Bayesian Language Model Interpolation for Mobile Speech Input

Cyril Allauzen, Michael Riley

Google Research, 76 Ninth Avenue, New York, NY, USA

allauzen@google.com, riley@google.com

Abstract

This paper explores various static interpolation methods for approximating a single dynamically-interpolated language model used for a variety of recognition tasks on the Google Android platform. The goal is to find the statically-interpolated first-pass LM that best reduces search errors in a two-pass system or that even allows eliminating the more complex dynamic second pass entirely. Static interpolation weights that are uniform, prior-weighted, and the maximum likelihood, maximum a posteriori, and Bayesian solutions are considered. Analysis argues and recognition experiments on Android test data show that a Bayesian interpolation approach performs best.

Index Terms: speech recognition, language modeling, language model interpolation

1. Introduction

Various speech-enabled features are available on the Android mobile operating system. These include search-by-voice, voice input into any text field and an application developer's API [1, 2, 3]. By voice, a user can search for a restaurant in Google Maps, dictate a message in an SMS client, and look for mobile applications in the Android Market, among the many options. This diverse input makes accurate language modeling, in particular, a challenge.

As described in [2], the following methods are employed for this language modeling problem. First, K component n-gram language models (LMs) are constructed from available large corpora deemed relevant to the recognition tasks at hand. These could include, for example, SMS messages and text search queries. Second, applications and text fields are pooled into T tasks, such that the very frequent text fields (e.g., Google Maps) are individual tasks while the rarest fields are pooled together into a single 'other' task. For each task, a representative amount of text input is collected into a development set. Next, K mixture weights are selected for each task that minimize the perplexity of the corresponding development data when used to create an interpolated LM from the component n-gram models. Finally, at recognition time a mixture LM for each task is used with its corresponding task-specific mixture weights. This approach resulted in an 11.2% reduction in relative WER compared to a single task-independent LM whose interpolation weights were selected to minimize the perplexity of all the task development text data.

Deploying such a system presents its own challenges. Given T is over one hundred, creating T statically-interpolated task LMs is impractical. They can not be switched efficiently on a single server and distributing different task LMs among different servers creates serious provisioning and task load-balancing problems. As described in [2], a single *dynamically-interpolated* language model can be used. This model stores the component LM probabilities separately and when provided with a set of mixture weights, it performs the linear interpolation on-demand. In this way, the mixture weights can be changed for each utterance.

Unfortunately, the dynamically-interpolated LM presents real-time challenges as well. If it is used in a first-pass recognition, the overhead of accessing the larger set of weights and combining them on-demand per utterance takes significantly more computation than using a statically-interpolated LM. It also inhibits any static compilation or dynamic caching across utterances of the underlying recognition network. On the other hand, the dynamically-interpolated LM can be used in a second-pass recognition of lattices created from a first-pass using a single task-independent LM. In this case, the dynamic interpolation is quite fast given the greatly reduced lattice search space. However, if the task-independent LM consists of a statically-interpolated LM whose mixtures weights are selected to minimize the perplexity of all the task development text data, then the dynamically-interpolated second-pass LM reduces the WER only 5.1% over the 1st-pass compared to the 11.2% reduction when the dynamically-interpolated LM is used in the 1st pass [2]. In other words, significant additional search errors are incurred with this two-pass strategy.

Faced with these issues, one can either try to speed up the dynamically-interpolated first-pass LM or close the gap between using it in a first and second pass. This paper addresses the latter problem. The goal is to find a static task-independent LM that is 'as close as possible' to the dynamically-interpolated LM. Such a task-independent LM can then be used as a better first-pass LM for the task-dependent second pass. If it were good enough, it might even be possible to eliminate the second pass.

In Section 3, we describe and compare several possible ways of creating this task independent LM. These include interpolated LMs that use mixture weights that are (1) uniform, (2) prior-weighted, (3) maximum likelihood, (4) maximum a posteriori, and (5) Bayesian (state-dependent) solutions. We also formalize the notion of finding the static LM that is as close as possible to the dynamically-interpolated LM and show the Bayesian-interpolated LM optimizes this criterion. In Section 4, we present recognition experiments that show the Bayesian-interpolated LM out-performs other LMs in this scenario.

