Shrinking Exponential Language Models

Stanley F. Chen
IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
stanchen@watson.ibm.com

Abstract

In (Chen, 2009), we show that for a variety of language models belonging to the exponential family, the test set cross-entropy of a model can be accurately predicted from its training set cross-entropy and its parameter values. In this work, we show how this relationship can be used to motivate two heuristics for "shrinking" the size of a language model to improve its performance. We use the first heuristic to develop a novel class-based language model that outperforms a baseline word trigram model by 28% in perplexity and 1.9% absolute in speech recognition word-error rate on Wall Street Journal data. We use the second heuristic to motivate a regularized version of minimum discrimination information models and show that this method outperforms other techniques for domain adaptation.

1 Introduction

An exponential model $p_\Lambda(y|x)$ is a model with a set of features $\{f_1(x,y),\ldots,f_F(x,y)\}$ and equal number of parameters $\Lambda = \{\lambda_1,\ldots,\lambda_F\}$ where

$$p_\Lambda(y|x) = \frac{\exp(\sum_{i=1}^F \lambda_i f_i(x,y))}{Z_\Lambda(x)} \quad (1)$$

and where $Z_\Lambda(x)$ is a normalization factor. In (Chen, 2009), we show that for many types of exponential language models, if a training and test set are drawn from the same distribution, we have

$$\mathcal{H}_{\text{test}} \approx \mathcal{H}_{\text{train}} + \frac{\gamma}{D}\sum_{i=1}^F |\tilde{\lambda}_i| \quad (2)$$

where $\mathcal{H}_{\text{test}}$ denotes test set cross-entropy; $\mathcal{H}_{\text{train}}$ denotes training set cross-entropy; D is the number of events in the training data; the $\tilde{\lambda}_i$ are *regularized* parameter estimates; and γ is a constant independent of domain, training set size, and model type.[1] This relationship is strongest if the $\tilde{\Lambda} = \{\tilde{\lambda}_i\}$ are estimated using $\ell_1+\ell_2^2$ regularization (Kazama and Tsujii, 2003). In $\ell_1 + \ell_2^2$ regularization, parameters are chosen to optimize

$$\mathcal{O}_{\ell_1+\ell_2^2}(\Lambda) = \mathcal{H}_{\text{train}} + \frac{\alpha}{D}\sum_{i=1}^F |\lambda_i| + \frac{1}{2\sigma^2 D}\sum_{i=1}^F \lambda_i^2 \quad (3)$$

for some α and σ. With $(\alpha = 0.5, \sigma^2 = 6)$ and taking $\gamma = 0.938$, test set cross-entropy can be predicted with eq. (2) for a wide range of models with a mean error of a few hundredths of a nat, equivalent to a few percent in perplexity.[2]

In this paper, we show how eq. (2) can be applied to improve language model performance. First, we use eq. (2) to analyze backoff features in exponential n-gram models. We find that backoff features improve test set performance by reducing the "size" of a model $\frac{1}{D}\sum_{i=1}^F |\tilde{\lambda}_i|$ rather than by improving training set performance. This suggests the following principle for improving exponential language models: if a model can be "shrunk" without increasing its training set cross-entropy, test set cross-entropy should improve. We apply this idea to motivate two language models: a novel class-based language model and regularized minimum discrimination information (MDI) models. We show how these models outperform other models in both perplexity and word-error rate on Wall Street Journal (WSJ) data.

The organization of this paper is as follows: In Section 2, we analyze the use of backoff features in n-gram models to motivate a heuristic for model design. In Sections 3 and 4, we introduce our novel

[1] The cross-entropy of a model $p_\Lambda(y|x)$ on some data $\mathcal{D} = (x_1,y_1),\ldots,(x_D,y_D)$ is defined as $-\frac{1}{D}\sum_{j=1}^D \log p_\Lambda(y_j|x_j)$. It is equivalent to the negative mean log-likelihood per event as well as to log perplexity.

[2] A *nat* is a "natural" bit and is equivalent to $\log_2 e$ regular bits. We use nats to be consistent with (Chen, 2009).

features	$\mathcal{H}_{\text{eval}}$	$\mathcal{H}_{\text{pred}}$	$\mathcal{H}_{\text{train}}$	$\sum \frac{\lvert\tilde{\lambda}_i\rvert}{D}$
3g	2.681	2.724	2.341	0.408
2g+3g	2.528	2.513	2.248	0.282
1g+2g+3g	2.514	2.474	2.241	0.249

Table 1: Various statistics for letter trigram models built on a 1k-word training set. $\mathcal{H}_{\text{eval}}$ is the cross-entropy of the evaluation data; $\mathcal{H}_{\text{pred}}$ is the predicted test set cross-entropy according to eq. (2); and $\mathcal{H}_{\text{train}}$ is the training set cross-entropy. The evaluation data is drawn from the same distribution as the training; $\mathcal{H}$ values are in nats.

