GENERALIZED LINEAR INTERPOLATION OF LANGUAGE MODELS

Bo-June (Paul) Hsu

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA 02139, USA
bohsu@mit.edu

ABSTRACT

Despite the prevalent use of model combination techniques to improve speech recognition performance on domains with limited data, little prior research has focused on the choice of the actual interpolation model. For merging language models, the most popular approach has been the simple linear interpolation. In this work, we propose a generalization of linear interpolation that computes context-dependent mixture weights from arbitrary features. Results on a lecture transcription task yield up to a 1.0% absolute improvement in recognition word error rate (WER).

Index Terms— Language modeling, interpolation, adaptation, mixture models

1. INTRODUCTION

With the increasing focus of speech recognition and natural language processing applications on domains with limited amount of in-domain training data, enhanced system performance often relies on approaches involving model combinations. For language modeling, these techniques include improving the estimation of the underlying probability distributions via class n-grams [1] and topic mixtures [2], and adapting to additional training text from the web [3] and the initial recognizer hypotheses [4]. While many of these techniques involve the combination of multiple n-gram language models, most existing works only evaluate their performance using simple linear interpolation [5].

1.1. Existing Techniques

Given a set of M training texts, we can build a combined language model (LM) using multiple techniques. One of the simplest techniques, sometimes referred to as the brute-force approach, is to merge all texts and train a single smoothed n-gram LM. Because the training corpora often differ in size and relevance to the target domain, simply summing the n-gram counts rarely achieves the best result.

Linear interpolation, the most popular model combination technique, first trains individual n-gram LMs from each training corpus. Given the resulting set of n-gram language models, it computes the weighted average of the component model probabilities $p^{LI}(w|h) = \sum_i \lambda_i p_i(w|h)$, where $p_i(w|h)$ is the smoothed probability of word w following n-gram history h in model i. The interpolation weight λ_i, satisfying $\sum_i \lambda_i = 1$, is typically tuned to optimize the development set perplexity.

In practice, such an interpolated model is less efficient for speech recognition than a single backoff n-gram LM, as it requires M probability evaluations for each possible word expansion and the storages of each component LM. Thus, as an approximation, Stolcke [6] constructs a single n-gram backoff model where the probability for all observed n-grams is the weighted average of the component model probabilities. The remaining probabilities are computed via appropriately normalized backoffs. Empirically, as is also observed by Stolcke, the resulting static interpolation generally achieves lower perplexity than the original model. Thus, in subsequent usages, linear interpolation will refer to this static technique.

In [4], Bacchiani et al. suggested count merging as an alternative interpolation technique. Instead of taking the weighted average of the n-gram probabilities, count merging computes the overall probability by scaling the n-gram counts as follows:

$$p^{CM}(w|h) = \frac{\sum_i \beta_i c_i^{disc}(hw)}{\sum_j \beta_j c_j(h)}$$

where β_i is the model scaling factor, $c_i^{disc}(hw)$ is the discounted count that model i assigns to n-gram hw, and $c_i(h)$ is the count of history h in model i. Leveraging the lower order model probabilities for unseen n-grams, we can define the count merging interpolation model with backoff as:

$$p_{bo}^{CM}(w|h) = \begin{cases} p^{CM}(w|h) & \text{if } \sum_i c_i(hw) > 0 \\ \alpha(h) p_{bo}^{CM}(w|h') & \text{otherwise} \end{cases}$$

where $\alpha(h)$ is the backoff weight and h' is the backoff history for history h. Similar to linear interpolation, the scaling factors β_i are tuned against a disjoint development set. As observed in both [4] and section 3.2, count merging generally achieves lower perplexity than linear interpolation.

Previous work has also investigated log-linear interpolation [7] and exponential models [8]. Unlike linear interpola-

tion and count merging, the resulting models from these techniques cannot be efficiently represented as a backoff n-gram model. Thus, they are not suitable for use as a first-pass language model in a speech recognizer. More typically, these language models are used for lattice or n-best rescoring.

1.2. Motivation

By definition, an n-gram LM with discounting assigns probability $p(w|h) = c^{disc}(hw)/c(h)$ to observed n-grams. Refactoring the terms, we see that count merging is simply a generalization of linear interpolation, where the interpolation weight $\lambda_i(h) = \beta_i c_i(h) / \sum_j \beta_j c_j(h)$ now depends on the n-gram history via its count:

$$p^{CM}(w|h) = \frac{\sum_i \beta_i c_i(h) p_i(w|h)}{\sum_j \beta_j c_j(h)} = \sum_i \lambda_i(h) p_i(w|h)$$

Instead of a constant interpolation weight, count merging applies an interpolation weight proportional to the number of observances of n-gram history h. The more data used to train the word distribution following h, the more we trust and weigh the resulting estimate.

