Performance Prediction for 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

We investigate the task of performance prediction for language models belonging to the exponential family. First, we attempt to empirically discover a formula for predicting test set cross-entropy for n-gram language models. We build models over varying domains, data set sizes, and n-gram orders, and perform linear regression to see whether we can model test set performance as a simple function of training set performance and various model statistics. Remarkably, we discover a very simple relationship that predicts test performance with a correlation of 0.9996. We provide analysis of why this relationship holds, and show how this relationship can be used to motivate two heuristics for improving existing language models. We use the first heuristic to develop a novel class-based language model that outperforms a baseline word trigram model by up to 28% in perplexity and 2.1% absolute in speech recognition word-error rate on Wall Street Journal data. We use the second heuristic to provide a new motivation for minimum discrimination information (MDI) models (Della Pietra et al., 1992), and show how this method outperforms other methods for domain adaptation on a Wall Street Journal data set.

1 Introduction

In this paper, we investigate the following question for language models belonging to the exponential family: given some training data and test data drawn from the same distribution, can we accurately predict the test set performance of a model estimated from the training data? This problem is known as *performance prediction* and is useful for *model selection*, the task of selecting the best model from a set of candidate models given data.

Let us first define some notation and terminology. Events in our data have the form (x, y), where we attempt to predict the current word y given previous words x. We denote the training data $\mathcal{D}$ as

$$\mathcal{D} = (x_1, y_1), \ldots, (x_D, y_D) \tag{1}$$

and define $\tilde{p}(x, y)$ to be the empirical distribution of the training data:

$$\tilde{p}(x, y) = \frac{\text{count}_{\mathcal{D}}(x, y)}{D} = \frac{|\{d : (x_d, y_d) = (x, y)\}|}{D} \tag{2}$$

Similarly, we have a test set $\mathcal{D}^*$ and an associated empirical distribution $p^*(x, y)$. We take the performance of a conditional language model $p(y|x)$ to be the cross-entropy $\mathcal{H}(p^*, p)$ between the empirical test distribution p^* and the model $p(y|x)$:

$$\mathcal{H}(p^*, p) = -\sum_{x,y} p^*(x, y) \log p(y|x) \tag{3}$$

This is equivalent to the negative mean log-likelihood per event, as well as to log perplexity.

We only consider models in the exponential family. An exponential model $p_\Lambda(y|x)$ is a model with a set of *feature* functions $\mathcal{F} = \{f_1(x,y), \ldots, f_F(x,y)\}$ and equal number of parameters $\Lambda = \{\lambda_1, \ldots, \lambda_F\}$ where

$$p_\Lambda(y|x) = \frac{\exp(\sum_{i=1}^F \lambda_i f_i(x,y))}{Z_\Lambda(x)} \tag{4}$$

and where $Z_\Lambda(x)$ is a normalization factor defined as

$$Z_\Lambda(x) = \sum_{y'} \exp(\sum_{i=1}^{F} \lambda_i f_i(x,y')) \tag{5}$$

One of the seminal methods for performance prediction is the Akaike Information Criterion (AIC) (Akaike, 1973). Let $\hat{\Lambda}$ be the maximum likelihood estimate of Λ for a model on some training data. Akaike derived the following estimate for the expected value of the test set cross-entropy $\mathcal{H}(p^*, p_{\hat{\Lambda}})$:

$$\mathcal{H}(p^*, p_{\hat{\Lambda}}) \approx \mathcal{H}(\tilde{p}, p_{\hat{\Lambda}}) + \frac{F}{D} \tag{6}$$

$\mathcal{H}(\tilde{p}, p_{\hat{\Lambda}})$ is the cross-entropy of the training set, F is the number of parameters in the model, and D is the number of events in the training data. Since Akaike's original paper, many refinements and variations of this criterion have been developed, *e.g.*, (Takeuchi, 1976; Hurvich and Tsai, 1989; Lebreton et al., 1992), and these methods remain popular today (Burnham and Anderson, 2002). However, maximum likelihood estimates for language models typically yield infinite cross-entropy on test data, and thus AIC behaves poorly for these domains.

In this work, instead of deriving a performance prediction relationship theoretically, we attempt to *empirically* discover a formula for predicting test performance. Initially, we consider only n-gram language models, and build models over varying domains, data set sizes, and n-gram orders. We perform linear regression to discover whether we can model test set cross-entropy as a simple function of training set cross-entropy and other model statistics. For the 200+ n-gram models we evaluate, we find that the empirical relationship

$$\mathcal{H}(p^*, p_{\tilde{\Lambda}}) \approx \mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}}) + \frac{\gamma}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i| \tag{7}$$

holds with a correlation of 0.9996 where γ is a constant and where $\tilde{\Lambda} = \{\tilde{\lambda}_i\}$ are *regularized estimates* of the parameters; *i.e.*, rather than estimating model performance for maximum likelihood models as in AIC, we attempt to estimate performance for regularized models. In other words, test set cross-entropy can be approximated by the sum of the training set cross-entropy and the scaled sum of the magnitudes of the model parameters.

To maximize the correlation achieved by eq. (7), we find that it is necessary to use the same regularization method and regularization hyperparameters across models and that the optimal value of γ depends on the values of the hyperparameters. Consequently, we first evaluate several types of regularization and find which of these (and which hyperparameter values) work best across all domains, and use these values in all subsequent experiments. While ℓ_2^2 regularization gives the best performance reported in the literature for n-gram models, we find here that $\ell_1 + \ell_2^2$ regularization works even better.