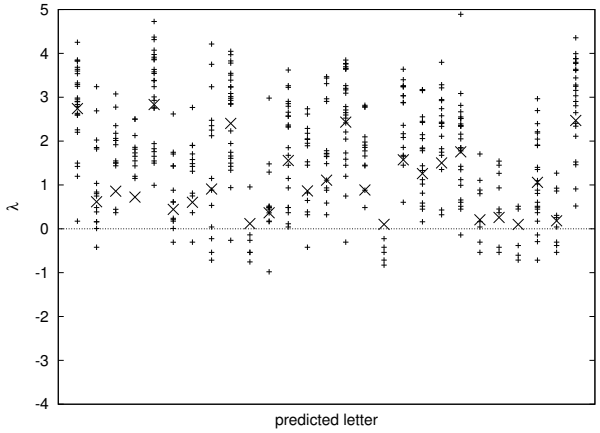

Figure 1: Nonzero $\tilde{\lambda}_i$ values for bigram features in letter bigram model without unigram backoff features. If we denote bigrams as $w_{j-1}w_j$, each column contains the $\tilde{\lambda}_i$'s corresponding to all bigrams with a particular w_j. The '$\times$' marks represent the average $|\tilde{\lambda}_i|$ in each column; this average includes history words for which no feature exists or for which $\tilde{\lambda}_i = 0$.

class-based model and discuss MDI domain adaptation, and compare these methods against other techniques on WSJ data. Finally, in Sections 5 and 6 we discuss related work and conclusions.[3]

2 N-Gram Models and Backoff Features

In this section, we use eq. (2) to explain why backoff features in exponential n-gram models improve performance, and use this analysis to motivate a general heuristic for model design. An exponential n-gram model contains a binary feature f_ω for each n'-gram ω occurring in the training data for $n' \leq n$, where $f_\omega(x, y) = 1$ iff xy ends in ω. We refer to features corresponding to n'-grams for $n' < n$ as *backoff* features; it is well known that backoff features help

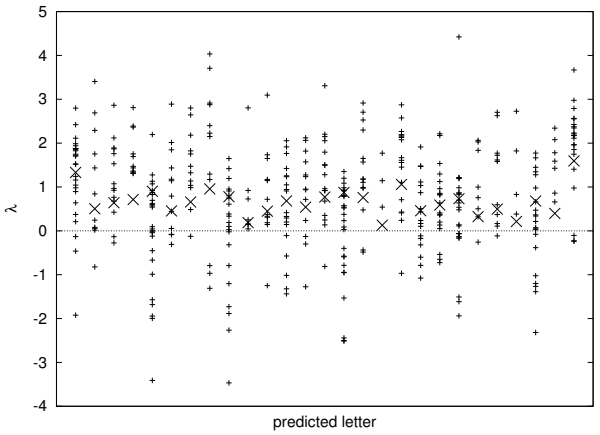

Figure 2: Like Figure 1, but for model with unigram backoff features.

performance a great deal. We present statistics in Table 1 for various letter trigram models built on the same data set. In these and all later experiments, all models are regularized with $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$. The last row corresponds to a normal trigram model; the second row corresponds to a model lacking unigram features; and the first row corresponds to a model with no unigram or bigram features. As backoff features are added, we see that the training set cross-entropy improves, which is not surprising since the number of features is increasing. More surprising is that as we add features, the "size" of the model $\frac{1}{D}\sum_{i=1}^{F} |\tilde{\lambda}_i|$ decreases.

We can explain these results by examining a simple example. Consider an exponential model consisting of the features $f_1(x, y)$ and $f_2(x, y)$ with parameter values $\tilde{\lambda}_1 = 3$ and $\tilde{\lambda}_2 = 4$. From eq. (1), this model has the form

$$p_{\tilde{\Lambda}}(y|x) = \frac{\exp(3f_1(x, y) + 4f_2(x, y))}{Z_\Lambda(x)} \qquad (4)$$

Now, consider creating a new feature $f_3(x, y) = f_1(x, y) + f_2(x, y)$ and setting our parameters as follows: $\lambda_1^{\text{new}} = 0$, $\lambda_2^{\text{new}} = 1$, and $\lambda_3^{\text{new}} = 3$. Substituting into eq. (1), we see that $p_{\Lambda^{\text{new}}}(y|x) = p_{\tilde{\Lambda}}(y|x)$ for all x, y. As the distribution this model describes does not change, neither will its training performance. However, the (unscaled) size $\sum_{i=1}^{F} |\lambda_i|$ of the model has been reduced from 3+4=7 to 0+1+3=4, and consequently by eq. (2) we predict that test performance will improve.[4]

[3] A long version of this paper can be found at (Chen, 2008).

