RC24970 (W1004-019) April 6, 2010
Computer Science

IBM Research Report

Scaling Shrinkage-Based Language Models

Stanley F. Chen, Lidia Mangu, Bhuvana Ramabhadran, Ruhi Sarikaya, Abhinav Sethy
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Scaling Shrinkage-Based Language Models

Stanley F. Chen, Lidia Mangu, Bhuvana Ramabhadran, Ruhi Sarikaya, Abhinav Sethy
IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
{stanchen,mangu,bhuvana,sarikaya,asethy}@us.ibm.com

Abstract

In (Chen, 2009b), we show that a novel class-based language model, Model M, and the method of regularized minimum discrimination information (rMDI) models outperform comparable methods on moderate amounts of Wall Street Journal data. Both of these methods are motivated by the observation that *shrinking* the sum of parameter magnitudes in an exponential language model tends to improve performance (Chen, 2009a). In this paper, we investigate whether these shrinkage-based techniques also perform well on larger training sets and on other domains. First, we explain why good performance on large data sets is uncertain, by showing that gains relative to a baseline n-gram model tend to decrease as training set size increases. Next, we evaluate several methods for data/model combination with Model M and rMDI models on limited-scale domains, to uncover which techniques should work best on large domains. We also show how to speed up Model M by using *unnormalized* exponential models. Finally, we apply these methods on a variety of medium-to-large-scale domains covering several languages, and show that Model M consistently provides significant gains over existing language models for state-of-the-art systems in both speech recognition and machine translation.

1 Introduction

In (Chen, 2009b), we proposed a novel class-based language model, *Model M*, that outperforms a Katz-smoothed word trigram model by 28% in perplexity and 1.9% absolute in automatic speech recognition (ASR) word-error rate; these are among the best results ever reported for a class-based language model. In addition, we showed that for the task of domain adaptation, the method of *regularized minimum discrimination information* (rMDI) modeling outperforms linear interpolation by up to 0.7% absolute in word-error rate (WER). However, these experiments were restricted to Wall Street Journal data with training sets less than 25 million words in length and were conducted with a non-state-of-the-art acoustic model. While Wall Street Journal is the canonical test bed for language modeling (LM) research, it is not representative of the data used in modern language modeling applications, many of which use languages other than English. Furthermore, the use of training sets of 1 billion words or more is not uncommon, *e.g.*, (Brants et al., 2007).

In this paper, we investigate whether the gains of Model M and regularized minimum discrimination information models scale to larger data sets, other domains and languages, and other applications, specifically, machine translation (MT). One particular concern is that both Model M and rMDI models were motivated as ways to *shrink* a word n-gram model. That is, when training and test data are drawn from the same distribution, it has been found for many types of exponential language models that

$$\log \text{PP}_{\text{test}} \approx \log \text{PP}_{\text{train}} + \frac{\gamma}{D} \sum_i |\tilde{\lambda}_i| \tag{1}$$

where PP_{test} and PP_{train} denote test and training set perplexity; D is the number of words in the training data; $\tilde{\lambda}_i$ are *regularized* (*i.e.*, smoothed) estimates of the model parameters; and γ is a constant independent of domain, training set size, and model type (Chen, 2009a; Chen, 2008). Thus, one can improve test performance by shrinking the parameter sum $\sum_i |\tilde{\lambda}_i|$, and both Model M and rMDI models are designed to improve upon word n-gram models in this way. However, as training set size increases, the last term in eq. (1) tends to grow smaller, which suggests the gain to be had by shrinking parameter values will also decrease. Thus, it is uncertain whether Model M and rMDI models will retain their performance improvements over word n-gram models on larger training corpora.

The outline of this paper is as follows: In Section 2, we review Model M and rMDI models. In Section 3, we elaborate on why performance gains decrease as training sets grow, and show how gains vary for some actual models. In Section 4, we examine the task of model combination when using Model M and rMDI models, as this is a key issue when tackling large-scale domains. In Section 5, we discuss the run-time speed and training time of these models, and show how Model M can be accelerated with no loss in performance by using an *unnormalized* version of this model. In Section 6, we apply these methods to a variety of medium-to-large-scale tasks: English voicemail transcription; English Broadcast News transcription; GALE Arabic transcription; and Arabic/English and Spanish/English machine translation. Finally, we present some conclusions in Section 7.

2 Background

In this section, we review Model M and rMDI models as well as the results for performance prediction for exponential language models given in (Chen, 2009a; Chen, 2008). An exponential model $p_\Lambda(y|x)$ is a model with a set of features $\{f_i(x,y)\}$ and equal number of parameters $\Lambda = \{\lambda_i\}$ where

$$p_\Lambda(y|x) = \frac{\exp(\sum_i \lambda_i f_i(x,y))}{\sum_{y'} \exp(\sum_i \lambda_i f_i(x,y'))} \tag{2}$$

Remarkably, eq. (1) holds for many exponential language models including Model M and rMDI models; the relationship is strongest if the $\tilde{\Lambda} = \{\tilde{\lambda}_i\}$ are estimated using $\ell_1 + \ell_2^2$ regularization (Kazama and Tsujii, 2003); *i.e.*, parameters are chosen to optimize

$$\mathcal{O}_{\ell_1+\ell_2^2}(\Lambda) = \log \text{PP}_{\text{train}} + \frac{\alpha}{D} \sum_i |\lambda_i| + \frac{1}{2\sigma^2 D} \sum_i \lambda_i^2 \tag{3}$$

for some α and σ. For a data set $\mathcal{D} = (x_1, y_1), \ldots, (x_D, y_D)$, log perplexity (or cross-entropy) can be computed as $-\frac{1}{D}\sum_{j=1}^{D} \log p_\Lambda(y_j|x_j)$. When using natural logs in eq. (1) and taking $(\alpha = 0.5, \sigma^2 = 6)$, the constant $\gamma = 0.938$ yields a mean error equivalent to a few percent in perplexity over the models evaluated in (Chen, 2008). These values of α and σ also yield good test set performance over a wide variety of training sets. Thus, the right-hand side of eq. (1) is a faithful proxy for test set perplexity. In practice, we have found that improving this proxy score for a model leads not only to improvements in test set perplexity, but in speech recognition word-error rate as well.