Using eq. (7), we analyze the use of backoff features in n-gram models. AIC predicts that backoff features should degrade test performance since the maximum likelihood training performance is unchanged, while the number of parameters is increased. In fact, backoff features improve test performance a great deal and we show how this can be explained using eq. (7). In particular, we find that backoff features improve test performance by reducing the "size" of the model $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ rather than by improving training set performance. This suggests the following method for improving an existing language model: if the model can be "shrunk" without increasing its training cross-entropy, test cross-entropy will improve.

We then apply this technique to motivate two language models: a novel class-based language model and minimum discrimination information (MDI) models for domain adaptation (Della Pietra et al., 1992). We show how these models outperform other models in both perplexity and in speech recognition word-error rate on Wall Street Journal data. Furthermore, we show that eq. (7) predicts test performance accurately not only for word n-gram models, but for these other types of exponential models as well.

The organization of this paper is as follows: In Section 2, we evaluate various regularization techniques for n-gram models and select the method and associated hyperparameters that work best across all domains under consideration. In Section 3, we discuss our experiments to find a formula for predicting n-gram model performance, and provide an explanation for why eq. (7) can predict performance so well. In Section 4, we analyze the use of backoff features in n-gram models and motivate a heuristic for model design. In Sections 5 and 6, we introduce our novel class-based model and discuss MDI domain adaptation, and compare these methods against other techniques on Wall Street Journal data. Finally, in Sections 7 and 8 we talk about related work and present some conclusions.

2 Selecting a Regularization Method and Regularization Hyperparameters

In Section 3, we will attempt to discover a formula for performance prediction for regularized n-gram models. In this section, we address the issue of how exactly should we perform regularization in these later experiments. As mentioned in the introduction, we will find that performance prediction works best if we use the same regularization method and hyperparameters for all models under consideration, and so we attempt to find a regularization method and associated hyperparameters that perform well across many domains.

Following the terminology used by Dudík and Schapire (2006), the most widely-used and effective methods for regularizing exponential models are ℓ_1 regularization (Tibshirani, 1994; Khudanpur, 1995; Williams, 1995; Kazama and Tsujii, 2003; Goodman, 2004) and ℓ_2^2 regularization (Hoerl and Kennard, 1970; Lau, 1994; Chen and Rosenfeld, 2000; Lebanon and Lafferty, 2001). In these methods, we have a prior distribution over parameters Λ and choose the maximum *a posteriori* (MAP) parameter estimates given this prior. For ℓ_1 regularization, a Laplace prior is chosen; this translates to selecting Λ to optimize the objective function

$$\mathcal{O}_{\ell_1}(\Lambda) = \mathcal{H}(\tilde{p}, p_\Lambda) + \frac{\alpha}{D}\sum_{i=1}^{F}|\lambda_i| \tag{8}$$

The first term on the right is the cross-entropy of the training set and the second term penalizes large λ_i values. For ℓ_2^2 regularization, a Gaussian prior is chosen and the penalty term is quadratic in Λ:

$$\mathcal{O}_{\ell_2^2}(\Lambda) = \mathcal{H}(\tilde{p}, p_\Lambda) + \frac{1}{2\sigma^2 D}\sum_{i=1}^{F}\lambda_i^2 \tag{9}$$

Here, we omit the constant term corresponding to the prior normalization factor, and assume a single global hyperparameter (α or σ) is used rather than feature-specific hyperparameters.[1] While ℓ_2^2 regularization has been shown to be superior in language modeling (Chen and Rosenfeld, 2000; Goodman, 2004), ℓ_1 regularization performs better in many other domains, *e.g.*, (Ng, 2004).

While not as commonly used, another regularization scheme that has been shown to be effective for exponential models is *2-norm inequality* regularization (Kazama and Tsujii, 2003) which is an instance of $\ell_1 + \ell_2^2$ regularization as noted by Dudík and Schapire (2006). The corresponding objective function is

$$\mathcal{O}_{\ell_1+\ell_2^2}(\Lambda) = \mathcal{H}(\tilde{p}, p_\Lambda) + \frac{\alpha}{D}\sum_{i=1}^{F}|\lambda_i| + \frac{1}{2\sigma^2 D}\sum_{i=1}^{F}\lambda_i^2 \tag{10}$$

Note that ℓ_1 regularization can be considered a special case of this (by taking $\sigma = \infty$) as can ℓ_2^2 regularization (by taking $\alpha = 0$).

In this section, we evaluate ℓ_1, ℓ_2^2, and $\ell_1+\ell_2^2$ regularization for exponential n-gram models.[2] We define an exponential n-gram model on a training set as in (Chen and Rosenfeld, 2000). We assume raw text data consists of a sequence of sentences of the form $w_1 \cdots w_l$. For each sentence, we generate $l+1$ events of the form (x, y) where $y = w_j$ and $x = w_{j-n+1} \cdots w_{j-1}$ for $j = 1, \ldots, l+1$. We take w_{l+1} to be a distinguished end-of-sentence token and w_j for $j < 1$ to be a distinguished beginning-of-sentence token (Chen and Goodman, 1998). We say an n-gram ω occurs in the training data if ω is a suffix of xy for some event (x, y) in the training data. Then, an exponential n-gram model is an exponential model that has a feature f_ω for each n'-gram ω occurring in the training data for $n' \leq n$ where

$$f_\omega(x,y) = \begin{cases} 1 & \text{if } xy \text{ ends in } \omega \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

We would like to find the regularization method and associated hyperparameters that work best across different domains, training set sizes, and n-gram orders. As it is computationally expensive to evaluate a large number of hyperparameter settings over a large collection of models, we divide this search into two phases. First, we evaluate a large set of hyperparameter settings on a limited set of models to come up with a short list of candidate hyperparameter values. We then evaluate these candidates on our full set of models to find the best one.