Motivated by Witten-Bell smoothing [9], Zhou et al. proposed a linear interpolation model where the interpolation weight is defined as a function of not just $c(h)$, but also the number of unique words that follow h [10]. However, this heuristic-based scheme assumes the existence of an in-domain training set and does not perform parameter optimization to maximize data likelihood. Intuitively, we should be able to obtain better performance by both leveraging additional features and optimizing interpolation model parameters on a development set.

In this work, we will investigate a data-driven approach to learn the interpolation weight function. As an extension to both linear interpolation and count merging, the generalized linear interpolation model significantly reduces the perplexity and WER over both existing techniques.

2. GENERALIZED LINEAR INTERPOLATION

The generalized linear interpolation model extends the constant mixture weights λ_i in linear interpolation to interpolation weight functions $\lambda_i(h)$ over arbitrary n-gram history features. Specifically, given language models $p_i(w|h)$ for $i = 1, ..., M$, we define the generalized linear interpolation model as:

$$p^{GLI}(w|h) = \sum_i \lambda_i(h) p_i(w|h)$$

where $\lambda_i(h) \geq 0$ and $\sum_i \lambda_i(h) = 1$, for all observed history h. Similar to count merging, we can define the corresponding backoff model as:

$$p_{bo}^{GLI}(w|h) = \begin{cases} p^{GLI}(w|h) & \text{if } \sum_i c_i(hw) > 0 \\ \alpha(h) p_{bo}^{GLI}(w|h') & \text{otherwise} \end{cases}$$

where h' is the backoff history for history h and $\alpha(h)$ is the backoff weight computed to satisfy $\sum_w p_{bo}^{GLI}(w|h) = 1$.

In general, we can model the interpolation weight function with both parametric and non-parametric functions. In this work, we will focus on the following log-linear family of parametric functions [1] and leave additional choices of interpolation weight functions for future work:

$$\lambda_i(h) = \frac{\text{rel}_i(h)}{\sum_j \text{rel}_j(h)} \qquad \text{rel}_i(h) = \exp\left(\boldsymbol{f}_i(h) \cdot \boldsymbol{\theta} + \gamma_i\right)$$

This particular parametric family computes the interpolation weights by normalizing the relevance functions $\text{rel}_i(h)$ for each model. It was designed such that the constraints on $\lambda_i(h)$ are always satisfied regardless of the feature vectors $\boldsymbol{f}_i(h)$, feature parameter values $\boldsymbol{\theta}$, and model bias parameter values $\boldsymbol{\gamma}$. Furthermore, an easily computed gradient helps improve the performance of gradient-based optimization techniques when tuning the interpolation parameters. Lastly, we can interpret the learned parameters as adjusting the multiplicative effect a feature has on the component model probabilities. Since a constant offset to $\boldsymbol{\gamma}$ yields identical interpolation weights due to cancellation, we will set $\gamma_M = 1$ to constrain the solution space.

Within this framework, linear interpolation can be represented as a special case where $\boldsymbol{f}_i(h) = [\,]$ is an empty vector, resulting in constant interpolation weights $\lambda_i(h) \propto e^{\gamma_i}$. Count merging can be represented with $\boldsymbol{f}_i(h) = [\log(c_i(h))]$, $\boldsymbol{\theta} = [1]$, and count scaling parameter $\beta_i = e^{\gamma_i}$.

Training the generalized linear interpolation model involves adjusting the model parameters that characterize the interpolation weight functions to minimize the development set perplexity. Since no constraints are placed on the parameters, we can apply any numerical optimization technique to iteratively estimate the optimal parameter values. By choosing the parametric family of the interpolation weight function to include linear interpolation and count merging as special cases, we guarantee that the interpolated model will at least match and almost always exceed the performance of the existing techniques on the development set. Given a sufficiently large development set to avoid overfitting, this gain generally will also carry over to unseen test sets.

2.1. Features

Unlike existing model combination methods, generalized linear interpolation supports arbitrary combination of features in the computation of the interpolation weight function. In order to compare fairly with existing techniques, in this work, we will limit ourselves to features that can be automatically derived from n-gram counts. Specifically, we will examine the use of the n-gram history count $c(h)$, the left branch count, and the right branch count (defined below) to derive the model

[1] We can also interpret this interpolation weight function as a two-layer neural network with a softmax activation function on the output layer [11].

features. We will leave other features such as part-of-speech tags, topic labels, and document counts to future work.

