JOINT TRAINING OF INTERPOLATED EXPONENTIAL N-GRAM MODELS

Abhinav Sethy[1], *Stanley Chen*[1], *Ebru Arisoy*[1], *Bhuvana Ramabhadran*[1]
Kartik Audkhasi[2], Shrikanth Narayanan[2], Paul Vozila[3]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
[2] University of Southern California, Los Angeles, CA, USA
[3] Nuance Communications, Burlington, MA, USA

ABSTRACT

For many speech recognition tasks, the best language model performance is achieved by collecting text from multiple sources or *domains*, and interpolating language models built separately on each individual corpus. When multiple corpora are available, it has also been shown that when using a domain adaptation technique such as *feature augmentation* [1], the performance on each individual domain can be improved by training a joint model across *all* of the corpora. In this paper, we explore whether improving each domain model via joint training also improves performance when interpolating the models together. We show that the *diversity* of the individual models is an important consideration, and propose a method for adjusting diversity to optimize overall performance. We present results using word n-gram models and Model M, a class-based n-gram model, and demonstrate improvements in both perplexity and word-error rate relative to state-of-the-art results on a Broadcast News transcription task.

1. INTRODUCTION

Language models are critical components in various natural language processing applications such as machine translation and speech recognition, and n-gram language models continue to be the dominant technology in state-of-the-art systems. For many tasks, training text is available from a collection of diverse data sources or *domains*, and the best performance can be achieved by combining information from all of these sources. While one can simply train a single model on the pooled data, better performance can often be achieved by building a separate language model on each individual corpus and linearly interpolating the component models. Model combination (in contrast to *data* combination) has the added advantage that the final model can be inexpensively adapted to a specific domain by adjusting a few interpolation weights, rather than requiring retraining of the models themselves.

In this paper, we investigate whether *domain adaptation* can be used to improve the performance of an interpolated model by improving the quality of each component model. In domain adaptation, one attempts to improve performance on an *in-domain* test set by supplementing an in-domain training set with a (typically larger) *out-of-domain* training set. With training sets from multiple domains as in interpolated models, each domain model can be improved by viewing the data from the remaining domains as out-of-domain data, and applying a domain adaptation technique.

Here, we focus on the *feature augmentation* algorithm developed for domain adaptation for general classifiers, also known as "frustratingly easy" domain adaptation [1, 2]. The core idea is to construct a regularized model containing multiple copies of each *feature*, one copy for the global model, and one copy for each domain model. Global features capture statistics shared across all training corpora, while domain-specific features represent knowledge particular to each data source. Features are jointly trained across all of the data; the global features enable the sharing of information between subcorpora, potentially leading to better parameter estimates for each domain. To apply this model to a particular domain, only the global features and features specific to that domain are included. Empirically, this method generally leads to better performance on each component domain as compared to building a model for each domain solely from the corresponding subcorpus.

We explore whether applying feature augmentation to each component model in an interpolated model can improve the performance of the combined model. We find that a gain can in fact be achieved, but small details can have a significant effect on performance. In particular, the performance of an interpolated model tends to improve the more *diverse* its component models are, just as with other ensemble or system combination techniques [3]. However, joint training tends to make the domain-specific models *less* diverse, because of the sharing of global features. We show that good performance is contingent on finding the right level of diversity, and propose a scheme using log-linear interpolation to vary the amount of diversity present. We contrast this with the method of varying regularization hyperparameters from [2]. In addition, we show how to tune the level of diversity to optimize overall performance.

While word n-gram models are usually represented as back-off language models, they may also be represented using exponential models, and this representation is more well-suited to the use of feature augmentation. Furthermore, exponential models offer more flexibility in terms of the types of features that can be supported (also taking regularization into account), in contrast to the highly specialized smoothing techniques used in conventional n-gram models. As we utilize this flexibility when developing schemes for controlling model diversity, we consider only exponential language models in this work. In addition to word n-gram models, we also evaluate Model M, an exponential class-based language model that has achieved superior performance across many domains [4].

The next section discusses related work on language model adaptation. Section 3 describes exponential language models and feature augmentation in more detail. In Sections 4 and 5, we present our recipe for constructing interpolations of jointly trained models, as well as our method for controlling diversity. In Section 6, we provide experimental results on Broadcast News data and conclusions are presented in Section 7.