In many speech recognition domains using a smaller LM in a first pass, followed by a larger LM in the second pass gives nearly the same accuracy as using the larger LM in the first pass. For example, if the smaller LM is obtained by entropy pruning the second-pass LM [4] then this typically holds [5] provided the lattice beams are reasonably large. In fact, the first-pass LM can be pruned to a very small size without significantly impacting the second-pass accuracy. As a final set of experiments in Section 4, we explore how small we can make the task-independent first-pass LM with little loss in the second-pass task-dependent rescoring.

2. Related Work

There is a large literature on language model adaptation, in general, and mixture language models, in particular [6, 7]. One scenario is topic modeling, where parts of a corpus are labeled by pre-defined topics and sub-language models are built on these topics and used in interpolation [8, 9]. Another scenario is dialog modeling, where the dialog state is used instead to define the sub-corpora [10, 11, 12].

One difference between our and these scenarios is that in our case the source corpora for the language models is usually distinct from the tasks. The sources are chosen for their availibility of large of amounts of hopefully relevant text. The tasks are determined by our applications, there are large number of them, and we often have only a modest amount of transcribed development data for them. Another difference is that our tasks are very diverse; many are independent domains.

3. Interpolated Language Models

Assume that there are K n-gram language models trained on the available large corpora. For the mobile speech problem, $K = 6$ and includes SMS, spoken search queries, and text search queries. Call these $G_1, \ldots, G_K$ and let $p_k(w|h)$ be the probability that word w follows history h according to LM G_k. Assume there are T tasks that are to be performed and that $p(t)$ gives the a priori probability of performing a given task. For the mobile speech problem, T is approximately one hundred and includes SMS, Google Maps, Gmail, and Android Market search. Note there are many more tasks than there are available, relevant LM text corpora.

For the t-th task, assume the mixture weights $(\lambda_{1,t}, \ldots, \lambda_{m,t})$ have been selected such that:

$$p(w|ht) = \sum_{k=1}^{K} \lambda_{k,t} p_k(w|h) \tag{1}$$

gives the mixture language model that minimizes the perplexity on development data from the t-th task. Assume that this *task-dependent* interpolated LM is used in a second-pass recognition of that task.

Our goal, for efficiency reasons explained in the introduction, is to find a *task-independent* LM that can be used in a first-pass that is as close as possible to the task-dependent LM, gives the fewest additional search errors in a two-pass scenario and potentially, if good enough, could be used to eliminate the dynamic second-pass entirely. In this section, the LMs considered are restricted to those that can be generated from the component LMs, the task mixtures weights and the task prior probabilities. In other words, $p_k(w|h)$, $(\lambda_{1,t}, \ldots, \lambda_{m,t})$ and $p(t)$ are the given information. Note $p_k(w|h)$ and $(\lambda_{1,t}, \ldots, \lambda_{m,t})$ are also the building blocks of the second-pass LM while $p(t)$ is found from the task usage frequencies. Saying the first-pass LM is task-independent here means that the input sentence's task is not provided to this stage during recognition.

Note the task priors $p(t)$ can be well-estimated by logging task usage. We have not assumed that we have development data that is sampled representatively according to the tasks but only enough per task to accurately estimate $(\lambda_{1,t}, \ldots, \lambda_{m,t})$.

We now present several possible task-independent LMs that are constructed from this information. All our models have the form of a mixture LM over the component LMs. We will use α_k to denote the mixture weights of the task-independent LM.

3.1. Uniform Interpolated LM

A particularly simple task-independent LM is to create the interpolated LM with uniform weights:

$$\alpha_k = \frac{1}{T} \tag{2}$$

This choice, however, makes no use of the a priori task probabilities, $p(t)$.

3.2. Prior-weighted Interpolated LM

Another simple model, but one taking advantage of the a priori task probabilities, uses the average of the per-task mixture weights:

$$\alpha_k = \sum_{t=1}^{T} \lambda_{k,t} p(t) \tag{3}$$

3.3. Maximum Likelihood Interpolated LM

The above choices, however, seem ad hoc. Consider instead:

$$t_{ml} = \operatorname*{argmax}_t p(\mathbf{w}|t) = \prod_i p(w_i|h_i, t) \tag{4}$$

where $\mathbf{w} = w_1 \ldots w_l$ is the input sentence of length l and $h_i = w_{i-n+1} \ldots w_{i-1}$ is the $(n-1)$-gram history. This equality holds under the n-gram Markov assumption which we assume is valid throughtout. The mixture weights $\alpha_k = \lambda_{k,t_{ml}}$ give the *maximum likelihood* solution.