Following the notation in [12], we define the left branch count $c^l(h) = N_{1+}(\bullet h)$ as the number of unique words that appear before h. This count is used in Kneser-Ney smoothing [13] to estimate the probability of lower-order models. Symmetrically, we define the right branch count $c^r(h) = N_{1+}(h\bullet)$, motivated by Witten-Bell smoothing [9], as the number of unique words that appear after h. To simplify notation, we will define $\boldsymbol{C}(h) = \left[c(h), c^l(h), c^r(h)\right]$ and apply operators on the vector element-wise.

The generalized linear interpolation model with the log-linear weight function computes the relevance of a model as the exponential of a weighted sum of the model features. To include count merging as a special case, we will consider the logarithms of the three counts $\log \boldsymbol{C}(h)$ as possible features [2].

Similar to how introducing higher powers of the original features improves the fit for polynomial regression, we can include functions of the original counts as additional features to the interpolation model to obtain a better fit. In this work, we will limit our study to the squares of the log counts from above. To obtain a monotonically increasing function, we will add 1 to the counts and consider $\mathrm{logSq}(\boldsymbol{C}(\mathrm{h})) = [\log(1 + \boldsymbol{C}(\mathrm{h}))]^2$ as possible model features.

3. EXPERIMENTS

3.1. Setup

In this work, we will compare the performance of the generalized linear interpolation model against a few existing interpolation techniques by evaluating the perplexity and recognizer WER in a lecture transcription domain [14]. The target domain consists of 20 lectures from an introductory computer science course, from which we withheld the first 10 lectures for the development set (CS Dev) and used the last 10 for the test set (CS Test). For training, we used high-fidelity transcripts from approximately 115 hours of audio from 99 lectures on a variety of topics (Lectures). To supplement the off-topic lecture transcripts with topic-specific resources, we included the course textbook (Textbook) as additional training data. Finally, to evaluate the effect of using a large, out-of-domain dataset on the overall performance, we added the transcripts from the LDC Switchboard corpus of spontaneous conversational speech [15] (Switchboard) as a third training set. Table 1 summarizes all the evaluation data.

For each training text, we built a trigram language model with modified Kneser-Ney smoothing [12] and the default corpus-specific vocabulary using SRILM [6]. As models interpolated over the same components share a common vocabulary regardless of the interpolation technique, we can compare the perplexities computed only over n-grams with non-zero probabilities for each technique.

[2] Zero counts result in $\mathrm{rel}_i(h) = \exp(\log 0 + \cdots) = \exp(-\infty) = 0$.

Dataset	Sentences	Vocabulary	Words
Textbook	6,762	4,686	131,280
Lectures	58,626	26,906	1,113,312
Switchboard	262,744	26,837	3,129,827
CS Dev	4,102	3,289	93,353
CS Test	3,595	3,357	87,527

Table 1. Summary of evaluation datasets.

In all experiments, the interpolation model parameters are initialized to 0 and tuned to minimize the development set perplexity using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) unconstrained optimization technique [16], a quasi-Newton method that uses the second derivative Hessian matrix iteratively estimated from the gradients evaluated along the search path to improve the convergence towards the function minimum.

To compute the word error rates associated with each language model, we used a speaker-independent speech recognizer [17]. The evaluation lectures were pre-segmented into utterances via forced alignment against the reference transcription [18]. Since the interpolated language models can be encoded as n-gram backoff models, they are applied directly in the first recognition pass instead of a separate n-best rescoring step. Table 2 summarizes the performance of various interpolation models on the test set.

3.2. Results

For Textbook and Lectures, all model combination techniques achieve significantly reduced perplexity (though not strictly comparable) and WER over the individual component models. Compared with the linear interpolation (LI) baseline, the brute-force (BF) text concatenation method resulted in worse perplexity and no statistically significant reduction in WER. Validating the observations in [4], count merging (CM) outperforms linear interpolation, with $p < 0.001$ on the Matched Pairs Sentence Segment Word Error significance test [19].

With $\log c(h)$ as the only feature, generalized linear interpolation (GLI) extends count merging by allowing an arbitrary exponent on the count features. Intuitively, given sufficient n-gram histories to yield good estimates of the word distribution, the count should no longer play a significant role in determining the interpolation weight. Thus, we expect the optimal exponent on the count to be less than the fixed value of 1.0 for count merging. Empirically, we find this to be the case, with the optimal exponent for the n-gram history count to be around 0.8.