2. RELATED WORK

The two fundamental issues underlying this work are: what is the best way to combine information from training sets from multiple sources; and what is the best way to use out-of-domain data to improve an in-domain model, *i.e.*, domain adaptation. Both of these issues fall under the umbrella of *language model adaptation*, which deals with the general task of handling training data not matched to test data. An extensive survey of the field can be found in [5].

The most widely-used method for combining multiple data sources is to use linear interpolation to combine separate models built from each source, due to its good performance over a wide range of situations and its ease of implementation. Typically, interpolation weights are static, but dynamically varying interpolation weights according to the language model history can sometimes produce gains [6, 7]. Other variations of linear interpolation include maximum *a posteriori* (MAP) and Bayesian interpolation [8], and log-linear interpolation has also been shown effective [9]. While model combination methods like interpolation are most popular, data pooling methods such as *count merging* can also work well. However, these approaches are not robust to the case where much more data is available from some sources as compared to others.

Domain adaptation can be considered as a special case of combining multiple data sources, and thus all of the previously mentioned techniques can be applied. In addition, there are methods that are tailored to the situation where there is a clear "in-domain" corpus and "out-of-domain" corpus. Minimum Discrimination Information (MDI) and regularized MDI [10, 11] and MAP adaptation [12] use the out-of-domain model as a prior when training the in-domain model. These techniques work well for domain adaptation, though regularization hyperparameters may need to be chosen carefully. The MDI and MAP techniques are very similar to feature augmentation [1, 2] in terms of the features retained in the models. All of these methods include both global and domain-specific copies of features. The main difference is in how training is performed; in the former methods, the out-of-domain model is trained first and fixed as a prior when training the in-domain model. In feature augmentation, the global and domain-specific models are trained jointly. The use of domain-specific features in exponential n-gram models has also been explored in [13, 14]. Unlike in our work, domain features are created only for n-grams that are correlated (or anti-correlated) with a topic, rather than for most or all of the n-grams that occur.

In this work, we use domain adaptation to improve component models when using linear interpolation for model combination. A similar idea was evaluated on a small Broadcast News task in [10], except using rMDI instead of feature augmentation for domain adaptation. Linearly interpolating each domain model with a global model (*i.e.*, a model built on all of the data pooled together) is another way to improve each domain model. This is equivalent to the well-known technique of adding the global model into the overall interpolation. In [10], linear interpolation with the global model was slightly better than the baseline linear interpolation, and the use of rMDI did not appear to add any further improvement.

3. BACKGROUND

In this section, we cover exponential language models and feature augmentation in more detail.

3.1. Exponential language models

We briefly review exponential language models including Model M. For a set of *target* symbols $y \in Y$ and *history* symbols $x \in X$, an exponential model with parameters $\Lambda = \{\lambda_i\}$ and corresponding features $f_i(x,y), \ldots, f_F(x,y)$ has the form

$$P_\Lambda(y|x) = \frac{\exp(\sum_{i=1}^{F} \lambda_i f_i(x,y))}{Z(x)} \tag{1}$$

$$Z(x) = \sum_{y \in Y} \exp(\sum_{i=1}^{F} \lambda_i f_i(x,y)) \tag{2}$$

In language models, the target y is typically the current word w_j and the history x is some number of previous words $w_{j-n+1} \cdots w_{j-1}$.

In an *exponential word n-gram model* for $n = 3$, say, we have binary features $f_{(\mathbf{x},\mathbf{y})}(\cdot)$ for $(\mathbf{x},\mathbf{y})$ of the forms

$$(\epsilon, w_j), (w_{j-1}, w_j), (w_{j-2}w_{j-1}, w_j) \tag{3}$$

where $f_{(\mathbf{x},\mathbf{y})}(x,y) = 1$ iff the history x ends in $\mathbf{x}$ and the target word y is $\mathbf{y}$. Although the number of possible features for an n-gram model is V^n where V is the size of the word vocabulary, we only consider features that correspond to n-grams that occur at least once in the training corpus.

We train exponential n-gram models using a combination of ℓ_1 and ℓ_2^2 regularization [15]; *i.e.*, parameters λ_i are chosen to optimize

$$\mathcal{O}_{\ell_1+\ell_2^2}(\Lambda) = \log \text{PP}_{\text{train}} + \frac{\alpha}{D}\sum_i |\lambda_i| + \frac{1}{2\sigma^2 D}\sum_i \lambda_i^2 \quad (4)$$

for some α and σ, where PP_{train} is the training set perplexity and D is the size of the training set in words [10].