It follows that if one can *shrink* the "size" of a model (proportional to $\sum_i |\lambda_i|$) while not damaging training set performance, test set performance should improve. In (Chen, 2009b), we use this reasoning to motivate Model M and rMDI models, two types of exponential language models. Model M is a class-based n-gram model that can be viewed as the result of shrinking an exponential

word n-gram model using word classes. If we assume each word w is mapped to a single class $c(w)$, we can write

$$p(w_1 \cdots w_l) = \prod_{j=1}^{l+1} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) \times \prod_{j=1}^{l} p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) \quad (4)$$

where c_{l+1} is the end-of-sentence token. Let f_θ denote a binary n-gram feature such that $f_\theta(x, y) = 1$ iff xy "ends" in the n-gram θ. Let $p_{\text{ng}}(y|\theta)$ denote an exponential n-gram model, where we have a feature $f_{\theta'}$ for each suffix θ' of each θy occurring in the training set. For example, the model $p_{\text{ng}}(w_j|w_{j-1}c_j)$ has a feature f_θ for each n-gram θ in the training set of the form w_j, $c_j w_j$, or $w_{j-1}c_j w_j$. Let $p_{\text{ng}}(y|\theta_1, \theta_2)$ denote a model containing all features in $p_{\text{ng}}(y|\theta_1)$ and $p_{\text{ng}}(y|\theta_2)$. Then, we can define (the trigram version of) Model M as

$$\begin{aligned} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) &\equiv p_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \\ p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) &\equiv p_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \quad (5)$$

Regularized minimum discrimination information (rMDI) models can be viewed as the result of shrinking an exponential model using a prior distribution. Minimum discrimination information (MDI) models (Della Pietra et al., 1992) have the form

$$p_\Lambda(y|x) = \frac{q(y|x) \exp(\sum_i \lambda_i f_i(x, y))}{\sum_{y'} q(y'|x) \exp(\sum_i \lambda_i f_i(x, y'))} \quad (6)$$

for some prior distribution $q(y|x)$. While regularization is not used in (Della Pietra et al., 1992), we found in (Chen, 2009a) that when regularizing $p_\Lambda(y|x)$ in the way described earlier, eq. (1) holds for these models if $q(y|x)$ is ignored in computing model size (assuming $q(y|x)$ is estimated on an independent training corpus). Regularized MDI models are well-suited to the task of domain adaptation, where one has a test set and small training set from one domain, and a large training set from a different domain. One can build a language model on the outside domain, and use this model as the prior when building a model on the in-domain data. While exponential models of any form can be used in eq. (6), (Chen, 2009b) evaluated the use of exponential word n-gram models, *i.e.*, models of the form $p_{\text{ng}}(w_j|w_{j-2}w_{j-1})$ (for trigrams), and found good performance relative to linear interpolation and count merging on small data sets. In this paper, we also evaluate a variant of rMDI, *cascaded rMDI*, that can be used to combine an arbitrary number of training corpora rather than just two. In this method, one orders the available corpora from most "out-of-domain" to most "in-domain" and applies the rMDI technique repeatedly. That is, one first builds a model on the most out-of-domain corpus and uses this as a prior when building a model on the next most out-of-domain corpus. The resulting model is then used as the prior when training a model on the *next* most out-of-domain corpus, etc.

3 Analyzing How Models Scale

In this section, we discuss why it's important to study how models scale with training set size, *i.e.*, why good performance on small data sets often does not carry over to large ones. One obvious reason to worry about this issue is that many algorithms in the literature have been shown not to scale well. Here, we show how to explain this phenomenon for many types of models by using eq. (1), and study how this effect affects Model M and rMDI models by plotting relative performances over a variety of training set sizes.

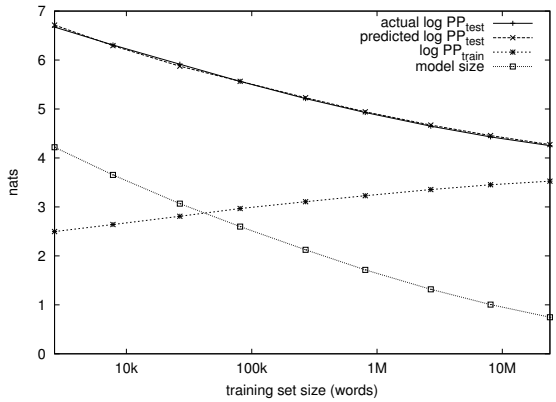

Figure 1: Predicted and actual $\log \text{PP}_{\text{test}}$, $\log \text{PP}_{\text{train}}$, and model size ($\frac{\gamma}{D} \sum_i |\tilde{\lambda}_i|$) for word trigram models built on varying amounts of WSJ data. A nat is a "natural" bit, or $\log_2 e$ regular bits.

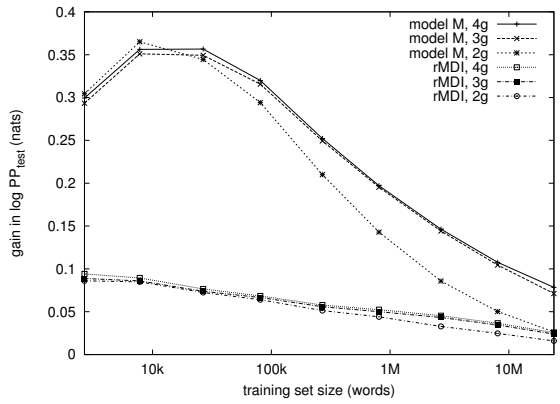

Figure 2: Gains in $\log \text{PP}_{\text{test}}$ for Model M and rMDI models as compared to a word n-gram baseline, for bigram, trigram, and 4-gram models built on varying amounts of WSJ data. For rMDI, the out-of-domain corpus is Broadcast News text and is the same length as the in-domain WSJ corpus.

If we define the *size* of a model p_Λ to be $\frac{\gamma}{D} \sum_i |\tilde{\lambda}_i|$, eq. (1) tells us that test performance (in log PP) is approximately equal to the sum of training performance (in log PP) and model size. In Figure 1, we graph these quantities over varying amounts of training data (from 100 to 900k sentences) for an exponential word trigram model built on Wall Street Journal (WSJ) data with a 21k word vocabulary. As with all other models discussed in this paper, we use $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$. While the sum $\sum_i |\tilde{\lambda}_i|$ grows with more data, it grows slower than D, so the overall model size tends to decrease as we go to the right. Unintuitively, training perplexity tends to *increase* as training set size increases. We can explain this by noting that with more training data, the model cannot overfit the data as much.

Because Model M and rMDI models achieve their performance improvements over n-gram models by shrinking model size, it seems likely that if n-gram model sizes decrease, so will the shrinkage gain. In the limit of infinite data, we expect the size of a trigram model, say, to go to zero and hence expect no improvement from the corresponding Model M or rMDI models. These models condition their predictions on exactly two words of history, so they can do no better than

an "ideal" trigram model. Hence, the issue is not *whether* Model M and rMDI model gains will disappear as training set size grows, but *when*.