A task-independent solution can, in principle, be computed by running T parallel recognitions using LMs with each of the task-specific mixture weights and then selecting the best scoring solution, but this is hardly practical. This model, like the uniform one, does not use the a priori task information.

3.4. Maximum A Posteriori Interpolated LM

A solution similar to the ML solution, but which takes advantage of the a priori task information, is:

$$t_{map} = \operatorname*{argmax}_t p(t|\mathbf{w}) = \operatorname*{argmax}_t p(t\mathbf{w}) \tag{5}$$

$$= \operatorname*{argmax}_t p(\mathbf{w}|t)p(t) \tag{6}$$

$$= \operatorname*{argmax}_t p(t) \prod_i p(w_i|h_i, t) \tag{7}$$

The mixture weights $\alpha_k = \lambda_{k,t_{map}}$ give the *maximum a posteriori* solution.

A task-independent solution can, in principle, be computed by running T parallel recognitions using LMs each premultiplied by the task-specific prior probabilities and with task-specific mixture weights and then selecting the best scoring solution. Like the maximum likelihood solution, however, this is hardly practical.

3.5. Bayesian Interpolated LM

Instead of selecting the a posteriori best task-independent mixture weights, one can average over the task priors to obtain the Bayesian solution:

$$p(\mathbf{w}) = \sum_{t=1}^{T} p(t, \mathbf{w}) = \sum_{t=1}^{T} p(\mathbf{w}|t)p(t) \tag{8}$$

$$= \sum_{t=1}^{T} p(t) \prod_i p(w_i|h_i, t) \tag{9}$$

Given the accurate task prior distribution that is available, this seems an optimal use of it. However, at first glance, it appears even less practical than the maximum likelihood and maximum a posteriori solutions.

However, note that:

$$p(\mathbf{w}) = \prod_i p(w_i|h_i) = \prod_i \sum_{t=1}^{T} p(w_i, t|h_i) \quad (10)$$

$$= \prod_i \sum_{t=1}^{T} p(t|h_i) p(w_i|h_i t) \quad (11)$$

$$= \prod_i \sum_{t=1}^{T} p(t|h_i) \sum_{k=1}^{K} \lambda_{k,t} p_k(w_i|h_i) \quad (12)$$

$$= \prod_i \sum_{k=1}^{K} \left[\sum_{t=1}^{T} p(t|h_i) \lambda_{k,t} \right] p_k(w_i|h_i) \quad (13)$$

$$= \prod_i \sum_{k=1}^{K} \alpha_{k,h_i} p_k(w_i|h_i) \quad (14)$$

where:

$$\alpha_{k,h_i} = \sum_{t=1}^{T} p(t|h_i) \lambda_{k,t} \quad (15)$$

Equation (14) corresponds to an interpolated LM with state-dependent mixture weights α_{k,h_i}. These can be computed from the task priors and the component LMs using (1) and

$$p(t|h_i) = \frac{p(h_i|t)p(t)}{\sum_{t=1}^{T} p(h_i|t)p(t)} \quad (16)$$

$$p(h_i|t) = \prod_{j=1}^{i} p(w_j|h_j, t). \quad (17)$$

As such, it is quite practical to construct this Bayesian mixture solution for even very large LMs. Further, if the components are backoff language models, the Bayesian LM is one as well. Its n-grams are the union of those of the component LMs with probablities calculated according to the sum in Equation 14 and with the backoff weights appropriately normalized.

The result is a task-independent statically-interpolated LM. Comparing (3) and (15), we see both use task probabilities to average the task mixture weights. However, in the former case these are a priori probablities, while in the latter case these are dependent on the current LM history.

A Bayesian approach has previously been explored in the simple case of one in-domain (the 'task') and one out-of-domain (the 'background') language model[13]. The state-dependent mixing parameter depends on the state probabilities from each model and a single prior probability that is tuned on development data.