Model M is a class-based n-gram model composed of two separate exponential models, one for predicting classes and one for predicting words. Let $P_{\text{ng}}(y|\lambda)$ denote an exponential n-gram model and let $P_{\text{ng}}(y|\lambda_1,\lambda_2)$ denote a model containing all features in $P_{\text{ng}}(y|\lambda_1)$ and $P_{\text{ng}}(y|\lambda_2)$. If we assume that every word w is mapped to a single word class, the trigram version of Model M is defined as

$$\begin{aligned} P_M(w_j|w_{j-2}w_{j-1}) \equiv\, & P_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \times \\ & P_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \quad (5)$$

where c_j is the word class of word w_j. Model M has achieved among the largest word-error rate improvements over word n-gram models ever reported, with gains as high as 3% absolute as compared to a Katz-smoothed trigram model [4].

3.2. Domain Adaptation by Feature Augmentation

Feature augmentation [1, 2] can be viewed as taking a global feature set derived from the entirety of the training data, and including an additional copy of these features for each domain-specific data set. The global features are "active" for all training events, while domain-specific features are active only for the corresponding domain data (and otherwise pinned to zero). In other words, if there are F global features $\{f_i(x,y)\}$ and K domain corpora, there will be $F \times (K+1)$ features in the augmented model. We can view the first F features as the global copy and each succeeding set of F features the copy for each domain. If the vector $\vec{\phi}$ (of size F) represents the global feature values $\{f_i(x,y)\}$ for some training event (x,y), then the augmented feature values will be

$$\vec{\phi}_{\text{aug}} = (\vec{\phi}; \vec{\phi}, \vec{0}, \vec{0}, \ldots, \vec{0})$$

if the event belongs to the first domain corpus;

$$\vec{\phi}_{\text{aug}} = (\vec{\phi}; \vec{0}, \vec{\phi}, \vec{0}, \ldots, \vec{0})$$

if the event belongs to the second domain corpus; and so forth. For each event, the only parts of the augmented vector that are nonzero are the global copy and the copy corresponding to the corpus from which the event is drawn.

Using global features enables the model to share information across domains, by encoding domain-independent statistics in the parameters of the global features and domain-specific statistics in the parameters of the domain features. In particular, it has been shown that the sum of the parameter mass $\sum_i |\lambda_i|$ in a model is a very good predictor of training set overfitting (in log perplexity space), and global features reduce overall parameter mass by absorbing parameter mass that is shared across domains [4].

Note that although the total number of features is $F \times (K+1)$, many features may never occur with nonzero values in many of the domain corpora, and the associated feature copies can be ignored. Consequently, the effective dimensionality of the augmented vector can be significantly smaller than the upper bound of $F \times (K+1)$, as will be discussed further in following sections.

4. JOINT TRAINING WITH FEATURE AUGMENTATION

To apply the feature augmentation framework described in Section 3.2 to exponential n-gram language models, we supplement the (global) feature templates given in Equation 3 (for $n = 3$) with domain-specific features of the form

$$(k, w_j), (w_{j-1}k, w_j), (w_{j-2}w_{j-1}k, w_j) \quad (6)$$

where $k \in \{1,\ldots,K\}$ identifies which domain the feature is associated with. Then, the parameters $\{\lambda_i\}$ attached to each feature can be trained to optimize Equation 4 as before.

The jointly trained model can be used as a model for domain k by considering only the global features along with the features associated with domain k; we denote the model for domain k as $P_k(y|x)$. We refer to the K domain models as *component models*. Linear interpolation of the component models can be written as $\sum_k w_k P_k(y|x)$ where the weights w_k are selected from the unit simplex. In common practice, the weights w_k are estimated using the expectation-maximization (EM) algorithm to maximize the likelihood of a held-out set.

As noted in [2], the value of the regularization hyperparameters (α, σ^2) used in Equation 4 can have a large impact on performance. Rather than using a single global value for (α, σ^2), it is beneficial to use at least two different values, one for the global features and one for the domain-specific features. A high regularization penalty for the global features encourage the domain-specific models to behave like models trained independently on each domain training set (*i.e.*, to be more diverse), while a high penalty on the domain-specific models encourage them to approach a common global model (*i.e.*, to be less sparsely estimated).

Ultimately, we would like to select global and domain-specific regularization hyperparameters to optimize the performance of the interpolated model, *e.g.*, in terms of perplexity on a held-out set. However, for each hyperparameter setting evaluated, the parameters $\{\lambda_i\}$ need to be retrained which is quite expensive, making hyperparameter search difficult. Also, note that the hyperparameter settings that optimize interpolated model performance may be unrelated to the settings that optimize the performance of each domain model individually. In the next section, we propose a simple alternative to hyperparameter optimization.