2.1 Finding a Small Set of Candidate Hyperparameters

We build n-gram models over data from five different sources and consider three different vocabulary sizes for one of these sources, giving us seven "domains" in total. We refer to these domains by the letters *A*–*G*. The domains *C*–*G* consist of regular text data tokenized as words, while domains *A* and *B* consist of letter and part-of-speech sequences, respectively.

domain *A* Letter sequences corresponding to word spellings from a version of the Random House (RH) dictionary.

[1] In past work, feature-specific hyperparameters have failed to improve performance over using a single global hyperparameter; *e.g.*, (Lau, 1994; Kazama and Tsujii, 2003).

[2] We also considered adding a constant penalty for each feature with nonzero $\tilde{\lambda}_i$, which can be viewed as $\ell_0 + \ell_1 + \ell_2^2$ regularization. With this modification, the training objective function is no longer convex in Λ and we devised a greedy estimation algorithm for finding a local optimum in the search space. However, in limited trials, we found a penalty of zero gives the best test performance among the nonnegative penalties we evaluated. Consequently, we decided not to pursue this avenue further.

domain	data source	token type	range of n	training sents.	avg. sent. length	vocab. size
A	RH	letter	2–7	100–75k	8.1	27
B	WSJ	POS	2–7	100–30k	23.9	45
C	WSJ	word	2–5	100–100k	25.8	300
D	WSJ	word	2–5	100–100k	25.8	3k
E	WSJ	word	2–5	100–100k	25.8	21k
F	BN	word	2–5	100–100k	15.3	84k
G	SWB	word	2–5	100–100k	11.9	19k

Table 1: Statistics of data sets used in Sections 2 and 3 and elsewhere. For domain *A*, a "sentence" is a single word spelling.

	candidate (α, σ^2) values
ℓ_1	(0.7, ∞), (0.8, ∞), (0.9, ∞), (1.0, ∞), (1.1, ∞), (1.2, ∞)
ℓ_2^2	(0, 1), (0, 1.2), (0, 1.5), (0, 2), (0, 2.5), (0, 3), (0, 4), (0, 5)
$\ell_1 + \ell_2^2$	(0.5, 5), (0.5, 6), (0.5, 7), (0.6, 7), (0.6, 8), (0.6, 10)

Table 2: Best hyperparameter settings found in Section 2.1 for each regularization method.

domain *B* Part-of-speech (POS) sequences corresponding to sentences from tagged Wall Street Journal (WSJ) text from the Penn Treebank 3 (Marcus et al., 1993).

domains *C–E* 1993 Wall Street Journal text with verbalized punctuation from the CSR-III Text corpus from the Linguistic Data Consortium. These three domains differ only in vocabulary. In domains *C* and *D*, the vocabulary consists of the 300 and 3000 most frequent words, respectively, in our entire WSJ data set. In domain *E*, the vocabulary consists of the union of the training vocabulary and 20k word "closed" test vocabulary from the first Wall Street Journal Continuous Speech Recognition (CSR) corpus (Doddington, 1992; Paul and Baker, 1992).

domain *F* 1997 Broadcast News (BN) text (Graff, 1997). An internal IBM vocabulary is used.

domain *G* Switchboard (SWB) text (Godfrey et al., 1992). The vocabulary consists of all words occurring at least twice in the entire data set.

We provide summary statistics for each domain in Table 1. For each domain, we first take all of our data and randomize the order of sentences in that data. We partition off two development sets and an evaluation data set (5000 "sentences" each in domain *A* and 2500 sentences elsewhere) and use the remaining data as training data. In this way, we assure that our training and test data are drawn from essentially the same distribution as is assumed in our experiments for performance prediction. Training set sizes in sentences are 100, 300, 1000, 3000, etc., up to the maximums given in Table 1.

For each domain, we choose several "representative" training set sizes and n-gram orders from the range of values given in Table 1, amounting to 57 models in total over the seven domains. For each of these models, we estimate parameters using $\ell_1 + \ell_2^2$ regularization over many different values of the hyperparameters (α, σ^2). In particular, we perform a grid search, trying each value

$$\alpha \in \{0.0, 0.1, 0.2, \dots, 1.2\} \tag{12}$$

with each value

$$\sigma^2 \in \{1, 1.2, 1.5, 2, 2.5, 3, 4, 5, 6, 7, 8, 10, \infty\}, \tag{13}$$

giving us $13 \times 13 = 169$ hyperparameter settings in all for each model. Recall that setting $\sigma = \infty$ corresponds to ℓ_1 regularization and that $\alpha = 0$ corresponds to ℓ_2^2 regularization, so we are in effect evaluating ℓ_1 and ℓ_2^2 regularization in addition to $\ell_1 + \ell_2^2$ regularization. We use a variant of iterative scaling for parameter estimation; we note that our regularized objective functions are all convex in Λ.[3]

For each model and each (α, σ^2), we compute the cross-entropy of the first development set for the corresponding domain; we denote this value as $H^m_{\alpha,\sigma}$ for the mth model, $m \in \{1, \ldots, 57\}$. Then, for each m and (α, σ^2), we can compute how much worse the settings (α, σ^2) perform with model m as compared to the best hyperparameter settings for that model:

$$\hat{H}^m_{\alpha,\sigma} = H^m_{\alpha,\sigma} - \min_{\alpha,\sigma} H^m_{\alpha,\sigma} \tag{18}$$