In Figure 2, we display gain in $\log \text{PP}_{\text{test}}$ relative to a word n-gram model for Model M and rMDI models for $n \in \{2, 3, 4\}$. For Model M, we build 150 word classes using the algorithm of (Brown et al., 1992) on our largest training set and use these classes in all runs. For the rMDI models, we use exponential n-gram models and combine the in-domain WSJ training set with an out-of-domain Broadcast News training set of equal length (in sentences). As predicted, gains generally decrease as the training set expands. Similarly, gains are smaller for smaller n since n-gram model sizes shrink as n shrinks.[1]

At the right edge of the graph, the gains for most algorithms are 0.03 nats or less, where a nat is a "natural" bit, or $\log_2 e$ regular bits. Each 0.01 nat difference corresponds to about 1% in PP. However, the gain for 4-gram Model M is 0.08 nats, which translates to a 1.4% absolute reduction in word-error rate (22.2% $\Rightarrow$ 20.8%) using the ASR setup described in Section 4.1. While gain is dropping as training set size increases, Model M still appears promising for data sets substantially larger than 900k sentences. Interestingly, while the gain from Model M over a word n-gram model is only about 3% in PP for a bigram model on the largest training set, the gain in WER is still 0.6% absolute (25.7% $\Rightarrow$ 25.1%), so it is possible that small PP gains for Model M still translate to significant WER gains. In Section 6, we present experiments on larger data sets and see whether these predictions hold true.

4 Scaling Model Combination

For large-scale domains, one typically has language model training data from multiple sources; for example, the IBM GALE Arabic ASR system uses 16 separate corpora. Furthermore, these corpora generally differ in relevance and amount, and aggregating the data into a single corpus may not work best. Thus, a central issue in handling large domains is how to best combine multiple data sets or models. In this section, we attempt to discover the best methods for combining Model M models and to characterize when rMDI modeling can improve model combination performance. We use small and medium-sized data sets so we can evaluate a large number of methods under a large number of conditions, and attempt to predict performance on large tasks via extrapolation. We use these findings to inform which algorithms to assess in the large-scale experiments in Section 6.

The best way to combine data or models will depend on the relationship between the training and test corpora, so we investigate two different scenarios. In Section 4.1, we consider a typical domain adaptation task where we have a modest amount of training data from the same domain as the test data, and equal or larger amounts of out-of-domain data. In Section 4.2, we consider a model combination task where we have many corpora from similar domains as the test data.

4.1 Domain Adaptation

These ASR experiments are an expanded version of the domain adaptation experiments in (Chen, 2009b); here, we consider more corpora, larger data sets, and more algorithms. The acoustic model is a cross-word quinphone system built from 50h of Broadcast News data and contains 2176 context-dependent states and 50k Gaussians. The front end is a 13-dimensional PLP front end with cepstral mean subtraction; each frame is spliced together with four preceding and four succeeding frames

[1] In addition to shrinking model size, another way to improve test performance is to improve training set performance. One way to do this is to increase the order of an n-gram model. Up to some limit, gains grow as training set size increases, as the training set performance of a lower-order model saturates before that of a higher-order one.

	in-domain (WSJ) training set (sents.)			
	1k	10k	100k	900k
word n-gram models				
WSJ only				
KN n-gram	34.5%	30.4%	26.1%	22.6%
exp. n-gram	34.6%	30.3%	25.7%	22.5%
WSJ and BN, 1:1 ratio				
interp	34.3%	30.0%	25.4%	22.3%
merge	34.1%	29.6%	25.0%	22.1%
rMDI	34.0%	29.6%	25.1%	22.1%
merge/rMDI	34.0%	29.6%	25.1%	22.1%
WSJ and BN, 1:3 ratio				
interp	33.9%	29.6%	25.1%	
merge	33.5%	29.3%	25.1%	
rMDI	33.4%	29.0%	24.7%	
merge/rMDI	33.5%	28.9%	24.6%	
WSJ and BN and SWB, 1:1:1 ratio				
interp	34.3%	30.0%	25.4%	22.3%
merge	33.8%	29.6%	25.5%	22.2%
casc. rMDI	33.9%	29.5%	25.1%	22.1%
merge+rMDI	33.8%	29.5%	25.1%	22.2%
WSJ and BN and SWB, 1:3:10 ratio				
interp	33.8%	29.7%	25.0%	
merge	33.3%	29.3%	25.2%	
casc. rMDI	33.1%	28.7%	24.6%	
merge+rMDI	33.1%	28.8%	24.7%	
Model M				
WSJ only				
Model M	35.3%	29.1%	24.2%	21.5%
WSJ and BN, 1:1 ratio				
interp (WSJ/BN cl.)	34.9%	28.4%	23.9%	21.2%
interp (WSJ/WSJ cl.)	33.9%	28.3%	23.9%	21.2%
merge	33.9%	28.2%	23.9%	21.2%
rMDI	34.8%	28.6%	23.8%	21.2%
merge/rMDI	34.3%	28.6%	23.9%	21.3%
WSJ and BN, 1:3 ratio				
interp (WSJ/BN cl.)	34.0%	27.8%	23.6%	
interp (WSJ/WSJ cl.)	32.8%	27.6%	23.8%	
merge	33.5%	27.6%	23.4%	
rMDI	34.3%	28.1%	23.7%	
merge/rMDI	33.9%	27.9%	23.6%	

Table 1: Word-error rates for various methods for domain adaptation.

and then LDA is performed to yield 40-dimensional feature vectors. We use a 47k-word WSJ test set and in-domain WSJ training sets of various sizes. For the out-of-domain data, we consider the cases where only Broadcast News (BN) data is available and where both BN and Switchboard (SWB) data are available, and assume the length of each out-of-domain corpus (in sentences) is a particular multiple of the length of the in-domain corpus, either 1, $\sim$3, or $\sim$10.

To evaluate our language models, we use lattice rescoring. We generate lattices on both our development and evaluation data sets using the LattAIX decoder (Saon et al., 2005) in the Attila speech recognition system (Soltau et al., 2005). The language model for lattice generation is created by building a modified Kneser-Ney-smoothed word trigram model on our largest WSJ training set; this model is pruned to contain a total of 350k n-grams using the algorithm of Stolcke (1998). We choose the acoustic weight for each model to optimize word-error rate on the development set; we do a Powell search to find the best weight with a granularity of 0.002 (unlike in our previous work where we used a granularity of 0.005).

We compare the techniques of *linear interpolation*, *count merging*, and rMDI modeling. In linear interpolation, separate language models are built on each corpus and linearly interpolated, with interpolation weights being optimized on a held-out set (Jelinek et al., 1991). In count merging, the component corpora are concatenated into a single corpus, and a single language model is built on the merged data set. Count merging has been motivated as an instance of MAP adaptation (Federico, 1996; Masataki et al., 1997). Optionally, a component corpus may be replicated multiple times to weight the data more; we do not consider this option here for simplicity. Unlike in linear interpolation where each model is assigned a fixed weight independent of history for each word prediction, count merging can be viewed as assigning a weight proportional to the history count of each model. In contrast, rMDI modeling can be viewed as backing off from the in-domain model to the out-of-domain model.

In Table 1, we display our ASR WER results. The top part of the table corresponds to word n-gram models, while the bottom part corresponds to Model M. Each column represents a different in-domain training set size. Each subsection of the table corresponds to using a different amount of out-of-domain data. For example, the *WSJ and BN and SWB, 1:3:10 ratio* section corresponds to using a BN corpus three times larger than the in-domain data and a SWB corpus ten times larger than the in-domain data. All of the word n-gram models are exponential n-gram models except for the first row, which corresponds to a conventional word n-gram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998). We use the trigram versions of each model.