5. COMBINING MODEL AND PARAMETER SPACE INTERPOLATION

The performance of a linearly interpolated model depends both on the "strength" of the individual models (*e.g.*, how sparsely estimated they are) as well as on their diversity. We now develop a new recipe for interpolation which trades off diversity and model strength to achieve better performance.

First, let us rewrite our component models $P_k(y|x)$ as $P_k(y|x; \Lambda_g, \Lambda_k)$, indicating that the parameters for the global features Λ_g and the parameters for the domain-specific features Λ_k are taken "as is" from the jointly trained model. Then, for each domain k, we associate a scaling factor s_k^g with the global parameters and a scaling factor s_k with the domain-specific parameters; *i.e.*, the λ_i parameters are scaled by this factor in the component models. (The original $P_k(y|x)$ correspond to $s_k^g = s_k = 1$.) These scaling factors can be used as a knob to tradeoff the diversity that the domain-specific features bring against the more robust estimates from the global features. Our proposed interpolation method can then be written as

$$P(y|x) = \sum_k w_k P_k(y|x; s_k^g \Lambda_g, s_k \Lambda_k)$$

This can be seen as a combination of model space interpolation (across components) and parameter space interpolation (between global and domain-specific λ_i's). For each domain k, we estimate three parameters: its interpolation weight w_k; a scale factor for global features s_k^g; and a scale factor for the domain-specific features s_k. For Model M, we use separate scale factors for the class and word models, leading to 4 parameters per domain.

These parameters can be trained to optimize held-out set likelihood using the generalized EM algorithm [16], alternating between assigning fractional counts to each domain for each event; reestimating the mixture weights w_k; and reestimating the scaling parameters s_k^g and s_k. Estimating the scale parameters can be framed as training an exponential model with one feature per scaling parameter. Referring to Equation 1, the feature corresponding to s_k^g has the value $\sum_i \lambda_{g,i} f_i(x, y)$ and the feature corresponding to s_k has the value $\sum_i \lambda_{k,i} f_i(x, y)$. We use conjugate gradient descent for parameter optimization, using the n-gram sorting process described in [17] to efficiently compute the required gradients. Note that the λ_i's and regularization hyperparameters are fixed during this process, simplifying training.

	PP	WER
data pooling	121	13.2%
baseline	116	13.0%
adapt	109	12.8%
adapt+scale	104	12.7%

Table 1. Perplexities and WERs for various domain combination methods for word n-gram models.

6. RESULTS

We present results on an English Broadcast News task. The language model training text for our experiments consists of a total of 300M words from the following four data sources: GALE Phase 2 Distillation GNG Evaluation Supplemental Multilingual data (2007EN), EARS BN03 closed captions (BN03), 1996 CSR Hub 4 Language Model data (98T31), and Hub 4 acoustic model training transcripts (Hub4). A vocabulary of 84k words is used. We compute word-error rates (WERs) using lattice rescoring on a 2.5h *rt04* evaluation set containing 45k words. The held-out set consists of 45k words of *dev04* data. Building a separate unpruned modified Kneser-Ney-smoothed language model on each subcorpus and interpolating produces a WER of 13.0%. We primarily report results for exponential word n-gram models, including analysis of how regularization hyperparameters and scaling factors affect diversity; and how diversity affects overall performance. A smaller set of results is provided for Model M in Section 6.2.

6.1. Word n-gram models

We first compare the perplexities (PPs) and WERs of various domain combination methods in Table 1. The first line corresponds to building a single exponential n-gram model on the pooled data; and *baseline* refers to linearly interpolating separate models built on each subcorpus. All models are 4-gram models, and the regularization hyperparameters (α, σ^2) are set to (0.5, 6) unless otherwise noted, as these values have been found to work well over a wide range of tasks and model types. On this data set, we see that the baseline linear interpolation does indeed outperform data pooling.