We would like to select (α, σ^2) with small "error" $\hat{H}^m_{\alpha,\sigma}$ across all models m; *i.e.*, we want to minimize how much performance we lose by using that single (α, σ^2) across models as compared to optimizing hyperparameters separately for each model. In particular, we choose the (α, σ^2) value that minimizes the root mean squared (RMS) error across models:

$$\hat{H}^{\text{RMS}}_{\alpha,\sigma} = \sqrt{\frac{1}{57}\sum_{m=1}^{57} (\hat{H}^m_{\alpha,\sigma})^2} \tag{19}$$

For each of ℓ_1, ℓ_2^2, and $\ell_1 + \ell_2^2$ regularization, we choose the 6–8 best hyperparameter settings according to this metric; we evaluate these candidate hyperparameters further in the next section. We list the candidate hyperparameters for each regularization method in Table 2.

[3]Here, we describe how we adapt improved iterative scaling (Della Pietra et al., 1997) to $\ell_1 + \ell_2^2$ regularization. The original update is to take

$$\lambda_i^{(t+1)} \leftarrow \lambda_i^{(t)} + \delta_i^{(t)} \tag{14}$$

where $\delta_i^{(t)}$ satisfies

$$\sum_{x,y} \tilde{p}(x,y) f_i(x,y) = \sum_{x,y} \tilde{p}(x) p_{\Lambda^{(t)}}(y|x) f_i(x,y) \exp(\delta_i^{(t)} f^\#(x,y)) \tag{15}$$

and where $f^\#(x,y) = \sum_i f_i(x,y)$. As the right-hand side of this equation is strictly monotonic in $\delta_i^{(t)}$, we can solve for $\delta_i^{(t)}$ using a simple search algorithm such as binary search.

To adapt this algorithm to ℓ_2^2 regularization, one simply adds the term $\frac{\lambda_i^{(t)} + \delta_i^{(t)}}{D\sigma_i^2}$ to the right-hand side of eq. (15) (Chen and Rosenfeld, 2000). However, handling the penalty term for ℓ_1 regularization is trickier because of the discontinuity in the derivative of $|\lambda_i|$ at $\lambda_i = 0$. To address this, we do not allow λ_i to jump across $\lambda_i = 0$ in a single iteration; *i.e.*, if $\delta_i^{(t)}$ will cause $\lambda_i^{(t+1)}$ to have the opposite sign as $\lambda_i^{(t)}$, we reduce the magnitude of $\delta_i^{(t)}$ so that $\lambda_i^{(t+1)} = 0$.

The algorithm we use is as follows: If $\lambda_i^{(t)} \neq 0$, we choose $\delta_i^{(t)}$ to satisfy

$$\sum_{x,y} \tilde{p}(x,y) f_i(x,y) = \sum_{x,y} \tilde{p}(x) p_{\Lambda^{(t)}}(y|x) f_i(x,y) \exp(\delta_i^{(t)} f^\#(x,y)) + \frac{\alpha_i}{D} \operatorname{sgn} \lambda_i^{(t)} + \frac{\lambda_i^{(t)} + \delta_i^{(t)}}{D\sigma_i^2} \tag{16}$$

If $\operatorname{sgn}(\lambda_i^{(t)} + \delta_i^{(t)}) = -\operatorname{sgn} \lambda_i^{(t)}$, we set $\delta_i^{(t)} = -\lambda_i^{(t)}$. If $\lambda_i^{(t)} = 0$, in eq. (16) we replace $\operatorname{sgn} \lambda_i^{(t)}$ with

$$\operatorname{sgn}\left(\sum_{x,y} \tilde{p}(x,y) f_i(x,y) - \sum_{x,y} \tilde{p}(x) p_{\Lambda^{(t)}}(y|x) f_i(x,y)\right) \tag{17}$$

(α, σ^2)	RMS error	mean error	max error
(0.5, 6)	0.011	0.007	0.033
(0.6, 8)	0.011	0.008	0.032
(0.6, 10)	0.011	0.009	0.037
(0.5, 7)	0.012	0.007	0.036
(0.6, 7)	0.012	0.009	0.047
(0.5, 5)	0.013	0.009	0.061
(0.0, 2.5)	0.034	0.025	0.137
(0.0, 2)	0.037	0.027	0.164
(0.0, 3)	0.039	0.028	0.123
(0.0, 4)	0.054	0.041	0.141
(0.0, 1.5)	0.056	0.037	0.249
(0.9, ∞)	0.062	0.044	0.230
(0.8, ∞)	0.062	0.044	0.236
(0.0, 5)	0.070	0.056	0.172
(0.7, ∞)	0.072	0.054	0.246
(0.0, 1.2)	0.082	0.052	0.354
(1.0, ∞)	0.092	0.068	0.306
(1.1, ∞)	0.093	0.069	0.306
(1.2, ∞)	0.095	0.070	0.307
(0.0, 1)	0.107	0.069	0.450

Table 3: Mean and maximum values of $\hat{H}^m_{\alpha,\sigma}$ for each candidate hyperparameter setting over all 218 models evaluated in Section 2.2. All errors are in nats, or natural bits.