The algorithm *merge/rMDI* corresponds to using count merging to merge the in-domain and out-of-domain corpora into one corpus and then using rMDI to combine the in-domain and merged corpus. The algorithm *merge+rMDI* corresponds to using count merging to merge the two out-of-domain corpora into one corpus and then using rMDI to combine the in-domain and merged out-of-domain corpus. The *cascaded rMDI* method amounts to using rMDI to repeatedly do 2-way model combination; *i.e.*, we first use rMDI to combine the BN and SWB data (with the SWB data being the out-of-domain corpus), and then use this model as the prior when training on the WSJ data.

Unlike in Section 3, we induce word classes on the given training set(s), rather than always using word classes from the largest training set. This is why Model M performs poorly on the smallest training set. We note that it is straightforward to combine rMDI domain adaptation with Model M; one can simply do rMDI domain adaptation separately for each of the two component models given in eq. (5), as long as the same word classes are used everywhere.

When doing domain adaptation, one must select which word classes are used with each training corpus. In the *interp (WSJ/WSJ cl.)* runs, we use classes induced from the in-domain training set for

word n-gram models

	PP	WER
interp+, rMDI	180.8	14.3%
interp, rMDI	181.2	14.4%
interp+, exp.	184.9	14.4%
merge, exp.	188.4	14.5%
interp, KN	190.6	14.5%
interp, exp.	194.0	14.6%
merge, KN	194.2	14.6%
casc. rMDI	210.1	14.8%

Model M

	PP	WER
interp+, rMDI	169.3	13.7%
interp+	169.4	13.6%
interp, rMDI	170.1	13.7%
merge	175.0	13.8%
interp	175.3	13.7%
casc. rMDI	203.3	14.2%

Table 2: Word-error rates for various methods for model combination.

all corpora. In the *interp (WSJ/BN cl.)* runs, we induce the word classes used for each corpus from *that* corpus. For *merge*, we induce classes on the merged corpus, and for *rMDI*, we induce classes on the in-domain training set.

For word n-gram models, the rMDI methods generally perform best or near best in all conditions, where we do not see appreciable differences between rMDI and *merge/rMDI* or between cascaded rMDI and *merge+rMDI*. While WER gains for rMDI over interpolation can be as large as 1% absolute (10k sentence in-domain data, 1:3:10 ratio), the difference between techniques when using 900k sentences of in-domain data is much smaller. Intuitively, the backoff-like behavior of rMDI should be well-suited to domain adaptation, as it seems reasonable that in-domain counts should take priority over out-of-domain counts, when present.[2]

Overall, Model M outperforms word n-gram models for all of the training sets except the smallest, and gains from domain adaptation are comparable to those for word n-gram models. However, with Model M, rMDI does not perform particularly well, and no one algorithm dominates the others. For the 900k-sentence in-domain training set, there is no significant difference between algorithms. In summary, for larger training sets, we hypothesize that when combining word n-gram models for domain adaptation, rMDI may yield small gains over other methods; for Model M, we predict that all methods will perform about equally. Domain adaptation experiments on larger corpora are presented in Section 6.1.

4.2 Model Combination

In these experiments, we use the same data sets as in the English Broadcast News task described in Section 6.2, except we subsample each training set to $\frac{1}{10}$th its size and build trigram versions of each model instead of 4-gram models. There are a total of six training corpora ranging in size from 170k words to 14.7M words after subsampling; each contains Broadcast News data of some sort. Thus, this task is qualitatively different from our domain adaptation task, where some corpora are clearly in-domain and others are not. As in the previous section, we compare linear interpolation, count merging, and (cascaded) rMDI modeling, for combining both word n-gram models and Model M. We consider both exponential n-gram models and conventional n-gram models smoothed with modified Kneser-Ney smoothing. Cascaded rMDI requires that corpora be ordered from most out-of-domain to most in-domain. To do this, we build n-gram models on each corpus and compute the

[2]This assumes that the in-domain data really is in-domain and the out-of-domain is not. In contrast, if the "out-of-domain" data is actually in-domain, we would expect that using count merging to merge the corpora would do best.

	training set (sents.)			
	1k	10k	100k	900k
normalized	31.0%	27.1%	23.8%	21.5%
unnorm., unreg. hist.	30.8%	26.9%	23.7%	21.3%
unnorm., reg. hist.	30.9%	26.9%	23.7%	21.5%
unnorm., no hist.	30.9%	27.0%	23.9%	21.5%

Table 3: Word-error rates for normalized and unnormalized versions of Model M. For unnormalized models, we evaluate using unregularized history (or normalization) features, regularized history features, and no history features at all. Experiments are for the trigram version of Model M using 150 word classes on WSJ data.

perplexity of an in-domain held-out set to guide us.

One unappealing aspect of linear interpolation is that when one of the component models has no counts for a particular history (while the others do), it still gets its full prediction weight. We can attempt to improve prediction in this situation by combining each component model with a "general" model built on all of the training data combined, *i.e.*, the count-merged model. This is analogous to the technique used in topic-based language modeling of combining each topic-specific model with a topic-independent model, *e.g.*, (Seymore and Rosenfeld, 1997). We consider two different ways of combining each corpus-specific model with the general model: linear interpolation and rMDI modeling. With linear interpolation, interpolating each component model with the general model is equivalent to just adding the general model into the overall interpolation. In rMDI modeling, we use the general model as the prior when training each corpus-specific model.

In Table 2, we display development set PP and test set WER for a variety of model combination algorithms applied to both word n-gram models and Model M. The notation *interp+* refers to doing interpolation where the general/count-merged model is included in the mix; *exp.* means exponential n-gram models whereas *KN* refers to conventional n-gram models; and *rMDI* (with interpolation) refers to training each corpus-specific model using the general model as a prior. In each subtable, algorithms are ordered in terms of increasing PP on the development set.

The most popular model combination techniques are linear interpolation and count merging with conventional n-gram models, yielding WER's of 14.5% and 14.6%, respectively. The algorithm yielding the best performance on the development set is *interp+, rMDI*, giving a WER of 14.3% for word n-gram models and 13.7% for Model M. However, a WER of 13.7% can also be achieved through simple linear interpolation with Model M. In summary, we speculate that for large training sets when using word n-gram models, small gains over simple interpolation may be possible with *interp+, rMDI*, but at the cost of a significantly larger model. With Model M, simple linear interpolation is the easiest to implement and performs as well as any other method.[3] We note that cascaded rMDI performs by far the worst, probably because model combination is a very different task than domain adaptation, the task which rMDI is tailored for. In Section 6.2, we present results on the full-scale Broadcast News task.