By tuning the exponent, we obtain a slight, but insignificant, reduction in both perplexity and WER. However, with the addition of the left and right branch counts, GLI with $\log \boldsymbol{C}(h)$ achieves a significant WER improvement over count merging, with $p = 0.043$. Including second-order features $\mathrm{logSq}(\boldsymbol{C}(\mathrm{h}))$ further drops the perplexity by another 1.5%

Model	Perplexity	WER
Textbook	332.6	47.4%
Lectures	225.2	41.6%
Switchboard (Swbd)	287.5	45.8%
Textbook + Lectures		
LI	165.4	37.5%
BF	170.6 (+3.1%)	37.3%
CM	158.1 (-4.4%)	*36.8%*
GLI: $\log c(h)$	157.9 (-4.6%)	*36.8%*
GLI: $\log \boldsymbol{C}(h)$	157.3 (-4.9%)	*36.7%*
GLI: $\log \boldsymbol{C}(h), \text{logSq}(\boldsymbol{C}(\text{h}))$	154.8 (-6.4%)	*36.6%*
Textbook + Switchboard		
LI	178.2	39.5%
CM	175.2 (-1.7%)	*38.7%*
GLI: $\log \boldsymbol{C}(h), \text{logSq}(\boldsymbol{C}(\text{h}))$	170.1 (-4.5%)	*38.5%*
Lectures + Switchboard		
LI	224.6	41.2%
CM	227.1 (+1.1%)	41.2%
GLI: $\log \boldsymbol{C}(h), \text{logSq}(\boldsymbol{C}(\text{h}))$	224.6 (-0.1%)	*41.1%*
Textbook + Lectures + Swbd		
LI	161.1	37.3%
CM	155.0 (-3.8%)	*36.6%*
GLI: $\log \boldsymbol{C}(h), \text{logSq}(\boldsymbol{C}(\text{h}))$	150.7 (-6.5%)	*36.3%*

Table 2. Interpolation model performance on the test set. Relative perplexity changes from the linear interpolation baseline are included within parentheses. Statistically significant improvements in WER are in italics.

and improves the WER significance to $p = 0.001$. Overall, the generalized linear interpolation model achieves a statistically significant 0.2% and 0.9% absolute WER reduction over count merging and the baseline linear interpolation techniques, respectively.

Interpolating different combinations of training models generally yields similar trends, with the generalized linear interpolation using second-order features significantly outperforming count merging and standard linear interpolation, by up to 1.0% absolute reduction in WER. When combining Lectures and Switchboard, however, any improvement over linear interpolation is generally minimal, with only the full-featured GLI model able to obtain a statistically significant drop, with $p = 0.019$. Given that linear interpolation only reduced WER by 0.4% over the Lectures model, Switchboard's mismatch in both topic and style appears to leave little room for additional improvement over linear interpolation.

To obtain a better sense of how the generalized linear interpolation model performs under different training conditions, we measure the sensitivity of the interpolated model perplexity to the development set size. In Figure 1, we plot the perplexity of the full-featured generalized linear interpolation model for Textbook, Lectures, and Switchboard, with respect to the size of the development set. As shown, with

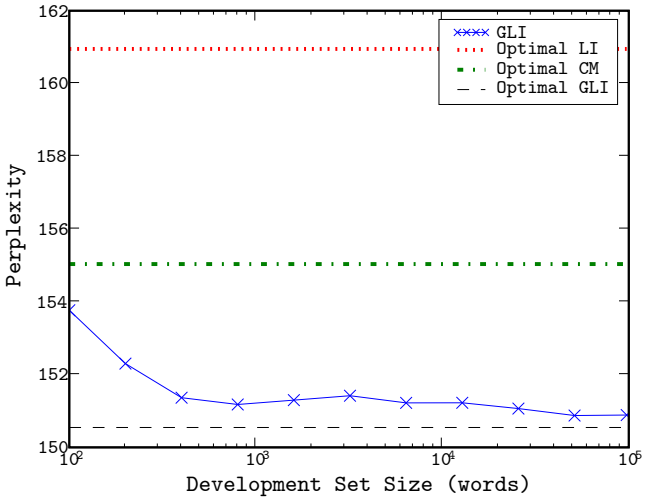

Fig. 1. Test set perplexity of the GLI model for Textbook + Lectures + Swbd optimized with varying development set sizes. Optimal LI/CM/GLI denote the test set perplexities obtained by tuning the respective models on the entire test set.

only 400 words, the perplexity of this 8-parameter model has converged to within 1 point of the optimal value obtained by directly training on the test set. Similar, if not better, behavior is observed with all combinations of the training data. This suggests that the performance gains from the generalized linear interpolation model can be obtained with relatively little in-domain data.