2.2 Selecting the Single Best Hyperparameter Setting

To choose the best hyperparameters from within the candidate set found in Section 2.1, we repeat the same analysis as before except over a larger set of models. Instead of selecting only "representative" training set sizes and n-gram orders from the range of values given in Table 1, we select *all* of them, resulting in a total of 218 models over the seven domains. For each of these models, we estimate parameters using each of the 20 candidate hyperparameter settings given in Table 2. In addition, for each of the seven domains, we select the six best *domain-specific* hyperparameter settings not present in the domain-independent candidates and evaluate these as well. As before, we can compute $\hat{H}^m_{\alpha,\sigma}$ for each model and (α, σ^2), and then compute the root mean squared error for each (α, σ^2) over all models. In Table 3, we display this value as well as the mean and maximum errors for all candidate parameter settings. All errors are reported in *nats*, or natural bits. That is, when we compute cross-entropies using eq. (3), we use natural logarithms rather than logarithms base 2.[4]

On the development sets, the (α, σ^2) value with the lowest squared error is (0.5, 6), and these are the hyperparameter settings we use in all later experiments unless otherwise noted. The root mean squared error, mean error, and maximum error for these hyperparameters on the evaluation

[4]The reason that we report cross-entropies in nats rather than bits is because of eq. (7). The most natural interpretation for the units of the λ_i's is nats, since they are exponentiated base e when computing model probabilities. This will let us directly compare γ values with average discounts in Section 3.2.

sets are 0.011, 0.007, and 0.033, respectively (the same values as for the development set). To put these values in perspective, these correspond to differences of 1.1%, 0.7%, and 3.4% in perplexity, respectively; differences in perplexity of about 1% are generally considered insignificant. Thus, we see that we can achieve good performance across domains, data set sizes, and n-gram orders using a single set of hyperparameters as compared to optimizing hyperparameters separately for each model.[5]

2.3 Comparing Regularization Methods for N-Gram Models

While this is not a focus of this work, we compare $\ell_1+\ell_2^2$ regularization with ℓ_1 and ℓ_2^2 regularization for n-gram models since this may be of general interest. Since ℓ_1 and ℓ_2^2 regularization are special cases of $\ell_1+\ell_2^2$ regularization, we expect them to do no better.[6] First, we consider the case where we use a single hyperparameter setting across all models for each regularization method. For ℓ_1 regularization, the best setting on the development data is $\alpha = 0.9$, and for ℓ_2^2 regularization, we have $\sigma^2 = 2.5$. On the evaluation data, this yields a maximum error of 0.233 and 0.131 nats, respectively, corresponding to perplexity differences of 26% and 14%. These perplexity differences are considered very significant, and thus ℓ_1 and ℓ_2^2 regularization are unsuitable when using a single hyperparameter setting across models.

In addition to comparing performance with the best performance found for each model, we can directly compare each regularization method with each other. For each of our 218 models, we compute the difference between the evaluation cross-entropy for ℓ_1 regularization (with $\alpha = 0.9$) and $\ell_1+\ell_2^2$ regularization (with $\alpha = 0.5, \sigma^2 = 6$). On average, $\ell_1+\ell_2^2$ regularization is 0.036 nats better with a median gain of 0.021 nats. The maximum difference is 0.216 nats (*i.e.*, $\ell_1+\ell_2^2$ is better by this much), and the minimum difference is -0.019 nats (*i.e.*, $\ell_1+\ell_2^2$ is worse by this much). Comparing ℓ_2^2 regularization with $\ell_1+\ell_2^2$ regularization, $\ell_1+\ell_2^2$ regularization is 0.018 nats better on average with a median gain of 0.014 nats. The maximum difference is 0.116 nats and minimum is -0.019 nats.

Instead of assuming the same hyperparameter values across all models, we can optimize hyperparameters separately for each model. For each model, we find the best hyperparameter settings for each regularization method on the development sets and then examine the performance of these hyperparameter settings on the evaluation sets. First, we compare ℓ_1 with $\ell_1+\ell_2^2$ regularization. We find that $\ell_1+\ell_2^2$ regularization is 0.040 nats better on average with a median gain of 0.027 nats. The maximum difference is 0.234 nats and minimum is -0.0001 nats. For ℓ_2^2 regularization, $\ell_1+\ell_2^2$ regularization is 0.012 nats better on average with a median gain of 0.007 nats. The maximum difference is 0.101 nats and minimum is -0.0002 nats.

Thus, we find that $\ell_1+\ell_2^2$ regularization is superior to both ℓ_1 and ℓ_2^2 regularization for n-gram models, with ℓ_2^2 regularization being better than ℓ_1. On average, the gain of $\ell_1+\ell_2^2$ regularization over ℓ_2^2 regularization is quite small, 1–2% in perplexity, but for some models, the gain can be more than 10% in perplexity. However, we see large performance differences only for sparsely estimated models; when there is a copious amount of training data, the difference in performance between

[5]When we compute the best performance $\min_{\alpha,\sigma} H^m_{\alpha,\sigma}$ for each model, we consider only the candidate hyperparameters given in Table 2 and the 6–8 additional domain-specific hyperparameter settings. However, it is possible that the best performance is actually achieved outside of these hyperparameter settings. To estimate the size of this effect, we compare the best performance achieved among these hyperparameters with the best performance achieved using the grid search in Section 2.1. Over the 57 models evaluated in Section 2.1, we found the largest difference between these two values to be about 0.003 nats, or 0.3% in perplexity. Thus, this issue is probably insignificant.