[3]One practical question with interpolation is that when one has data from two very similar sources, when should these corpora be merged into one? If the corpora are identical in nature, we expect merging should help. To give one data point, we compared combining two 100k-sentence WSJ training sets using count merging and linear interpolation, and this yielded WER's of 24.7% and 24.9%, respectively.

5 Speeding Things Up

In this section, we discuss how model training and probability evaluation can be accelerated through the use of *unnormalized* models, which can perform as well as or better than normalized models (Lebanon and Lafferty, 2001). An unnormalized model is defined just as in eq. (2) except without an explicit normalization term:

$$p_\Lambda(y|x) = \exp(\sum_i \lambda_i f_i(x,y)) \tag{7}$$

For probability computation, unnormalized models can be much faster since there is no need to compute the normalizer $Z_\Lambda(x) = \sum_{y'} \exp(\sum_i \lambda_i f_i(x,y'))$, which can involve computing the probability of every token in the vocabulary. However, for exponential n-gram models, the number of distinct $Z_\Lambda(x)$ is equal to the number of unique n-gram histories in the training data, so all of these values can be precomputed.[4] While most of the models discussed in this paper are exponential n-gram models, the class prediction model given in eq. (5) for Model M contains the features from *two* exponential n-gram models, and this optimization does not apply.

To train an unnormalized model, instead of optimizing the objective function given by eq. (3), one adds an additional penalty term $\frac{1}{D}\sum_j(\sum_y p_\Lambda(y|x_j) - 1)$. This corresponds to using the *extended* Kullback-Leibler divergence rather than the conventional Kullback-Leibler divergence between the model and the training data. (Notice that this penalty term is zero for normalized models.) To optimize this objective function, one can use iterative scaling exactly as before: the expectation computation and parameter updates are the same as for normalized models.

However, one issue that arises for unnormalized models is whether extra features that approximate normalization should be added. That is, the values $Z_\Lambda(x)$ in normalized models can be viewed as corresponding to additional features that enforce the constraint that all conditional probabilities correctly sum to 1. As these features are lacking in unnormalized models, it may make sense to create similar features to compensate for this loss. In this work, we consider adding a normalization feature corresponding to the history of each n-gram feature in our model; *i.e.*, for each n-gram feature f_θ, we create a feature $f_{\text{hist}(\theta)}$ such that $f_{\text{hist}(\theta)}(x,y) = 1$ iff there is a y' such that $f_\theta(x,y') = 1$. The number of additional normalization features is equal to the number of unique n-gram histories in the training data, which should be a fraction of the total number of regular features. When training normalization features, we consider both using the same regularization hyperparameters as for regular features and not regularizing at all, *i.e.*, taking $(\alpha = 0, \sigma^2 = \infty)$.

In Table 3, we display ASR performance using the setup from (Chen, 2009b) for normalized and unnormalized versions of Model M for various training set sizes. We evaluate both using and not using normalization (or history) features. While performance differences are quite small, using normalization features without regularization appears to be slightly better than the other unnormalized variants. As compared to normalized Model M, there appears to be no loss in performance. In terms of speed, we benchmarked various 4-gram models built on 900k sentences of WSJ data on a 2.8GHz Xeon CPU. Using our general exponential model implementation, normalized Model M probabilities can be evaluated at the rate of 35k lookups/s while the unnormalized version runs at 84k lookups/s. For reference, this implementation executes exponential word n-gram models at 140k lookups/s, while our code specific to word n-gram models runs at 300k lookups/s. Due to time constraints, we were unable to evaluate unnormalized models on large data sets in Section 6.

[4] In fact, one can convert an exponential n-gram model to an ARPA format n-gram model without loss (Chen and Rosenfeld, 2000).

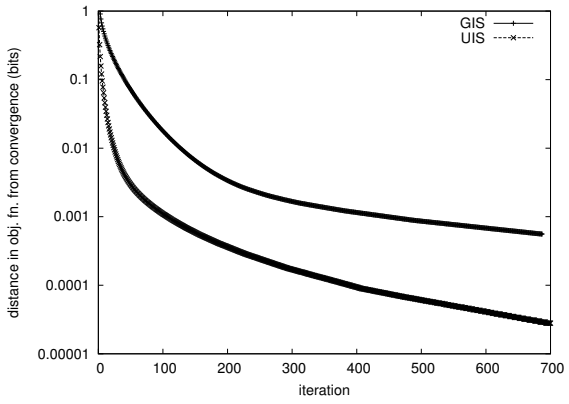

Figure 3: Convergence speed for GIS and UIS for exponential word trigram model trained on 10k sentences of WSJ. The y-axis represents the difference in the current value of the training objective function as given in eq. (3) as compared to the (estimated) value of the objective function at convergence.

5.1 Training

We have found that the ideas in unnormalized modeling can be used to accelerate iterative scaling for both unnormalized *and* normalized models. In generalized iterative scaling (GIS), each feature f_i has an observed training count $O[f_i]$ and an expectation $E_{p_\Lambda}[f_i] = \sum_j \sum_y p_\Lambda(y|x_j) f_i(x_j, y)$ given the current parameters Λ (Darroch and Ratcliff, 1972). In each iteration, one does an update of the form (ignoring regularization)

$$\lambda_i \Leftarrow \lambda_i + \frac{1}{f^\#} \log \frac{O[f_i]}{E_{p_\Lambda}[f_i]} \tag{8}$$

where $f^\#$ is the maximum number of simultaneously active features among the features being updated. It has been found that updating a subset of nonoverlapping features (such that $f^\# = 1$) in each iteration can be much faster than updating all features simultaneously (with $f^\# > 1$), *e.g.*, (Goodman, 2002). The problem with this approach is that recomputing $E_{p_\Lambda}[f_i]$ after each update is generally quite expensive, so this approach is only beneficial when this recomputation can be optimized.[5]

We observe that for *unnormalized* exponential n-gram models, there exists a special case when some of the $E_{p_\Lambda}[f_i]$ can be recomputed efficiently after an update. To give an example, consider updating the parameter of a unigram feature f_{w_j} by the amount $\Delta\lambda_{w_j}$. Note that the expectation $E_{p_\Lambda}[f_{w_{j-1}w_j}]$ for any bigram feature of the form $f_{w_{j-1}w_j}$ changes by the factor $e^{\Delta\lambda_{w_j}}$, and similarly for any other features corresponding to n-grams ending in w_j. (This won't hold for normalized models because of normalization.) This suggests the following variant of iterative scaling: In each iteration, we first compute all expectations $E_{p_\Lambda}[f_i]$ as in normal GIS. Then, we first update parameters for all unigram features. As these are non-overlapping, we can use a full step size with $f^\# = 1$. Then, we can efficiently compute how $E_{p_\Lambda}[f_i]$ will change for all bigram features given the unigram parameter updates, and we can update all bigram parameters with a full step size. Similarly, we can

[5]We note that improved iterative scaling (Della Pietra et al., 1997) is equivalent to GIS for unpruned n-gram models as exactly n features are active for each event. Sequential conditional GIS (Goodman, 2002) is generally not applicable for n-gram models due to its memory requirements.

proceed to update all higher-order n-gram parameters with a full step size. Thus, we can update all parameters with a full step size in each iteration, rather than with a $\frac{1}{n}$ step size as in conventional GIS.