Next, we report the results from using joint training to adapt the component models in the linear interpolation using the default regularization hyperparameters (*adapt*). Finally, we report results after applying optimized scale factors s_k^g and s_k to the model parameters (*adapt+scale*). We find that using feature augmentation to improve each component model does in fact improve overall performance, reducing WER by 0.2% absolute over the baseline. Applying parameter scaling improves performance even more, giving us a 0.3% absolute reduction in WER and 12% reduction in perplexity compared to the baseline. The WER reduction is statistically significant at a p-level less than 0.005 under the Matched Pair Sentence

	(α, σ^2)			
	(0.5, 6)	(0.3, 4.5)	(1, 3)	(500, 0.001)
2007EN	161	168	145	121
BN03	120	123	117	118
98T31	146	152	134	121
Hub4	131	138	119	118
adapt	109	109	110	117

Table 2. Effect of regularization hyperparameters on perplexities for component models and overall model after interpolation (*adapt*).

	(α, σ^2)			
	(0.5, 6)	(0.3, 4.5)	(1, 3)	(500, 0.001)
2007EN	0.029	0.011	0.038	0.002
BN03	0.150	0.080	0.178	0.006
98T31	0.045	0.029	0.055	0.005
Hub4	0.050	0.036	0.054	0.004

Table 3. Effect of regularization hyperparameters on diversity between models as measured by Equation 7.

Segment test.

Next, we examine the impact of varying regularization hyperparameters on the component models derived from joint training. We keep (α, σ^2) fixed at their default values for the global features and only vary the regularization penalty for domain-specific features. Perplexities for each component model and for the associated interpolated model are given in Table 2. The lower the α and the higher the σ^2, the lower the overall regularization penalty (see Equation 4).

As discussed earlier, high regularization penalties should cause the component models to approach the pooled model. This corresponds to the last column in the table, and accordingly we find that all of the component models have a perplexity similar to that of the pooled model (121), and interpolation improves performance only slightly. On the other hand, lower regularization penalties (as in the other columns) should encourage diversity, and the interpolated performance is significantly better even though the perplexities of the component models are generally worse. This demonstrates the importance of model diversity for overall performance.

In order to quantify the effect of regularization on model diversity, we measure the average empirical Kullback-Leibler (KL) distance between each component model and the remaining models on the held-out set. That is, for each component model $P_k(y|x)$ we compute

$$\text{KL}_k = -\frac{1}{D_h}\sum_{d=1}^{D_h}\sum_{k'=1}^{K} P_k(y_d|x_d)\ln\frac{P_k(y_d|x_d)}{P'_k(y_d|x_d)} \qquad (7)$$

where the held-out set is denoted as $(x_1, y_1), \ldots, (x_{D_h}, y_{D_h})$. As can be seen from Table 3, there is a clear decrease in the diversity between models as the regularization penalty on domain-specific features increase.

	separate opt.			joint opt.		
	s_k^g	s_k	div	s_k^g	s_k	div
2007EN	1.02	-0.10	0.0001	1.06	0.48	0.10
BN03	0.98	0.55	0.02	0.55	0.98	0.14
98T31	1.02	0.001	0.001	0.99	0.50	0.04
Hub4	1.01	0.34	0.001	0.92	1.45	0.016
PP	115			104		

Table 4. Average KL diversity (div) with global (s_k^g) and domain-specific (s_k) scale factors optimized independently or jointly.

	total	nonzero
data pooling	275M	
baseline	310M	287M
adapt	587M	513M

Table 5. Comparison of total number of parameters and number of nonzero parameters for various methods.

In Table 4, we demonstrate the importance of estimating the scaling factors s_k^g and s_k jointly. The left half of the table corresponds to the case where s_k^g and s_k are estimated separately for each domain model to optimize the perplexity of the held-out set, while the right half of the table corresponds to joint estimation as described in Section 5. We see that joint estimation leads to higher weights for the domain-specific models and to higher overall diversity, producing a better tradeoff between diversity and individual component model performance, and producing a better perplexity overall as compared to separate estimation or not using scaling at all.

Next, we compare the total number of parameters for different methods in Table 5. Data pooling results in the least number of parameters, as there is only a single copy of each feature. In the baseline method where there is a separate model for each domain, the same feature can occur in multiple domain models. Finally, joint training results in the most features, as there are both global features as in data pooling and domain-specific features as in the baseline. While the joint model can potentially have as many as $K + 1 = 5$ times as many features as compared to data pooling, we find that in this case the actual factor is close to two. This is expected as higher-order n-gram features tend to be sparse, and will usually occur in only one domain corpus. We also note that about 13% of the features in the joint model have $\lambda_i = 0$ and can be discarded, which is larger than the corresponding figure of 7% for the baseline. This can be attributed to feature overlap while training under regularization.