3.6. Discussion

Equation (10) shows that the Bayesian interpolated LM represents $p(\mathbf{w})$; this is the actual distribution we can observe during the first pass and it is hard to see how we could do better. Another way to view this is to measure how close some distribution $q(\mathbf{w})$, independent of the prior distribution $p(t)$, is to the appropriately prior-weighted task-dependent distribution $p(t)p(\mathbf{w}|t)$), i.e. to the joint word task distribution $p(\mathbf{w}, t)$. Kullback Leibier (KL) divergence (or relative entropy) can be used to formalize this distance:

$$D(p(\mathbf{w}, t) \parallel q(\mathbf{w})p(t)) = \sum_{\mathbf{w},t} p(\mathbf{w}, t) \log \frac{p(\mathbf{w}, t)}{q(\mathbf{w})p(t)} \quad (18)$$

Looking for the best distribution:

$$\hat{q} = \operatorname*{argmin}_q D(p(\mathbf{w}, t) \parallel q(\mathbf{w})p(t)) \quad (19)$$

$$= \operatorname*{argmin}_q \sum_{\mathbf{w},t} p(\mathbf{w}, t) \log \frac{p(\mathbf{w}, t)}{q(\mathbf{w})p(t)} \quad (20)$$

$$= \operatorname*{argmin}_q \sum_{\mathbf{w},t} p(\mathbf{w}, t) \log \frac{p(\mathbf{w})}{q(\mathbf{w})} \frac{p(\mathbf{w}, t)}{p(\mathbf{w})p(t)} \quad (21)$$

$$= \operatorname*{argmin}_q D(p(\mathbf{w}) \parallel q(\mathbf{w})) + I_p(\mathbf{w}; t) \quad (22)$$

$$= p(\mathbf{w}) \quad (23)$$

Equation (23) follows since the non-negative KL distance is zero when the distributions are identical and the mutual information term is independent of q. This shows the Bayesian interpolated LM will minimize the distance to the joint word task distribution assuming the $p(t)$ and $p(\mathbf{w}|t)$ are accurately estimated.

4. Experiments

In this section, we report experimental results comparing the uniform, prior-weighted, and Bayesian interpolation methods of the previous section. We exclude the maximum likelihood and maximum a posteriori methods since they are not practical to compute.

4.1. Data and Models

The test set consists of a randomized, anonymized selection of 49,649 transcribed utterances and 319,766 words covering 131 voice tasks from the Android mobile platform. Among the most frequent tasks are SMS, Google Translate, NotePad, Google Maps, Facebook, and Browser.

The acoustic model is a tied-state triphone GMM-based HMM whose input features are 13 PLP-cepstral coefficients, frame-stacked and projected onto 39 dimensions using LDA/STC, trained using ML, MMI, and boosted-MMI objective functions as described in [1].

Separate 4-gram language models were constructed using Katz backoff from each of six sources. The sources include anonymized and randomized voice input text from Android, transcribed and typed search queries, and SMS messages.

These were assembled into a single dynamically-interpolated language model as described in [2] that had 55 million n-grams (1 million unigrams, 27 million bigrams, 21 million 3-grams and 6 million 4-grams). Statically-interpolated language models were also constructed using several of the interpolation methods of Section 3.

4.2. Language Model Interpolation Results

Table 1 shows the recognition word error rate using these models. Uniform and prior-weighted mixture weight interpolation both give 13.7% word error rate in a first pass. Using lattices generated from these first passes and a dynamically-interpolated second pass with task-specific mixture weights, gives 13.0% word error rate with both these interpolation methods. Using Bayesian interpolation instead improves the first pass by 0.4% and the second pass by 0.1%. The gains due to Bayesian interpolation are comparable to those seen by [13].

The dynamically-interpolated LM, which uses task-specific mixture weights, has a word error rate of 12.9% when used in a first pass. This matches the Bayesian statically-interpolated first pass followed by the dynamically-interpolated second pass. In

Interpolation Method	1st-pass WER	2nd-pass WER
uniform	13.7	13.0
prior-weighted	13.7	13.0
bayesian	13.3	12.9
dynamic	12.9	-

Table 1: Word error rate of various interpolation methods in first and second-pass recognition. The first-pass and second pass LMs have 55 million n-grams.

Interpolation Method	1st-pass WER	2nd-pass WER
uniform	14.1	13.0
prior-weighted	14.0	13.0
bayesian	13.6	12.9
dynamic	12.9	-

Table 2: Word error rate of various interpolation methods in first and second-pass recognition. The first-pass LM has 17 million n-grams and second pass LMs has 55 million n-grams.

general, we did not see significant gaps between the rescoring and single-pass dynamic scenarios compared to [2]. This could be due to improvements in the models and data collection or variations in the developement and test sets that have occurred as the Google platform has matured.