[6]It is possible that ℓ_1 or ℓ_2^2 regularization does better than $\ell_1+\ell_2^2$ due to hyperparameter overfitting. That is, we may pick hyperparameters for $\ell_1+\ell_2^2$ regularization on the development set that do not perform as well on the evaluation set.

statistic	RMSE	coeff.
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$	0.043	0.938
$\frac{1}{D}\sum_{i:\tilde{\lambda}_i>0}\tilde{\lambda}_i$	0.044	0.939
$\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i$	0.047	0.940
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert^{\frac{4}{3}}$	0.162	0.755
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert^{\frac{3}{2}}$	0.234	0.669
$\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i^2$	0.429	0.443
$\frac{F_{\neq 0}}{D}$	0.709	1.289
$\frac{F_{\neq 0}\log D}{D}$	0.783	0.129
$\frac{F}{D}$	0.910	1.109
$\frac{F\log D}{D}$	0.952	0.112
1	1.487	1.698
$\frac{F}{D-F-1}$	2.232	-0.028
$\frac{F_{\neq 0}}{D-F_{\neq 0}-1}$	2.236	-0.023

Table 4: Root mean squared error (RMSE) in nats when predicting difference in development set and training set cross-entropy as linear function of a single statistic. The last column is the optimal coefficient found for that statistic.

statistics	RMSE
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$, $\frac{F_{\neq 0}}{D}$	0.0419
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$, $\frac{F}{D}$	0.0421
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$, $\frac{F_{\neq 0}\log D}{D}$	0.0421
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$, $\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i$	0.0423
$\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i$, $\frac{1}{D}\sum_{i:\tilde{\lambda}_i>0}\tilde{\lambda}_i$	0.0423

Table 5: Root mean squared error (RMSE) in nats when predicting difference in development set and training set cross-entropy as linear function of two statistics; the top five pairs are listed.

regularization methods is quite small.

3 Predicting Test Set Performance for N-Gram Models

3.1 Using Linear Regression to Find a Formula for Performance Prediction

Now that we have established which regularization method and hyperparameters to use ($\ell_1 + \ell_2^2$ regularization with $\alpha = 0.5$ and $\sigma^2 = 6$), we attempt to empirically discover a simple formula for predicting the test cross-entropy of regularized n-gram models. The basic strategy is as follows: We first build a large number of n-gram models over different domains, training set sizes, and n-gram orders. Then, we come up with a set of candidate statistics, *e.g.*, training cross-entropy, number of features, etc., and do linear regression to try to best model test cross-entropy as a linear function of these candidate statistics. We assume that training and test data come from the same distribution; otherwise, it would be difficult to predict test performance. Since our training and development sets are portions of the same randomly shuffled data set, this assumption should be satisfied here.

We use the same 218 n-gram models as in Section 2.2. Before we introduce the statistics we consider, we first review notation: Recall that we use D to denote the number of events in the

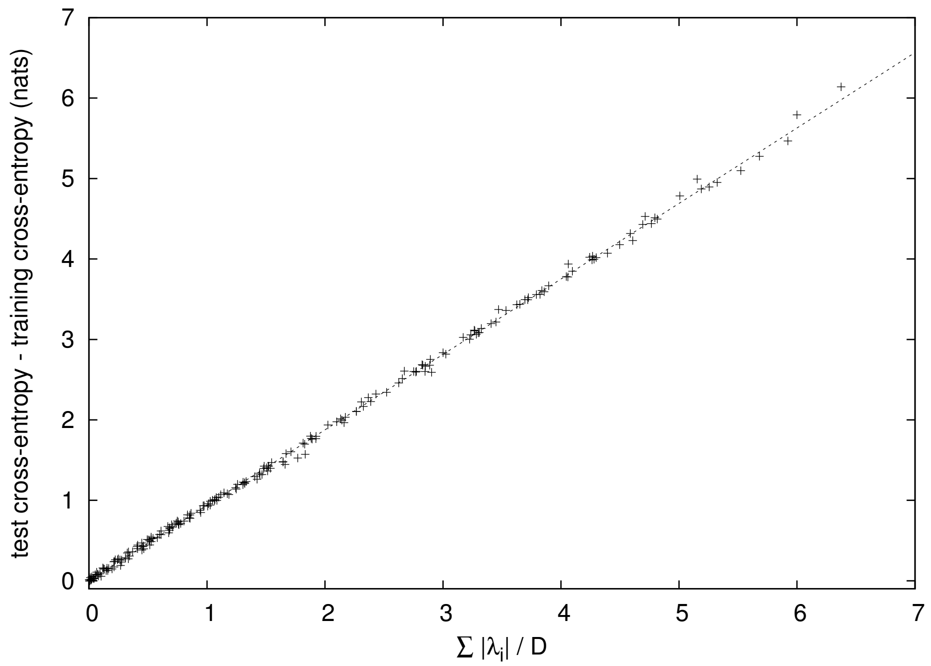

Figure 1: Graph of optimism on evaluation data *vs.* $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for various n-gram models under $\ell_1 + \ell_2^2$ regularization, $\alpha = 0.5$ and $\sigma^2 = 6$. The line represents the predicted optimism according to eq. (7) with $\gamma = 0.938$.

training data and F to denote the number of features (or parameters) in a model. Let $F_{\neq 0}$ be the number of features f_i with $\tilde{\lambda}_i \neq 0$. In exponential models, a feature f_i with $\tilde{\lambda}_i = 0$ has no effect in the model, so it is unclear whether such a feature should be included in the total feature count.[7] Also, note that cross-entropy is negative log-likehood *per event*, so most of the statistics below will be scaled by the factor $\frac{1}{D}$ to match.

Then, for each of our regularized models we compute the following statistics for use in linear regression:

$\mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}})$ This is the cross-entropy of the training data, and training set performance is an important term in many model selection methods. The coefficient for this statistic is usually taken to be 1.

