \lambda_{c,j} p_j(w_i|h_i)} \end{aligned}$$

$$\gamma'_c = \frac{1}{|\boldsymbol{W}|} \sum_{\boldsymbol{w}} p(c|\boldsymbol{w})$$

where N_c and $p(c|\boldsymbol{w})$ is defined as follows:

$$N_c = \sum_{\boldsymbol{w} \in \boldsymbol{W}} p(c|\boldsymbol{w}) |\boldsymbol{w}|$$

$$p(c|\boldsymbol{w}) = \frac{\gamma_c p_c(\boldsymbol{w})}{\sum_{i=1}^{C} \gamma_i p_i(\boldsymbol{w})}$$

$$p_c(\boldsymbol{w}) = \prod_{i=1}^{|\boldsymbol{w}|} p_c(w_i|h_i) = \prod_{i=1}^{|\boldsymbol{w}|} \sum_{m=1}^{M} \lambda_{c,m} p_m(w_i|h_i)$$

Note that the EM algorithm does not guarantee that the iterative updating of parameters converges to a global maximum likelihood estimation of the parameters. The EM algorithm may converge to a local maximum of likelihood function depending on the initial parameters.

3.2. Bayesian Language Model Interpolation

A language model interpolation technique - Bayesian LM interpolation - has been proposed to build a static task-independent LM to approximate a task-dependent dynamically-interpolated LM using the task priors and the estimated mixture weights for each task [2]. This technique is used to build a statically-interpolated n-gram LM with standard finite-state representation for our mixture of mixture language models. The only difference in the Bayesian LM interpolation formulation (see [2]) is that the task prior probabilities are replaced with the cluster mixture weights as follows:

$$p(\boldsymbol{w}) = \prod_{i} \sum_{m=1}^{M} \alpha_{m,h_i} p_m(w_i|h_i)$$

where state-dependent mixture weights α_{m,h_i} is defined as:

$$\alpha_{m,h_i} = \sum_{c=1}^{C} p(c|h_i) \lambda_{c,m}$$

$$p(c|h_i) = \frac{p(h_i|c)\gamma_c}{\sum_{c=1}^{C} p(h_i|c)\gamma_c}$$

$$p(h_i|c) = \prod_{j=1}^{i} p(w_j|h_j, c) = \prod_{j=1}^{i} \sum_{m=1}^{M} \lambda_{c,m} p_m(w_j|h_j)$$

using the cluster mixture weights γ_c and n-gram mixture model weights $\lambda_{c,m}$ for each cluster estimated with the EM algorithm.

4. SYSTEMS & DATA

For the language model interpolation experiments, we trained 10 5-gram language models individually with Katz-backoff from 10 separate data sources. Each language model is pruned to 23 million n-grams using Stolcke pruning [10]. The data sources used vary in size, from a few million to a few billion sentences. They consist of web documents, typed queries, SMS messages, voice actions, queries from various applications, dictated messages and speech recognition transcripts of utterances filtered with a threshold on recognition confidence scores. The transcripts are supposed to provide domain adaptation with self-supervision. All the user data used in the language models is anonymized. They are in the written-domain and the language models are trained in the written-domain without converting the sources to the verbal domain [11]. The language models are statically-interpolated and the final model is pruned again to 23 million n-grams. The vocabulary size of the final interpolated LM is 2.8 million.

We use a combined development set from 5 separate sources supposed to be representative of the expected traffic as the language model adaptation data in the interpolation experiments. The combined development set contains 92K sentences. Each source is obtained by hand-transcribing randomly selected anonymized utterances from the speech recognition logs. They consist of voice actions, voice search queries, queries from specific applications, and dictated messages.

Our acoustic models used in the speech recognition experiments are standard 3-state context dependent (triphone) HMM models which use a deep neural network (DNN) to estimate HMM-state posteriors [9]. The DNN model is a standard feed-forward neural network with 8 hidden layers of 2560 nodes. The input layer is the concatenation of 26 consecutive frames of 40-dimensional log filterbank energies calculated on 25ms windows of speech every 10ms. The 7969 softmax outputs estimate the posterior of each state.

5. EXPERIMENTAL RESULTS

We experimented with both soft and hard-clustering of development sentences to estimate and build the mixture of mixture n-gram language models. We saw that soft-clustering performs slightly better. Therefore, we choose to report only the perplexity results using the soft-clustering in this section.

Figure 1 shows the convergence of development set perplexity with increasing number of iterations in the (soft-)EM algorithm. We also vary the number of classes used in the clustering to see how it affects the development set perplexity. We see that the perplexity starts to converge around 12 clusters and 10 iterations of the EM algorithm. The interpolated language model with the 1-class corresponds to standard linear interpolation with interpolation weights optimized on all the development sentences. The development set perplexity with the 12-class model improves significantly by 17.5% over the 1-class model.