[4] When $\text{sgn}(\tilde{\lambda}_1) = \text{sgn}(\tilde{\lambda}_2)$, $\sum_{i=1}^{F} |\lambda_i|$ is reduced most by

In fact, since $p_{\Lambda^{\text{new}}} = p_{\tilde{\Lambda}}$, test performance will remain the same. The catch is that eq. (2) applies only to the *regularized* parameter estimates for a model, and in general, Λ^{new} will not be the regularized parameter estimates for the expanded feature set. We can compute the actual regularized parameters $\tilde{\Lambda}^{\text{new}}$ for which eq. (2) *will* apply; this may improve predicted performance even more.

Hence, by adding "redundant" features to a model to shrink its total size $\sum_{i=1}^{F} |\tilde{\lambda}_i|$, we can improve predicted performance (and perhaps also actual performance). This analysis suggests the following technique for improving model performance:

Heuristic 1 *Identify groups of features which will tend to have similar $\tilde{\lambda}_i$ values. For each such feature group, add a new feature to the model that is the sum of the original features.*

The larger the original $\tilde{\lambda}_i$'s, the larger the reduction in model size and the higher the predicted gain.

Given this perspective, we can explain why backoff features improve n-gram model performance. For simplicity, consider a bigram model, one without unigram backoff features. It seems intuitive that probabilities of the form $p(w_j|w_{j-1})$ are similar across different w_{j-1}, and thus so are the $\tilde{\lambda}_i$ for the corresponding bigram features. (If a word has a high unigram probability, it will also tend to have high bigram probabilities.) In Figure 1, we plot the nonzero $\tilde{\lambda}_i$ values for all (bigram) features in a bigram model without unigram features. Each column contains the $\tilde{\lambda}_i$ values for a different predicted word w_j, and the '$\times$' mark in each column is the average value of $|\tilde{\lambda}_i|$ over all history words w_{j-1}. We see that the average $|\tilde{\lambda}_i|$ for each word w_j is often quite far from zero, which suggests creating features

$$f_{w_j}(x, y) = \sum_{w_{j-1}} f_{w_{j-1}w_j}(x, y) \tag{5}$$

to reduce the overall size of the model.

In fact, these features are exactly unigram backoff features. In Figure 2, we plot the nonzero $\tilde{\lambda}_i$ values for all bigram features after adding unigram backoff features. We see that the average $|\tilde{\lambda}_i|$'s are closer to zero, implying that the model size $\sum_{i=1}^{F} |\tilde{\lambda}_i|$ has been significantly decreased. We can extend this idea to higher-order n-gram models as well; *e.g.*, bigram parameters can shrink trigram parameters, and can in turn be shrunk by unigram parameters. As shown in Table 1, both training set cross-entropy and model size can be reduced by this technique.

setting λ_3^{new} to the $\tilde{\lambda}_i$ with the smaller magnitude, and the size of the reduction is equal to $|\lambda_3^{\text{new}}|$. If $\text{sgn}(\tilde{\lambda}_1) \neq \text{sgn}(\tilde{\lambda}_2)$, no reduction is possible through this transformation.

	$\mathcal{H}_{\text{eval}}$	$\mathcal{H}_{\text{pred}}$	$\mathcal{H}_{\text{train}}$	$\sum \frac{\lvert\tilde{\lambda}_i\rvert}{D}$
word n-gram	4.649	4.672	3.354	1.405
model **M**	4.536	4.544	3.296	1.330

Table 2: Various statistics for word and class trigram models built on 100k sentences of WSJ training data.

3 Class-Based Language Models

In this section, we show how we can use Heuristic 1 to design a novel class-based model that outperforms existing models in both perplexity and speech recognition word-error rate. We assume a word w is always mapped to the same class $c(w)$. For a sentence $w_1 \cdots w_l$, we have

$$p(w_1 \cdots w_l) = \prod_{j=1}^{l+1} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) \times \prod_{j=1}^{l} p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) \tag{6}$$

where $c_j = c(w_j)$ and c_{l+1} is the end-of-sentence token. We use the notation $p_{\text{ng}}(y|\omega)$ to denote an exponential n-gram model, a model containing a feature for each suffix of each ωy occurring in the training set. We use $p_{\text{ng}}(y|\omega_1, \omega_2)$ to denote a model containing all features in $p_{\text{ng}}(y|\omega_1)$ and $p_{\text{ng}}(y|\omega_2)$.