4.3. Language-Model Pruning Results

In the above experiments, the statically-interpolated and the dynamically-interpolated language models had the same number of n-grams: 55 million. Given that using the statically-interpolated models in a first pass followed by rescoring with the dynamically-interpolated model did not introduce significant search errors, we chose to increase the 'distance' between the two passes by pruning the statically-interpolated models using entropy pruning [4]. This not only would further test these methods but also allows us to see how small we can make the first pass system, a important memory saving, and still obtain satisfactory results.

Table 2 shows the recognition word error rate using 17 million n-grams for the statically-interpolated LMs and 55 millon for the dynamically-interpolated LM. The second-pass results are identical to using the 55 million n-gram LMs in Table 1. The first pass shows a degradation of 0.3% to 0.4% WER with the Bayesian interpolation still 0.4% better than the other two.

Figure 1 shows first and second-pass results using a range of first-pass Bayesian-interpolated LM sizes. Even with only six million n-grams, an order of magnitude fewer than the second pass, only 0.2% WER is lost compared the full 55 million n-gram first-pass LM.

The conclusions of these experiments is that using the best-performing Bayesian interpolation method, little to nothing is lost in a two-pass strategy when using a range of statically-interpolated LM sizes in the first pass. Further, a simpler statically-interpolated single-pass strategy can be adopted if desired with a 0.4% loss, which is 0.4% better than the other interpolation methods.

5. Acknowledgements

We thank Brandon Ballinger, Ciprian Chelba, Alex Gruenstein and Johan Schalkwyk for discussions and assistance.

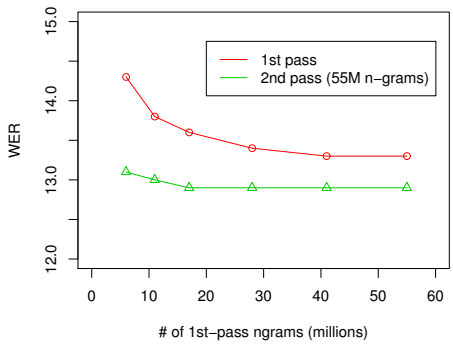

Figure 1: Word error rate with Bayesian interpolation in the first pass with various-sized LMs and dynamic interpolaton in the second pass with 55 million n-gram LMs.

6. References

[1] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, "Google Search by Voice: A case study," in *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*. Springer, 2010.

[2] B. Ballinger, C. Allauzen, A. Gruenstein, and J. Schalkwyk, "On-demand language model interpolation for mobile speech input," in *Proc. of Interspeech*, 2010, pp. 1812–1815.

[3] A. Gruenstein, "Speech input API for android," http://android-developers.blogspot.com/2010/03/speech-input-api-for-android.html, 2010.

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

[5] A. Ljolje, F. Pereira, and M. Riley, "Efficient general lattice generation and rescoring," in *Proc. Europseech*, 1999, pp. 1251–1254.

[6] M. Federico and R. D. Mori, "Language model adaptation," in *Computational Models of Speech Pattern Processing*, K. Ponting, Ed. Berlin: Springer-Verlag, 1999.

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

[8] R. Kneser and V. Steinbiss, "On the dynamic adaptation of stochastic LM," in *Proc. ICASSP*, 1993, p. 586589.

[9] R. Iyer and M. Ostendorf, "Modeling long distance dependence in language: topic mixtures versus dynamic cache models," *IEEE Trans. on Speech and Audio Proc.*, vol. 7, pp. 30–39, 1999.

[10] F. Wessel, A. Baader, and H. Ney, "A comparison of dialogue-state dependent language models," in *Proc. of ESCA Workshop on Interactive Dialogue in Multi-Modal Systems*, 1999, pp. 93–96.

[11] K. Visweswariah and H. Printz, "Language models conditioned on dialogue state," in *Proc. of EUROSPEECH*, 2001, pp. 251–254.

[12] W. Xu and A. Rudnicky, "Language modeling for dialog system," in *Proc. of ICSLP*, 2000, pp. 118–121.

[13] M. Weintraub, Y. Aksu, S. Dharanipragada, S. Khudanpur, H. Ney, J. Prange, A. Stolcke, F. Jelinek, and E. Shriberg, "LM95 project report: Fast training and portability," CLSP, Johns Hopkins University, Baltimore, MD, Tech. Rep., 1995.