We can apply this variant of iterative scaling, which we call *unnormalized iterative scaling* (UIS), to normalized exponential n-gram models as well. As noted earlier, a normalized exponential model can be viewed as an unnormalized exponential model with additional normalization features corresponding to the values $Z_\Lambda(x)$. Thus, we can do the same updates as for unnormalized models in each iteration; this can be viewed as updating all of the features in the model except for the normalization features. To "update" the normalization features, we can just recompute $Z_\Lambda(x)$ at the end of each iteration.

In addition to exponential n-gram models, we can also apply UIS to models that contain multiple sets of n-gram features, such as the class prediction model given in eq. (5) for Model M. In particular, we can use UIS to update only a single set of n-gram features in each iteration, and alternate between sets in different iterations.[6]

In Figure 3, we plot training objective function distance from convergence as a function of training iteration for GIS and UIS for a word trigram model trained on 10k sentences of WSJ. We estimate the value of the training objective function at convergence by running a very large number of training iterations. Clearly, UIS converges much faster; to get the training objective function to within 0.01 bits of convergence, UIS requires 25 iterations while GIS requires 128 iterations; to get within 0.001 bits, the values are 107 and 448 iterations, respectively. To give some idea of overall training times, on a 2.66GHz Xeon 5150, training an exponential word 4-gram model on 900k sentences of WSJ data takes about 3h; for Model M, the corresponding training time (excluding word class building) is about 8h. Training times appear to be slightly sublinear in training set size. In our implementation, we use *cluster expansion* (Lafferty and Suhm, 1995) to accelerate the computation of expectations.

6 Experiments

In this section, we investigate whether Model M and rMDI modeling can improve the performance of existing medium and large-scale state-of-the-art systems, three for ASR and one for machine translation. For each system, we compare against the current best language model for that system trained on all available training data; except where noted, this is the system we refer to as the baseline. We evaluate the best methods found in Section 4, but also do contrast runs with other methods to attempt to confirm the findings in that section. While Model M gives consistent gains over word n-gram models in Section 4, we verify whether these gains carry over to larger data sets.

All exponential models are trained with $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$; conventional n-gram models are trained using modified Kneser-Ney (KN) smoothing (Chen and Goodman, 1998). Unless otherwise noted, we use the 4-gram version of each model; we induce 150 word classes using the algorithm of (Brown et al., 1992) for Model M; and interpolation weights are trained to optimize the perplexity of a held-out set. Experiments with Model M are substantially more expensive in both time and memory than those with n-gram models, partially due to algorithmic considerations and partially because our exponential model code has not yet been optimized much. This constrained the number of Model M experiments we were able to run.

[6] In similar fashion, for unnormalized models with normalization features, we update only normalization features or only regular features in each training iteration.

word n-gram models

	WER
interp, KN n-gram	16.9%
rMDI, exp. n-gram	16.7%
merge, exp. n-gram	16.6%
interp, exp. n-gram	16.6%

Model M

	WER
rMDI	16.4%
merge	16.3%
interp	16.3%

Table 4: Comparison of language models on the voicemail transcription task.

6.1 English Voicemail Transcription

We evaluate the performance of Model M and rMDI models on the task of English voicemail transcription. Recently, ASR is increasingly being deployed in unified messaging systems to serve as an aid to human transcribers or as a standalone service. Here, we report on an in-house voicemail transcription task. The ASR system is based on the 2007 IBM GALE speech transcription system (Chen et al., 2006). The discriminatively-trained acoustic model was trained on 2000h of voicemail messages using speaker-adapted PLP features and contains 8000 context-dependent states and 300k Gaussians.

We have two sources of language model data: the verbatim transcripts of the acoustic training data (17M words), and 41M words of approximate voicemail transcripts cleaned up for readability. The first corpus is very well-matched to the test set; the second corpus less so. The baseline language model, built using a 40k-word lexicon, is the interpolation of two word 4-gram models, one trained on each of the LM training corpora. The 5.5h test set consists of 900 messages and 62k words; the perplexity of the baseline LM on this set is 43 and the WER is 16.9%. Language models are evaluated via lattice rescoring on lattices generated using the baseline LM. Increasing the lexicon size to 150k words did not impact the WER.

To decide which model combination method should work best with Model M, the main issue is whether the two corpora are similar enough to be considered a single corpus or not. If so, we expect count merging to do best; if not, we expect linear interpolation to do as well as anything else. In Table 4, we display the results for various algorithms. The first row in the table on the left represents the baseline method. The second and third rows represent replacing conventional n-gram models with exponential n-gram models, using either linear interpolation or rMDI models for model combination. The table on the right corresponds to using Model M instead of word n-gram models. Model M yields the best performance; a WER of 16.3% is obtained both through count merging and interpolation (using the same weights as in the baseline model), a gain of 0.6% absolute. Notably, the best Model M result surpasses the best word n-gram model number by only 0.3% absolute, which is significantly less than the gains found earlier. We speculate this is because the word n-gram models are already quite well-estimated: the merged model has a model size of only 0.7 nats/event.

For model combination, linear interpolation slightly outperforms rMDI models. We hypothesize that this is because the "out-of-domain" corpus is actually quite in-domain, in which case rMDI adaptation does not give the desired behavior.

6.2 English Broadcast News Transcription

In this section, we examine whether Model M can improve performance on an English Broadcast News task. The ASR system is based on the 2007 IBM GALE speech transcription system. The

	DEV07	DEV08	EVAL08
Interpolation over all 16 corpora			
Baseline: 16 KN LMs	9.5%	11.0%	9.4%
5 Model M + 11 KN LMs	9.1%	10.6%	9.0%
3 M (500c) + 2 M (150c) + 11 KN	9.0%	10.4%	8.9%
Interpolation over 5 of 16 corpora			
5 KN LMs	10.0%	11.3%	9.6%
5 Model M LMs	9.4%	10.8%	9.0%

Table 5: Word-error rates for interpolated LMs on several GALE Arabic test sets, varying how many component models are word n-gram models and how many are Model M.

	DEV07	DEV08	EVAL08
KN LM	12.1%	13.2%	11.8%
Model M, 150 classes	11.6%	12.9%	11.2%
Model M, 300 classes	11.5%	12.9%	11.1%
Model M, 500 classes	11.3%	12.7%	11.1%

Table 6: Word-error rates for a single Model M on GALE Arabic for different numbers of word classes.

acoustic model was trained on 430h of Broadcast News audio including the 1996 and 1997 English Broadcast News Speech corpora (LDC97S44 and LDC98S71) and the TDT4 Multilingual Broadcast News Speech corpus (LDC2005S11). The model contains 6000 context-dependent states and 250k Gaussians and was discriminatively trained using lattice-based fMPE and MPE with backing off to MMI estimates in I-smoothing.