We can define a class-based n-gram model by choosing parameterizations for the distributions $p(c_j|\cdots)$ and $p(w_j|\cdots)$ in eq. (6) above. For example, the most widely-used class-based n-gram model is the one introduced by Brown et al. (1992); we refer to this model as the IBM class model:

$$\begin{aligned} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(c_j|c_{j-2}c_{j-1}) \\ p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(w_j|c_j) \end{aligned} \tag{7}$$

(In the original work, non-exponential n-gram models are used.) Clearly, there is a large space of possible class-based models.

Now, we discuss how we can use Heuristic 1 to design a novel class-based model by using class information to "shrink" a word-based n-gram model. The basic idea is as follows: if we have an n-gram ω

and another n-gram ω' created by replacing a word in ω with a similar word, then the two corresponding features should have similar $\tilde{\lambda}_i$'s. For example, it seems intuitive that the n-grams *on Monday morning* and *on Tuesday morning* should have similar $\tilde{\lambda}_i$'s. Heuristic 1 tells us how to take advantage of this observation to improve model performance.

Let's begin with a word trigram model $p_{\text{ng}}(w_j|w_{j-2}w_{j-1})$. First, we would like to convert this model into a class-based model. Without loss of generality, we have

$$\begin{aligned} p(w_j|w_{j-2}w_{j-1}) &= \textstyle\sum_{c_j} p(w_j, c_j|w_{j-2}w_{j-1}) \\ &= \textstyle\sum_{c_j} p(c_j|w_{j-2}w_{j-1})p(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \quad (8)$$

Thus, it seems reasonable to use the distributions $p_{\text{ng}}(c_j|w_{j-2}w_{j-1})$ and $p_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j)$ as the starting point for our class model. This model can express the same set of word distributions as our original model, and hence may have a similar training cross-entropy. In addition, this transformation can be viewed as shrinking together word n-grams that differ only in w_j. That is, we expect that pairs of n-grams $w_{j-2}w_{j-1}w_j$ that differ only in w_j (belonging to the same class) should have similar $\tilde{\lambda}_i$. From Heuristic 1, we can make new features

$$f_{w_{j-2}w_{j-1}c_j}(x, y) = \sum_{w_j \in c_j} f_{w_{j-2}w_{j-1}w_j}(x, y) \quad (9)$$

These are exactly the features in $p_{\text{ng}}(c_j|w_{j-2}w_{j-1})$. When applying Heuristic 1, all features typically belong to the same model, but even when they don't one can achieve the same net effect.

Then, we can use Heuristic 1 to also shrink together n-gram features for n-grams that differ only in their histories. For example, we can create new features of the form

$$f_{c_{j-2}c_{j-1}c_j}(x, y) = \sum_{w_{j-2} \in c_{j-2}, w_{j-1} \in c_{j-1}} f_{w_{j-2}w_{j-1}c_j}(x, y) \quad (10)$$

This corresponds to replacing $p_{\text{ng}}(c_j|w_{j-2}w_{j-1})$ with the distribution $p_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1})$. We refer to the resulting model as model **M**:

$$\begin{aligned} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \\ p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \quad (11)$$

By design, it is meant to have similar training set cross-entropy as a word n-gram model while being significantly smaller.

To give an idea of whether this model behaves as expected, in Table 2 we provide statistics for this model (as well as for an exponential word n-gram model) built on 100k WSJ training sentences with 50 classes using the same regularization as before. We see that model **M** is both smaller than the baseline and has a lower training set cross-entropy, similar to the behavior found when adding backoff features to word n-gram models in Section 2. As long as eq. (2) holds, model **M** should have good test performance; in (Chen, 2009), we show that eq. (2) does indeed hold for models of this type.

3.1 Class-Based Model Comparison

In this section, we compare model **M** against other class-based models in perplexity and word-error rate. The training data is 1993 WSJ text with verbalized punctuation from the CSR-III Text corpus, and the vocabulary is the union of the training vocabulary and 20k-word "closed" test vocabulary from the first WSJ CSR corpus (Paul and Baker, 1992). We evaluate training set sizes of 1k, 10k, 100k, and 900k sentences. We create three different word classings containing 50, 150, and 500 classes using the algorithm of Brown et al. (1992) on the largest training set.[5] For each training set and number of classes, we build 3-gram and 4-gram versions of each model.

From the verbalized punctuation data from the training and test portions of the WSJ CSR corpus, we randomly select 2439 unique utterances (46888 words) as our evaluation set. From the remaining verbalized punctuation data, we select 977 utterances (18279 words) as our development set.