6.2. Model M

Results with Model M are presented in Table 6. Since Model M is a smoother model [4], it is expected that the benefit from

	PP	WER
baseline	108	12.4%
adapt+scale	103	12.2%

Table 6. Perplexities and WERs for various domain combination methods for Model M.

adaptation will be smaller. Accordingly, we get a 4% relative reduction in perplexity and 0.2% absolute reduction in WER as compared to the baseline WER of 12.4%, corresponding to a linear interpolation of independently-estimated domain-specific Model M's. The WER reduction is statistically significant at a p-level of 0.07 under the Matched Pair Sentence Segment test. Note that the best Model M perplexity of 103 is comparable to the best perplexity achieved with word n-gram models (104), but the WER is significantly better (12.2% *vs.* 12.7%).

7. CONCLUSION

In this paper, we show that the performance of an interpolated model can be improved by applying domain adaptation to improve its component models. We show that feature augmentation can be used to better state-of-the-art performance on a large-scale task by up to 0.3% absolute. We find that diversity is an important consideration when designing interpolated models, and show how diversity can be varied via regularization hyperparameters. However, searching for optimal hyperparameters is extremely expensive; evaluating just a single set of hyperparameters (corresponding to a single column in Table 2) took around 15 hours. Consequently, we propose a novel method for controlling diversity through parameter scaling and give an efficient method for training scaling factors. For this data, the entire scaling factor optimization required about an hour of computation.

Joint training results in a model about twice the size of the baseline. However, if log-linear interpolation is used to combine component models rather than linear interpolation, then this disparity disappears as multiple copies of the same feature can be merged into one. In addition, log-linear interpolation leads to lower run-time computation since interpolated parameters can be precomputed statically. Log-linear interpolation typically gives similar performance as compared to linear interpolation, and we plan to explore this avenue in future work. This paper was focused on exponential $n-$gram models. We plan to explore whether the techniques presented in this paper apply to other class of models including neural net language models.

8. REFERENCES

[1] Hal Daumé and D Marcu, "Frustratingly easy domain adaptation," in *Annual meeting-association for computational linguistics*, 2007, vol. 45, p. 256.

[2] Jenny Rose Finkel and Christopher D Manning, "Hierarchical bayesian domain adaptation," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 602–610.

[3] Ludmila I. Kuncheva and Christopher J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, May 2003.

[4] Stanley F. Chen, "Shrinking exponential language models," in *Proceedings of NAACL-HLT*, 2009.

[5] Jerome R Bellegarda, "Statistical language model adaptation: review and perspectives," *Speech communication*, vol. 42, no. 1, pp. 93–108, 2004.

[6] Bo-June Hsu, "Generalized linear interpolation of language models," in *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, 2007, pp. 136–140.

[7] X Liu, MJF Gales, and PC Woodland, "Context dependent language model adaptation," in *Proceedings of the 8th International Conference on Speech Communication and Technology (INTERSPEECH08)*, 2008, pp. 837–840.

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

[9] Dietrich Klakow, "Log-linear interpolation of language models," in *Proc. ICSLP*, 1998, vol. 5, pp. 1695–1698.

[10] Stanley F. Chen, Lidia Mangu, Bhuvana Ramabhadran, Ruhi Sarikaya, and Abhinav Sethy, "Scaling shrinkage-based language models," Tech. Rep. RC 24970, IBM Research Division, April 2010.

[11] Adam Berger and Robert Miller, "Just-in-time language modelling," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. IEEE, 1998, vol. 2, pp. 705–708.

[12] Ciprian Chelba and Alex Acero, "Adaptation of maximum entropy capitalizer: Little data can help a lot," *Computer Speech & Language*, vol. 20, no. 4, pp. 382–399, 2006.

[13] Jun Wu and Sanjeev Khudanpur, "Combining nonlocal, syntactic and n-gram dependencies in language modeling," in *Proceedings of Eurospeech*, 1999, vol. 99, pp. 2179–2182.

[14] Stanley F Chen, Kristie Seymore, and Ronald Rosenfeld, "Topic adaptation for language modeling using unnormalized exponential models," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. IEEE, 1998, vol. 2, pp. 681–684.

[15] Jun'ichi Kazama and Jun'ichi Tsujii, "Evaluation and extension of maximum entropy models with inequality constraints," in *Proceedings of EMNLP*, 2003, pp. 137–144.

[16] Radford M Neal and Geoffrey E Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*, pp. 355–368. Springer, 1998.

[17] Abhinav Sethy, Stanley F Chen, and Bhuvana Ramabhadran, "Distributed training of large scale exponential language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5520–5523.