$\frac{F}{D}$**,** $\frac{F_{\neq 0}}{D}$ The term $\frac{F}{D}$ is used in AIC (eq. (6)) among other methods. As noted above, it is unclear whether we should count features with $\tilde{\lambda}_i = 0$, so we consider using both F and $F_{\neq 0}$.

$\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ This term is used in the ℓ_1 regularization objective function (eq. (8)). Regularization can be viewed as a form of model selection.

$\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i$**,** $\frac{1}{D}\sum_{i:\tilde{\lambda}_i>0}\tilde{\lambda}_i$ We considered these terms as well since the impact of features with negative $\tilde{\lambda}_i$ is unclear.

$\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i^2$ This term is used in the ℓ_2^2 regularization objective function (eq. (9)).

[7]With ℓ_1 or $\ell_1 + \ell_2^2$ regularization, a significant fraction of the $\tilde{\lambda}_i$'s may be set to zero (Tibshirani, 1994; Kazama and Tsujii, 2003).

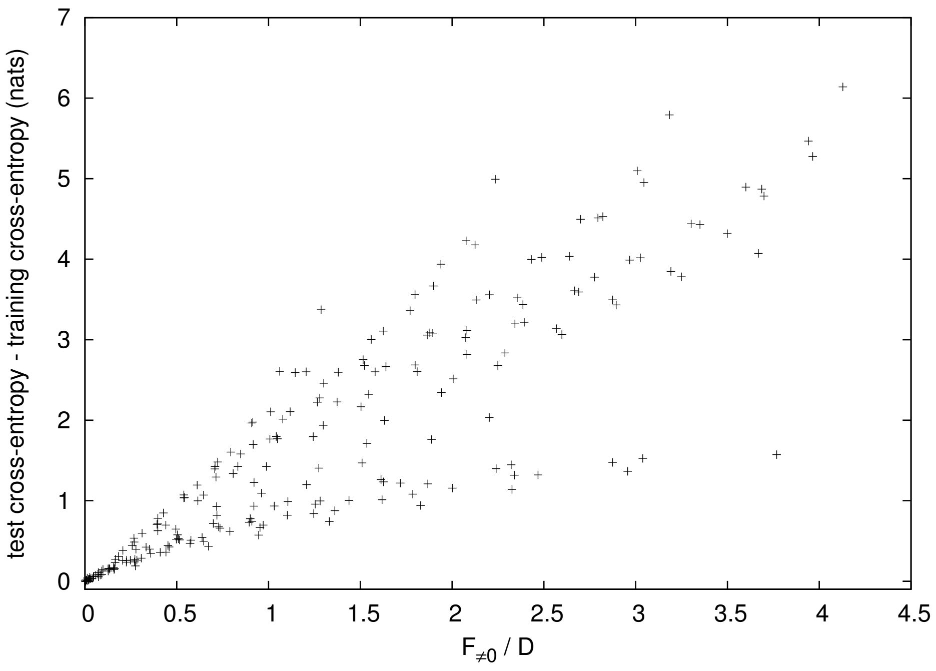

Figure 2: Graph of optimism on evaluation data *vs.* $\frac{F_{\neq 0}}{D}$ for various n-gram models under $\ell_1 + \ell_2^2$ regularization, $\alpha = 0.5$ and $\sigma^2 = 6$.

$\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|^{\frac{3}{2}}$, $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|^{\frac{4}{3}}$ These terms are suggested by the analysis in Section 3.2.

$\frac{F \log D}{D}$, $\frac{F_{\neq 0} \log D}{D}$ The term $\frac{F \log D}{D}$ is used in the Bayesian Information Criterion (Schwarz, 1978).

$\frac{F}{D-F-1}$, $\frac{F_{\neq 0}}{D-F_{\neq 0}-1}$ The term $\frac{F}{D-F-1}$ is used in a modification of AIC known as AIC$_c$ (Hurvich and Tsai, 1989).

1 The value 1 is present to handle constant offsets in linear regression.

After some initial investigation, it became clear that training set cross-entropy is a very good (partial) predictor of test set cross-entropy with coefficient 1. As there is ample theoretical support for this, instead of fitting test set performance directly, we chose to model the difference between test and training performance as a function of the remaining statistics. This difference is sometimes referred to as the *optimism* of a model:

$$\textit{optimism}(p_{\tilde{\Lambda}}) \equiv \mathcal{H}(p^*, p_{\tilde{\Lambda}}) - \mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}}) \quad (20)$$

As before, all cross-entropies are expressed in nats.

First, we attempt to model optimism as a linear function of a single statistic. For each statistic listed previously, we perform linear regression to minimize root mean squared error when predicting development set cross-entropies. In Table 4, we display the best coefficient and root mean squared error for each statistic. We see that three statistics have by far the lowest error: $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$, $\frac{1}{D}\sum_{i:\tilde{\lambda}_i>0}\tilde{\lambda}_i$, and $\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i$. In practice, most $\tilde{\lambda}_i$ in n-gram models are positive, so these statistics tend to have very similar values. We choose the best performing of these, $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$, and show in Section 3.2 why this statistic is more appealing than the other two. To account for the possibility that optimism may be a non-linear function of one of the statistics under consideration, we graphed optimism against each of our statistics over our 218 models. Visual inspection revealed no simple way of reducing prediction error using non-linear functions.