We compare the following model types: conventional (*i.e.*, non-exponential) word n-gram models; conventional IBM class n-gram models interpolated with conventional word n-gram models (Brown et al., 1992); and model **M**. All conventional n-gram models are smoothed with modified Kneser-Ney smoothing (Chen and Goodman, 1998), except we also evaluate word n-gram models with Katz smoothing (Katz, 1987). *Note:* Because word

[5] One can imagine choosing word classes to optimize model shrinkage; however, this is not an avenue we pursued.

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, Katz				
3g	**579.3**	**317.1**	**196.7**	137.5
4g	592.6	325.6	202.4	**136.7**
interpolated IBM class model				
3g, 50c	358.4	224.5	156.8	117.8
3g, 150c	346.5	210.5	149.0	114.7
3g, 500c	372.6	210.9	145.8	112.3
4g, 50c	362.1	220.4	149.6	109.1
4g, 150c	**346.3**	**207.8**	142.5	105.2
4g, 500c	371.5	207.9	**140.5**	**103.6**

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, modified KN				
3g	488.4	270.6	168.2	121.5
4g	**486.8**	**267.4**	**163.6**	**114.4**
model **M**				
3g, 50c	**341.5**	210.0	144.5	110.9
3g, 150c	342.6	203.7	140.0	108.0
3g, 500c	387.5	212.7	142.2	108.1
4g, 50c	345.8	209.0	139.1	101.6
4g, 150c	344.1	**202.8**	**135.7**	**99.1**
4g, 500c	390.7	211.1	138.5	100.6

Table 3: WSJ perplexity results. The best performance for each training set for each model type is highlighted in bold.

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, Katz				
3g	**35.5%**	**30.7%**	**26.2%**	**22.7%**
4g	35.6%	30.9%	26.3%	**22.7%**
interpolated IBM class model				
3g, 50c	32.2%	28.7%	25.2%	22.5%
3g, 150c	**31.8%**	28.1%	25.0%	22.3%
3g, 500c	32.5%	28.5%	**24.5%**	22.1%
4g, 50c	32.2%	28.6%	25.0%	22.0%
4g, 150c	**31.8%**	**28.0%**	24.6%	21.8%
4g, 500c	32.7%	28.3%	**24.5%**	**21.6%**

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, modified KN				
3g	**34.5%**	30.5%	26.1%	22.6%
4g	**34.5%**	**30.4%**	**25.7%**	**22.3%**
model **M**				
3g, 50c	**30.8%**	27.4%	24.0%	21.7%
3g, 150c	31.0%	**27.1%**	23.8%	21.5%
3g, 500c	32.3%	27.8%	23.9%	21.4%
4g, 50c	**30.8%**	27.5%	23.9%	21.2%
4g, 150c	31.0%	**27.1%**	**23.5%**	**20.8%**
4g, 500c	32.4%	27.9%	24.1%	21.1%

Table 4: WSJ lattice rescoring results; all values are word-error rates. The best performance for each training set size for each model type is highlighted in bold. Each 0.1% in error rate corresponds to about 47 errors.

classes are derived from the largest training set, results for word models and class models are comparable only for this data set. The interpolated model is the most popular state-of-the-art class-based model in the literature, and is the only model here using the development set to tune interpolation weights.

We display the perplexities of these models on the evaluation set in Table 3. Model **M** performs best of all (even without interpolating with a word n-gram model), outperforming the interpolated model with every training set and achieving its largest reduction in perplexity (4%) on the largest training set. While these perplexity reductions are quite modest, what matters more is speech recognition performance.

For the speech recognition experiments, we use a cross-word quinphone system built from 50 hours of Broadcast News data. The system contains 2176 context-dependent states and a total of 50336 Gaussians. To evaluate our language models, we use lattice rescoring. We generate lattices on both our development and evaluation data sets using the Latt-AIX 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 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.

In Table 4, we display the word-error rates for each model. If we compare the best performance of model **M** for each training set with that of the state-of-the-art interpolated class model, we find that model **M** is superior by 0.8–1.0% absolute. These gains are much larger than are suggested by the perplexity gains of model **M** over the interpolated model; as has been observed earlier, perplexity is

	$\mathcal{H}_{\text{eval}}$	$\mathcal{H}_{\text{pred}}$	$\mathcal{H}_{\text{train}}$	$\sum \frac{\lvert\tilde{\lambda}_i\rvert}{D}$
baseline n-gram model				
1k	5.915	5.875	2.808	3.269
10k	5.212	5.231	3.106	2.265
100k	4.649	4.672	3.354	1.405
MDI n-gram model				
1k	5.444	5.285	2.678	2.780
10k	5.031	4.973	3.053	2.046
100k	4.611	4.595	3.339	1.339

Table 5: Various statistics for WSJ trigram models, with and without a Broadcast News prior model. The first column is the size of the in-domain training set in sentences.