3.3. Implementation

Similar to linear interpolation and count merging, we can represent a generalized linear interpolation model in the ARPA LM format [20] consisting of only the observed n-grams across the component models. By representing LMs as vectors of probabilities and backoff weights, and pre-computing their contributions towards the development set perplexity, we can efficiently interpolate and evaluate each set of model interpolation parameters via simple vector arithmetics (additional details are forthcoming). In the above experiments, each evaluation step in the iterative optimization process takes less than a second to complete. Depending on the number of parameters, the entire parameter estimation procedure takes from a few seconds to a few minutes to converge. Thus, such a data-driven approach to language model interpolation presents a practical solution to improving model performance.

4. CONCLUSION & FUTURE WORK

In this work, we presented a model-based approach to language model combination. The resulting generalized linear interpolation model defines a family of interpolation weight functions that subsumes both linear interpolation and count

merging. Perplexity and WER evaluations on a lecture transcription domain with various combinations of training data demonstrated up to a 1.0% WER drop over the baseline linear interpolation model. In most cases, the performance also significantly improved over count merging. Detailed analysis showed that the interpolation model can be tuned to within 1 point of the optimal perplexity with only 400 words of development set data.

Despite the promising results achieved so far, an oracle experiment on the test set, where we allow an independent interpolation weight for each n-gram history, yields a lower bound perplexity of 132.2 on the combination of Textbook and Lectures, another 14% below the best perplexity achieved so far. Thus, in future work, we plan on studying the effectiveness of additional features and different families of interpolation weight functions, including non-parametric approaches and multilayer feed-forward networks. We would also like to apply the interpolation model to component LMs, such as topic models, that are not based on counts.

As effective language modeling techniques for new domains with limited data increasingly rely on model combination approaches, the traditional linear interpolation and count merging techniques are no longer sufficient to capture and combine the essence of the constituent models. This work presents the generalized linear interpolation model as a step towards a principled study of model combination techniques and encourages future research to consider more expressive approaches to model combination.

5. ACKNOWLEDGMENTS

We would like to thank Mike Phillips for the opportunity to work on this project at vlingo over the summer, Jim Glass, Igor Malioutov, and Chao Wang for the helpful discussions, and the anonymous reviewers for their constructive feedback.

6. REFERENCES

[1] G. Maltese, P. Bravetti, H. Crépy, B.J. Grainger, M. Herzog, and F. Palou, "Combining word- and class-based language models: A comparative study in several languages using automatic and manual word-clustering techniques," in *Proc. Eurospeech*, 2001.

[2] B. Hsu and J. Glass, "Style & topic language model adaptation using HMM-LDA," in *Proc. EMNLP*, 2006.

[3] I. Bulyko, M. Ostendorf, and A. Stolcke, "Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures," in *Proc. HLT*, 2003.

[4] M. Bacchiani, M. Riley, B. Roark, and R. Sproat, "MAP adaptation of stochastic grammars," *Computer Speech & Language*, vol. 20, no. 1, pp. 41–68, 2006.

[5] F. Jelinek and R.L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," in *Proc. Workshop on Pattern Recognition in Practice*, 1980.

[6] A. Stolcke, "SRILM – An extensible language modeling toolkit," in *Proc. ICSLP*, 2002.

[7] D. Klakow, "Log-linear interpolation of language models," in *Proc. ICSLP*, 1998.

[8] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Computer Speech & Language*, vol. 10, no. 3, pp. 187–228, 1996.

[9] I.H. Witten and T.C. Bell, "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression," *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1085–1094, 1991.

[10] Z.Y. Zhou, J.F. Gao, and E. Chang, "Improving language modeling by combining heteogeneous corpora," in *Proc. ISCSLP*, 2002.

[11] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, November 1995.

[12] S. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Technical Report TR-10-98*. Computer Science Group, Harvard University, 1998.

[13] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proc. ICASSP*, 1995.

[14] J. Glass, T. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay, "Recent progress in the MIT spoken lecture processing project," in *Proc. Interspeech*, 2007.

[15] J. Godfrey and E. Holliman, "Switchboard-1 transcripts," Linguistic Data Consortium, Philadelphia, 1993.

[16] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes*, Cambridge University Press, 3rd edition, September 2007.

[17] J. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech & Language*, vol. 17, no. 2-3, pp. 137–152, 2003.

[18] T.J. Hazen, "Automatic alignment and error correction of human generated transcripts for long speech recordings," in *Proc. Interspeech*, 2006.

[19] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. ICASSP*, 1989.

[20] A. Stolcke, "SRILM man pages: ngram-format," 2004, `http://www.speech.sri.com/projects/srilm/manpages/ngram-format.html`.