The LM training text consists of a total of 400M words from the following six sources: 1996 CSR Hub4 language model data; EARS BN03 closed captions; GALE Phase 2 Distillation GNG Evaluation Supplemental Multilingual data; Hub4 acoustic model training transcripts; TDT4 closed captions; and TDT4 newswire. The vocabulary is 80k words and the baseline language model is a linear interpolation of word 4-gram models, one for each corpus. Interpolation weights are chosen to optimize perplexity on a held-out set of 25k words, the rt03 evaluation set. The evaluation set is the 2.5h rt04 evaluation set containing 45k words; the WER of the baseline LM on this data set is 13.0%.

The experiments in Section 4.2 use a scaled-down version of this task, and thus we expect the same methods will work best. We build Model M on each source and interpolate them using the same weights as in the baseline, yielding a WER of 12.3%, or a gain of 0.7% absolute. To our knowledge, this is the best single-system result for this data set, surpassing the previous best of 12.6% (Gales et al., 2006). On the held-out set, the perplexity is reduced from 133 for the baseline to 121. As a contrast, we also evaluated cascaded rMDI for model combination, ordering models by their interpolation weight. This model performed much worse as in Section 4.2, yielding a WER of 13.1% and perplexity of 150.

6.3 GALE Arabic Transcription

Arabic broadcast transcription is a core component of DARPA's Global Autonomous Language Exploitation (GALE) program. In this section, we assess whether Model M can improve the performance of the best Arabic ASR system fielded in the January 2009 GALE evaluation. This will demonstrate whether Model M can outperform an optimized state-of-the-art language model where a billion words of data or more is available, in conjunction with a state-of-the-art acoustic model. The acoustic model is a discriminatively-trained Universal Background Model (Povey et al., 2008) trained on 1400h of transcribed audio (Soltau et al., 2007). We have 16 sources of language model training data totaling 1.3 billion words: transcripts of the audio data; the Arabic Gigaword corpus; newsgroup and weblog data; etc. The baseline language model has a vocabulary of 774k words and is a linear interpolation of 4-gram models (with modified Kneser-Ney smoothing) built on each of the 16 sources. Interpolation weights are chosen to minimize perplexity on a held-out set.

In our initial experiment, we build Model M models on the five corpora with the highest interpolation weights in the baseline model, with a combined weight of 0.6. We replace the corresponding n-gram models with Model M for these five sources and reoptimize interpolation weights. In the first two rows of Table 5, we present lattice rescoring results for the baseline LM and this new LM over a variety of test sets: DEV07 (2.6h), DEV08 (3h) and EVAL08 (3h). We see that a significant improvement of 0.4% absolute is achieved over the baseline even when using an off-the-shelf configuration (150 word classes) for only five out of the 16 LM sources on a state-of-the-art system with a very low baseline WER. To isolate the gains of Model M, we also display results when interpolating only the five sources under consideration. In the last two rows of Table 5, we show results for interpolating only conventional n-gram models and only Model M models; we see 0.5–0.6% absolute gain from Model M.

Given that our Arabic vocabulary is much larger than the original WSJ vocabulary used to optimize the number of word classes, we investigate whether using more than 150 word classes can improve performance. On the corpus with the highest interpolation weight in the baseline LM (Broadcast News audio transcripts, 5M words), we vary the number of word classes used with Model M. As can be seen in Table 6, 500 word classes yield the best results. We rebuild three of the five Model M models in the 16-way interpolation from before using 500 classes instead of 150, and this yields additional improvement as seen from the third row in Table 5. For reference, our best previous LM included interpolation with a 6-gram neural net LM and yielded WER's of 9.3%, 10.6%, and 9.1% on our three test sets.

6.4 Machine Translation

In this section, we evaluate whether Model M performs well on the task of machine translation. In addition, we evaluate whether the performance of Model M can be improved by linearly interpolating with a word n-gram model. We consider two different domains, Iraqi Arabic/English and Spanish/English bidirectional translation. For Iraqi Arabic/English, the parallel training corpus consists of 430k utterance pairs containing 98k unique Arabic words and 31k unique English words. The Arabic LM training data is composed of the 2.7M words of Arabic in the parallel training corpus. Applying a morphological segmentation algorithm (Afify et al., 2006) to the Iraqi Arabic training data results in 58k unique morphemes and 2.8M morpheme tokens. For English, we use 6.4M words of text, of which the English data in the MT training corpus is a subset. For English to Arabic, we have a development set of 19k words (2.2k sentences) to tune feature weights, and a test set of about the same size. For Arabic to English, the development and test sets are about 21k words (2.9k sentences).

	50-best	20-best	10-best
English ⇒ Iraqi Arabic			
N-best oracle	38.7/37.4	36.4/34.9	34.0/32.6
3-gram	30.6/29.7	30.6/29.7	30.6/29.7
Model M	31.0/30.3	31.1/30.4	31.1/30.2
3-gram + Model M	31.0/30.5	31.0/30.3	31.0/30.2
Iraqi Arabic ⇒ English			
N-best oracle	36.5/36.3	34.4/34.1	32.5/32.3
3-gram	25.4/24.7	25.4/24.7	25.4/24.7
Model M	24.9/26.0	25.1/25.8	25.3/25.9
3-gram + Model M	25.5/26.1	25.4/26.0	25.5/26.0
English ⇒ Spanish			
N-best oracle	34.8/36.1	32.7/34.2	30.8/31.7
4-gram	21.7/21.5	21.7/21.5	21.7/21.2
Model M	22.8/23.0	22.7/23.0	22.8/22.8
4-gram + Model M	22.6/22.9	22.5/22.8	22.8/22.8
Spanish ⇒ English			
N-best oracle	32.1/32.3	29.4/29.2	26.9/26.1
4-gram	18.1/17.6	18.3/17.6	18.0/17.6
Model M	19.6/18.4	19.3/18.8	19.1/18.3
4-gram + Model M	19.4/18.5	19.2/18.8	19.0/18.4

Table 7: BLEU scores for various language models for Iraqi Arabic/English and Spanish/English translation, using N-best list rescoring of N-best lists of various size. For each model, we report (development set/test set) results.

For Spanish/English, the target task is a travel application. The MT training data consists of conversational travel data as well as movie subtitles and TV show transcriptions, 2.1M sentence pairs in all with 14.3M English tokens (137k unique) and 13.5M Spanish tokens (176k unique). The MT training data is also used for language model training. The test and development sets consist of 711 sentence pairs each, with about 5.9k English and 5.6k Spanish tokens in each. The machine translation models are built according to a commonly used recipe: word alignment models are trained in both translation directions using parallel sentence pairs, and two sets of Viterbi alignments are derived. The alignments from both directions are combined in a heuristic fashion to form a single alignment (Och and Ney, 2003). All phrase pairs (up to a maximum length of six words) that satisfy a word alignment boundary constraint are identified and included in the phrase translation table, and translation parameters are computed using maximum likelihood estimation. We use a phrase-based multi-stack decoder using log-linear models similar to Pharaoh (Koehn et al., 2003). As in most maximum-entropy-based decoders, we include features for bidirectional translation probabilities, bidirectional lexicon weights, language model scores, distortion model scores, and a sentence length penalty.