not a reliable predictor of speech recognition performance. While we can only compare class models with word models on the largest training set, for this training set model **M** outperforms the baseline Katz-smoothed word trigram model by 1.9% absolute.[6]

4 Domain Adaptation

In this section, we introduce another heuristic for improving exponential models and show how this heuristic can be used to motivate a regularized version of minimum discrimination information (MDI) models (Della Pietra et al., 1992). Let's say we have a model $p_{\tilde{\Lambda}}$ estimated from one training set and a "similar" model q estimated from an independent training set. Imagine we use q as a *prior* model for p_Λ; *i.e.*, we make a new model $p^q_{\Lambda^{\text{new}}}$ as follows:

$$p^q_{\Lambda^{\text{new}}}(y|x) = q(y|x)\frac{\exp(\sum_{i=1}^F \lambda_i^{\text{new}} f_i(x,y))}{Z_{\Lambda^{\text{new}}}(x)} \quad (12)$$

Then, choose Λ^{new} such that $p^q_{\Lambda^{\text{new}}}(y|x) = p_{\tilde{\Lambda}}(y|x)$ for all x, y (assuming this is possible). If q is "similar" to $p_{\tilde{\Lambda}}$, then we expect the size $\frac{1}{D}\sum_{i=1}^F |\lambda_i^{\text{new}}|$ of $p^q_{\Lambda^{\text{new}}}$ to be less than that of $p_{\tilde{\Lambda}}$. Since they describe the same distribution, their training set cross-entropy will be the same. By eq. (2), we expect $p^q_{\Lambda^{\text{new}}}$ to have better test set performance than $p_{\tilde{\Lambda}}$ after reestimation.[7] In (Chen, 2009), we show that eq. (2) does indeed hold for models with priors; q need not be accounted for in computing model size as long as it is estimated on a separate training set.

[6] Results for several other baseline language models and with a different acoustic model are given in (Chen, 2008).

[7] That is, we expect the *regularized* parameters $\tilde{\Lambda}^{\text{new}}$ to yield improved performance.

This analysis suggests the following method for improving model performance:

Heuristic 2 *Find a "similar" distribution estimated from an independent training set, and use this distribution as a prior.*

It is straightforward to apply this heuristic to the task of domain adaptation for language modeling. In the usual formulation of this task, we have a test set and a small training set from the same domain, and a large training set from a different domain. The goal is to use the data from the outside domain to maximally improve language modeling performance on the target domain. By Heuristic 2, we can build a language model on the outside domain, and use this model as the prior model for a language model built on the in-domain data. This method is identical to the MDI method for domain adaptation, except that we also apply regularization.

In our domain adaptation experiments, our out-of-domain data is a 100k-sentence Broadcast News training set. For our in-domain WSJ data, we use training set sizes of 1k, 10k, and 100k sentences. We build an exponential n-gram model on the Broadcast News data and use this model as the prior model $q(y|x)$ in eq. (12) when building an exponential n-gram model on the in-domain data. In Table 5, we display various statistics for trigram models built on varying amounts of in-domain data when using a Broadcast News prior and not. Across training sets, the MDI models are both smaller in $\frac{1}{D}\sum_{i=1}^F |\tilde{\lambda}_i|$ and have better training set cross-entropy than the unadapted models built on the same data. By eq. (2), the adapted models should have better test performance and we verify this in the next section.

4.1 Domain Adaptation Method Comparison

In this section, we examine how MDI adaptation compares to other state-of-the-art methods for domain adaptation in both perplexity and speech recognition word-error rate. For these experiments, we use the same development and evaluation sets and lattice rescoring setup from Section 3.1.

The most widely-used techniques for domain adaptation are linear interpolation and count merging. In linear interpolation, separate n-gram models are built on the in-domain and out-of-domain data and are interpolated together. In count merging, the

	in-domain data (sents.)			in-domain data (sents.)		
	1k	10k	100k	1k	10k	100k
in-domain data only						
3g	488.4	270.6	168.2	**34.5%**	30.5%	26.1%
4g	**486.8**	**267.4**	**163.6**	**34.5%**	**30.4%**	**25.7%**
count merging						
3g	503.1	290.9	170.7	30.4%	28.3%	**25.2%**
4g	**497.1**	**284.9**	**165.3**	**30.0%**	**28.0%**	25.3%
linear interpolation						
3g	328.3	234.8	162.6	**30.3%**	28.5%	25.8%
4g	**325.3**	**230.8**	**157.6**	**30.3%**	**28.4%**	**25.2%**
MDI model						
3g	296.3	218.7	157.0	30.0%	28.0%	**24.9%**
4g	**293.7**	**215.8**	**152.5**	**29.6%**	**27.9%**	**24.9%**

Table 6: WSJ perplexity and lattice rescoring results for domain adaptation models. Values on the left are perplexities and values on the right are word-error rates.

in-domain and out-of-domain data are concatenated into a single training set, and a single n-gram model is built on the combined data set. The in-domain data set may be replicated several times to more heavily weight this data. We also consider the baseline of not using the out-of-domain data.