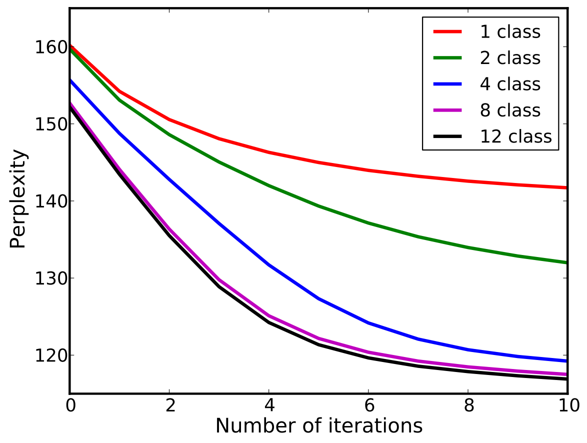
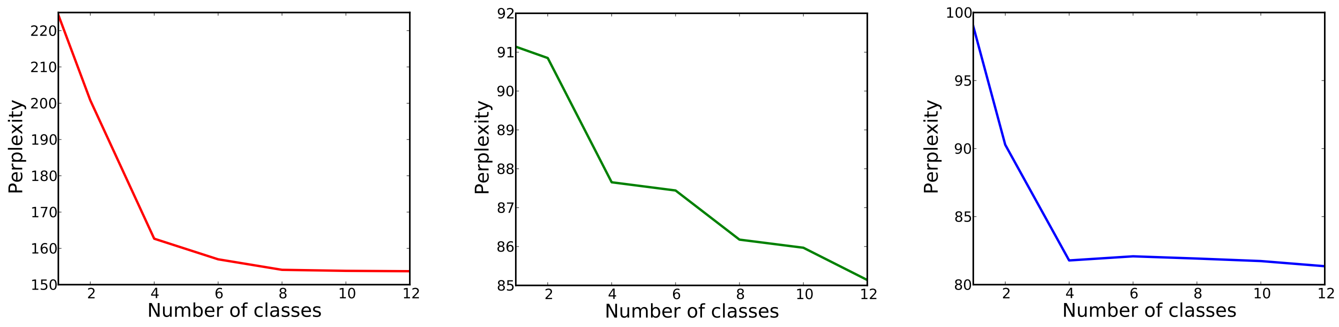


Fig. 1. The *development* set perplexity versus number of iterations in the (soft-)EM algorithm for various number of clusters.

Figure 2 shows the test set perplexity for three test sets used in the speech recognition experiments with various number of classes obtained after 10 iterations. The first test set – *Maps* has 64K words and consists of utterances from the Google maps application. The second one – *Search* has 98K words and consists of voice search utterances. The final one – *Unified* has 136K words and is a unified set of voice search and dictation utterances. The perplexity improvements for the 12-class model versus the 1-class model are 31%, 6.5% and 18% on *Maps*, *Search* and *Unified* test sets, relatively. The relatively large improvement on the *Maps* system is expected since the portion of maps like queries in the development set is relatively small and the development set distribution is biased towards voice search queries.

We evaluate the speech recognition accuracy of the interpolated language models on three test sets which are obtained by hand-transcribing anonymized and randomly selected utterances from our speech recognition system logs. All test sets are transcribed in the written domain (e.g. “set alarm for 12:30” rather than “set alarm for twelve thirty”) and we measure the speech recognition accuracy in the written domain.

Table 1 compares the word error rates (WERs) of mixture of mixture language models (statically interpolated with Bayesian LM interpolation technique) with various number of clusters and a Bayesian interpolated language model using



(a) *Maps* test set (b) *Search* test set (c) *Unified* test set

Fig. 2. Perplexity for test sets.

Class	1	4	8	12	Bayesian
Search	13.8	13.7	13.5	**13.5**	13.6
Maps	13.4	12.5	12.5	**12.4**	12.8
Unified	11.4	11.1	11.1	11.1	**11.0**

Table 1. Comparison of word error rates on three test sets using mixture of mixture language models and a Bayesian interpolated model.

task information on three test sets. Although the Bayesian model uses the contextual information (e.g. application id) in the development set to estimate the task priors and task specific mixture weights, mixture of mixture models generally performs better by estimating cluster and n-gram mixture weights using only the development sentences.

6. CONCLUSION

We presented mixture of mixture n-gram language models. This model is a statically-interpolated language model from a set of language models and aims to maximize the likelihood of a development set of sentences by clustering these sentences using the probabilities assigned by the component language models. The cluster and n-gram mixture weights are estimated with the EM algorithm. Using the estimated model parameters, we build a statically-interpolated n-gram language model using the Bayesian language model interpolation technique. We show that mixture of mixture language models results in better speech recognition accuracy when we need to build a single speech recognition system that needs to handle various types of recognition tasks.

7. REFERENCES

[1] Brandon Ballinger, Cyril Allauzen, Alexander Gruenstein, and Johan Schalkwyk, "On-demand language model interpolation for mobile speech input," in *Interspeech*, 2010, pp. 1812–1815.

[2] Cyril Allauzen and Michael Riley, "Bayesian language model interpolation for mobile speech input," in *Proceedings of Interspeech*, 2011, pp. 1429–1432.

[3] Jerome R. Bellegarda, "Statistical language model adaptation: review and perspectives," *Speech Communication*, vol. 42, pp. 93–108, 2004.

[4] Reinhard Kneser and Volker Steinbiss, "On the dynamic adaptation of stochastic language models," in *Proceedings of ICASSP*, 1993, pp. 586–589.

[5] Rukmini M. Iyer and Mari 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.

[6] Frank Wessel, Andrea Baader, and Hermann Ney, "A comparison of dialogue-state dependent language models," in *ESCA Tutorial and Research Workshop (ETRW) on Interactive Dialogue in Multi-Modal Systems*, 1999, pp. 93–96.

[7] Wei Xu and Alexander I. Rudnicky, "Language modeling for dialog system," pp. 118–121, 2000.

[8] Karthik Visweswariah and Harry Printz, "Language models conditioned on dialog state," in *INTERSPEECH*, 2001, pp. 251–254.

[9] Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proceedings of Interspeech*, 2012.

[10] Andreas Stolcke, "Entropy-based pruning of backoff language models," in *DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[11] Hasim Sak, Francoise Beaufays, Kaisuke Nakajima, and Cyril Allauzen, "Language model verbalization for automatic speech recognition," in *Acoustics, Speech and Signal Processing, 2013. ICASSP 2013. IEEE International Conference on*, 2013.