To evaluate Model M, we do N-best list rescoring and measure translation performance using BLEU score (Papineni et al., 2002) with one reference for each hypothesis. The baseline language

model is a conventional n-gram model, and this baseline model is used to generate translation N-best lists of various size (N=10, 20, and 50). Feature weights (including the language model weight) are optimized on the development data using the downhill simplex method (Och and Ney, 2001) to maximize BLEU score. In addition to the baseline, we evaluate Model M as well as Model M interpolated with the baseline n-gram model. For Model M and the interpolated model, we take the feature weights from the baseline model and re-optimize only the language model weight using a local line search. (Retuning all weights may improve results further.) For Arabic/English, we use 1-best weight tuning while for Spanish/English, N-best tuning is used. We have not observed significant performance differences between the two strategies. For Arabic/English, the trigram versions of each model are used due to the small amount of training data, over morphemes for Iraqi Arabic and over words for English. For Spanish/English, 4-gram versions of each model are used.

In Table 7, we display the BLEU scores for each model as well as the oracle BLEU scores for each different N-best list size, for both the development and test sets. We see consistent gains in test set BLEU scores across all conditions for Model M as compared to the baseline, with gains ranging from 0.5 to 1.6 points. Interpolating Model M with the baseline gives about the same performance as Model M alone, indicating that Model M already encompasses most or all of the information included in an n-gram model.

7 Discussion

We show that Model M consistently outperforms the best existing language models over a variety of domains and applications. While our analysis shows that shrinkage-based gains will decrease as training sets increase in size, we still find significant gains even on tasks where over a billion words of training data are available. We achieve WER gains of 0.5–0.7% absolute for three large-scale ASR systems, including state-of-the-art systems on the highly competitive English Broadcast News and GALE Arabic tasks. While other language modeling methods have provided gains over interpolations of word 4-gram models at smaller scale, we are unaware of any techniques that give nearly this much gain on systems of this scale (and which do not require interpolation with a word n-gram model). On the other hand, while rMDI models can give gains against other techniques for domain adaptation on moderately-sized corpora, it does not outperform linear interpolation on large data sets or on general model combination tasks, especially in conjunction with Model M.

The performance gains from Model M come at a computational cost. While Model M is smaller in parameter mass relative to a word n-gram model trained on the same corpus, it has many more parameters. For 4-gram models built on 900k sentences of WSJ data, Model M has about twice as many parameters as a word n-gram model (49M *vs.* 25M). In addition, training and probability evaluation are much slower as compared to n-gram models. However, training for exponential models can be efficiently parallelized, *e.g.*, (Rosenfeld, 1996), and for unnormalized models, a single probability lookup can be implemented via three n-gram model lookups.

In summary, despite the advances in language modeling over the past decades, word n-gram models remain the technology of choice in systems both large and small. Here, we show that Model M is a compelling alternative for a wide range of applications and operating points.

Acknowledgments

The authors would like to thank DARPA for funding part of this work under Grant HR0011-06-2-0001.

References

Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. 2006. On the use of morphological analysis for dialectal arabic speech recognition. In *Proceedings of Interspeech*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-CoNLL*, pages 858–867.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Stanley F. Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for maximum entropy models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Stanley F. Chen, Brian Kingsbury, Lidia Mangu, Daniel Povey, George Saon, Hagen Soltau, and Geoffrey Zweig. 2006. Advances in speech transcription at IBM under the DARPA EARS program. *IEEE Transactions on Audio, Speech and Language Processing*, pages 1596–1608.

Stanley F. Chen. 2008. Performance prediction for exponential language models. Technical Report RC 24671, IBM Research Division, October.

Stanley F. Chen. 2009a. Performance prediction for exponential language models. In *Proceedings of NAACL-HLT*.

Stanley F. Chen. 2009b. Shrinking exponential language models. In *Proceedings of NAACL-HLT*.

J.N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480.

Stephen Della Pietra, Vincent Della Pietra, Robert L. Mercer, and Salim Roukos. 1992. Adaptive language modeling using minimum discriminant estimation. In *Proceedings of the Speech and Natural Language DARPA Workshop*, February.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April.

Marcello Federico. 1996. Bayesian estimation methods for n-gram language model adaptation. In *Proceedings of ICSLP*, pages 240–243.

M. J. F. Gales, Do Yeong Kim, P. C. Woodland, Ho Yin Chan, D. Mrva, R. Sinha, and S. E. Tranter. 2006. Progress in the CU-HTK broadcast news transcription system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1513–1525, September.

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Proceedings of ACL*.

Frederick Jelinek, Bernard Merialdo, Salim Roukos, and Martin Strauss. 1991. A dynamic language model for speech recognition. In *Proceedings of the DARPA Workshop on Speech and Natural Language*, pages 293–295, Morristown, NJ, USA. Association for Computational Linguistics.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of EMNLP*, pages 137–144.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

J.D. Lafferty and B. Suhm. 1995. Cluster expansions and iterative scaling for maximum entropy language models. In K. Hanson and R. Silver, editors, *Maximum Entropy and Bayesian Methods*, pages 195–202. Kluwer Academic Publishers.

Guy Lebanon and John Lafferty. 2001. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems*, pages 447–454.

Hirokazu Masataki, Yoshinori Sagisaka, Kazuya Hisaki, and Tatsuya Kawahara. 1997. Task adaptation using MAP estimation in n-gram language modeling. In *Proceedings of ICASSP*, volume 2, pages 783–786, Washington, DC, USA. IEEE Computer Society.

Franz Josef Och and Hermann Ney. 2001. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295–302, Morristown, NJ, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Daniel Povey, Stephen M. Chu, and Balakrishnan Varadarajan. 2008. Universal background model based speech recognition. In *Proceedings of ICASSP*.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228.

George Saon, Daniel Povey, and Geoffrey Zweig. 2005. Anatomy of an extremely fast LVCSR decoder. In *Proceedings of Interspeech*, pages 549–552.

K. Seymore and R. Rosenfeld. 1997. Using story topics for language model adaptation. In *Proceedings of Eurospeech*.

Hagen Soltau, Brian Kingsbury, Lidia Mangu, Daniel Povey, George Saon, and Geoffrey Zweig. 2005. The IBM 2004 conversational telephony system for rich transcription. In *Proceedings of ICASSP*, pages 205–208.

Hagen Soltau, George Saon, Brian Kingsbury, Jeff Kuo, Lidia Mangu, Daniel Povey, and Geoffrey Zweig. 2007. The IBM 2006 GALE Arabic ASR system. In *Proceedings of ICASSP*.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274, Lansdowne, VA, February.