In Table 6, we display perplexity and word-error rates for each method, for both trigram and 4-gram models and with varying amounts of in-domain training data. The last method corresponds to the exponential MDI model; all other methods employ conventional (non-exponential) n-gram models with modified Kneser-Ney smoothing. In count merging, only one copy of the in-domain data is included in the training set; including more copies does not improve evaluation set word-error rate.

Looking first at perplexity, MDI models outperform the next best method, linear interpolation, by about 10% in perplexity on the smallest data set and 3% in perplexity on the largest. In terms of word-error rate, MDI models again perform best of all, outperforming interpolation by 0.3–0.7% absolute and count merging by 0.1–0.4% absolute.

5 Related Work

5.1 Class-Based Language Models

In past work, the most common baseline models are Katz-smoothed word trigram models. Compared to this baseline, model **M** achieves a perplexity reduction of 28% and word-error rate reduction of 1.9% absolute with a 900k-sentence training set. The most closely-related existing model to model **M** is the model *fullibmpredict* proposed by Goodman (2001):

$$\begin{aligned} &p(c_j|c_{j-2}c_{j-1},w_{j-2}w_{j-1})= \\ &\qquad \lambda\, p(c_j|w_{j-2}w_{j-1})+(1-\lambda)\, p(c_j|c_{j-2}c_{j-1}) \\ &p(w_j|c_{j-2}c_{j-1}c_j,w_{j-2}w_{j-1})= \\ &\qquad \mu\, p(w_j|w_{j-2}w_{j-1}c_j)+(1-\mu)\, p(w_j|c_{j-2}c_{j-1}c_j) \end{aligned} \quad (13)$$

This is similar to model **M** except that linear interpolation is used to combine word and class history information, and there is no analog to the final term in eq. (13) in model **M**. Using the North American Business news corpus, the largest perplexity reduction achieved over a Katz-smoothed trigram model baseline by *fullibmpredict* is about 25%, with a training set of 1M words. In N-best list rescoring with a 284M-word training set, the best result achieved for an individual class-based model is an 0.5% absolute reduction in word-error rate.

To situate the quality of our results, we also review the best perplexity and word-error rate results reported for class-based language models relative to conventional word n-gram model baselines. In terms of absolute word-error rate, the best gains we found in the literature are from *multi-class composite* n-gram models, a variant of the IBM class model (Yamamoto and Sagisaka, 1999; Yamamoto et al., 2003). These are called *composite* models because frequent word sequences can be concatenated into single units within the model; the term *multi-class* refers to choosing different word clusterings depending on word position. In experiments on the ATR spoken language database, Yamamoto et al. (2003) report a reduction in perplexity of 9% and an increase in word accuracy of 2.2% absolute over a Katz-smoothed trigram model.

In terms of perplexity, the best gains we found are from SuperARV language models (Wang and Harper, 2002; Wang et al., 2002; Wang et al., 2004). In these models, classes are based on *abstract role values* as given by a Constraint Dependency Grammar. The class and word prediction distributions are n-gram models that back off to a variety of mixed word/class histories in a specific order. With a WSJ training set of 37M words and a Katz-smoothed trigram model baseline, a perplexity reduction of up to

53% is achieved as well as a decrease in word-error rate of up to 1.0% absolute.

All other perplexity and absolute word-error rate gains we found in the literature are considerably smaller than those listed here. While different data sets are used in previous work so results are not directly comparable, our results appear very competitive with the body of existing results in the literature.

5.2 Domain Adaptation

Here, we discuss methods for supervised domain adaptation that involve only the simple static combination of in-domain and out-of-domain data or models. For a survey of techniques using word classes, topic, syntax, etc., refer to (Bellegarda, 2004).

Linear interpolation is the most widely-used method for domain adaptation. Jelinek et al. (1991) describe its use for combining a cache language model and static language model. Another popular method is count merging; this has been motivated as an instance of MAP adaptation (Federico, 1996; Masataki et al., 1997). In terms of word-error rate, Iyer et al. (1997) found linear interpolation to give better speech recognition performance while Bacchiani et al. (2006) found count merging to be superior. Klakow (1998) proposes log-linear interpolation for domain adaptation. As compared to regular linear interpolation for bigram models, an improvement of 4% in perplexity and 0.2% absolute in word-error rate is found.

Della Pietra et al. (1992) introduce the idea of minimum discrimination information distributions. Given a prior model $q(y|x)$, the goal is to find the nearest model in Kullback-Liebler divergence that satisfies a set of linear constraints derived from adaptation data. The model satisfying these conditions is an exponential model containing one feature per constraint with $q(y|x)$ as its prior as in eq. (12). While MDI models have been used many times for language model adaptation, *e.g.*, (Kneser et al., 1997; Federico, 1999), they have not performed as well as linear interpolation in perplexity or word-error rate (Rao et al., 1995; Rao et al., 1997).

One important issue with MDI models is how to select the feature set specifying the model. With a small amount of adaptation data, one should intuitively use a small feature set, *e.g.*, containing just unigram features. However, the use of regularization can obviate the need for intelligent feature selection. In this work, we include all n-gram features present in the adaptation data for $n \in \{3, 4\}$. Chueh and Chien (2008) propose the use of inequality constraints for regularization (Kazama and Tsujii, 2003); here, we use $\ell_1 + \ell_2^2$ regularization instead. We hypothesize that the use of state-of-the-art regularization is the primary reason why we achieve better performance relative to interpolation and count merging as compared to earlier work.

6 Discussion

For exponential language models, eq. (2) tells us that with respect to test set performance, the number of model parameters seems to matter not at all; all that matters are the magnitudes of the parameter values. Consequently, one can improve exponential language models by adding features (or a prior model) that shrink parameter values while maintaining training performance, and from this observation we develop Heuristics 1 and 2. We use these ideas to motivate a novel and simple class-based language model that achieves perplexity and word-error rate improvements competitive with the best reported results for class-based models in the literature. In addition, we show that with regularization, MDI models can outperform both linear interpolation and count merging in language model combination. Still, Heuristics 1 and 2 are quite vague, and it remains to be seen how to determine when these heuristics will be effective.

In summary, we have demonstrated how the tradeoff between training set performance and model size impacts aspects of language modeling as diverse as backoff n-gram features, class-based models, and domain adaptation. In particular, we can frame performance improvements in all of these areas as methods that shrink models without degrading training set performance. All in all, eq. (2) is an important tool for both understanding and improving language model performance.

Acknowledgements

We thank Bhuvana Ramabhadran and the anonymous reviewers for their comments on this and earlier versions of the paper.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108.

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

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

Chuang-Hua Chueh and Jen-Tzung Chien. 2008. Reliable feature selection for language model adaptation. In *Proc. of ICASSP*, pp. 5089–5092.

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

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

Marcello Federico. 1999. Efficient language model adaptation through MDI estimation. In *Proc. of Eurospeech*, pp. 1583–1586.

Joshua T. Goodman. 2001. A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research.

Rukmini Iyer, Mari Ostendorf, and Herbert Gish. 1997. Using out-of-domain data to improve in-domain language models. *IEEE Signal Processing Letters*, 4(8):221–223, August.

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

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, March.

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

Dietrich Klakow. 1998. Log-linear interpolation of language models. In *Proc. of ICSLP*.

Reinhard Kneser, Jochen Peters, and Dietrich Klakow. 1997. Language model adaptation using dynamic marginals. In *Proc. of Eurospeech*.

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

Douglas B. Paul and Janet M. Baker. 1992. The design for the Wall Street Journal-based CSR corpus. In *Proc. of the DARPA Speech and Natural Language Workshop*, pp. 357–362, February.

P. Srinivasa Rao, Michael D. Monkowski, and Salim Roukos. 1995. Language model adaptation via minimum discrimination information. In *Proc. of ICASSP*, volume 1, pp. 161–164.

P. Srinivasa Rao, Satya Dharanipragada, and Salim Roukos. 1997. MDI adaptation of language models across corpora. In *Proc. of Eurospeech*, pp. 1979–1982.

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

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

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

Wen Wang and Mary P. Harper. 2002. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proc. of EMNLP*, pp. 238–247.

Wen Wang, Yang Liu, and Mary P. Harper. 2002. Rescoring effectiveness of language models using different levels of knowledge and their integration. In *Proc. of ICASSP*, pp. 785–788.

Wen Wang, Andreas Stolcke, and Mary P. Harper. 2004. The use of a linguistically motivated language model in conversational speech recognition. In *Proc. of ICASSP*, pp. 261–264.

Hirofumi Yamamoto and Yoshinori Sagisaka. 1999. Multi-class composite n-gram based on connection direction. In *Proc. of ICASSP*, pp. 533–536.

Hirofumi Yamamoto, Shuntaro Isogai, and Yoshinori Sagisaka. 2003. Multi-class composite n-gram language model. *Speech Communication*, 41(2-3):369–379.