Next, we investigate modeling optimism as a linear function of a *pair* of statistics. In Table 5, we display the root mean squared error for the best pairs of statistics. Note that the best MSRE for two variables (0.042) is only slightly lower than the best MSRE for one (0.043), so it is questionable whether having a second variable actually helps. To examine this situation further, for each of our seven domains *A*–*G*, we redo our linear regression using models from only that domain and look at how the resulting coefficients vary by domain. For the first three pairs of statistics in Table 5, the coefficient for $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ is reasonably stable across domains, but the coefficient for the other statistic is near zero, sometimes positive, sometimes negative. This suggests that the second statistic is not significant. For the other two pairs of statistics, coefficients for both statistics varied widely across domains, sometimes large and negative, sometimes large and positive. This suggests that these are not robust predictions.

Thus, our analysis suggests that among our candidate functions, the best predictor of optimism is simply

$$\text{optimism} \approx \frac{\gamma}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i| \tag{21}$$

where $\gamma = 0.938$, with this value being independent of domain, training set size, and n-gram order. In other words, the difference between test and training cross-entropy is a linear function of $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$, the sum of the magnitudes of the model parameters scaled per event. Substituting this into eq. (20) and rearranging, we get eq. (7).

To assess the accuracy of eq. (7), we compute various statistics on our evaluation sets using the best γ from our development data, *i.e.*, $\gamma = 0.938$. In Figure 1, we graph optimism for the evaluation data against $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for each of our models; we see that the linear correlation is very good. For contrast, in Figure 2 we graph optimism against $\frac{F_{\neq 0}}{D}$ as would be suggested by an AIC-like approach. For eq. (7), the correlation between the actual and predicted optimism on the evaluation data is 0.9996; the mean absolute error is 0.030 nats; the root mean squared error is 0.043 nats; the median absolute error is 0.022 nats; and the maximum absolute error is 0.166 nats. Thus, on average we can predict optimism (and test set performance) to within 2–4% in perplexity, though in the worst case we may be off by as much as 18% in perplexity. In Section 7.1, we discuss how these results compare to other results for performance prediction.

3.2 Why Does Performance Prediction Work So Well?

The correlation in Figure 1 is remarkably high, and thus it begs for an explanation. First, let us express the difference in test and training cross-entropy for a model in terms of its parameters Λ. Substituting eq. (4) into eq. (3), we get

$$\begin{aligned}
\mathcal{H}(\tilde{p}, p_\Lambda) &= -\sum_{x,y}\tilde{p}(x,y)\log\frac{\exp(\sum_{i=1}^{F}\lambda_i f_i(x,y))}{Z_\Lambda(x)} && (22)\\
&= -\sum_{x,y}\tilde{p}(x,y)\sum_{i=1}^{F}\lambda_i f_i(x,y) + \sum_{x}\tilde{p}(x)\log Z_\Lambda(x) && (23)\\
&= -\sum_{i=1}^{F}\lambda_i E_{\tilde{p}}[f_i] + \sum_{x}\tilde{p}(x)\log Z_\Lambda(x) && (24)
\end{aligned}$$

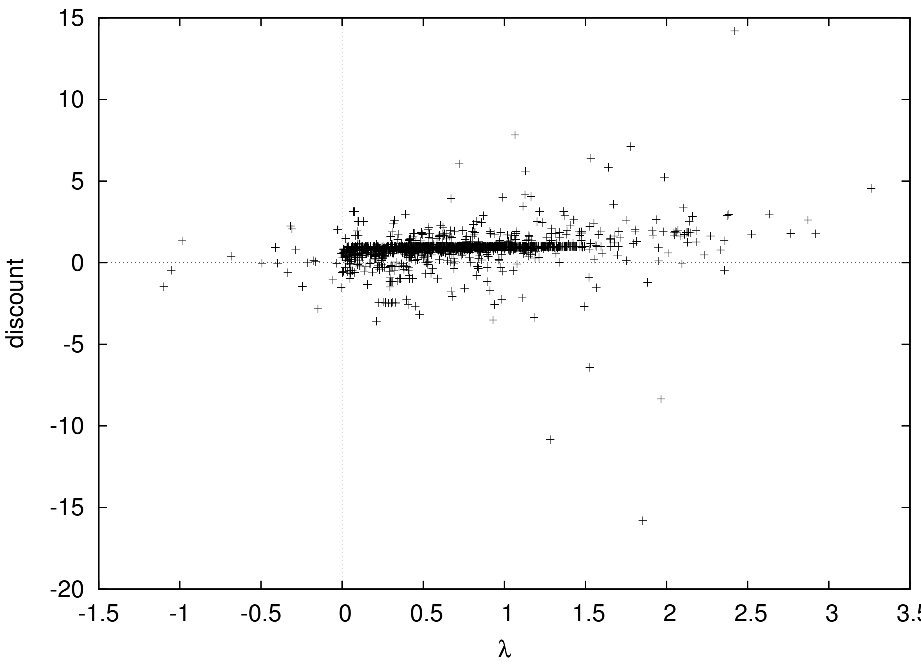

Figure 3: Graph of discount $(E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \times D$ versus $\tilde{\lambda}_i$ for all features in letter 5-gram model built on 100 word training set from domain *A*.

Then, we can express the difference in test and training performance as

$$\mathcal{H}(p^*, p_\Lambda) - \mathcal{H}(\tilde{p}, p_\Lambda) = \sum_{i=1}^{F} \lambda_i (E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) + \sum_x (p^*(x) - \tilde{p}(x)) \log Z_\Lambda(x) \tag{25}$$

If we ignore the last term on the right, we see that optimism for exponential models is a simple linear function of the λ_i's with corresponding coefficients $E_{\tilde{p}}[f_i] - E_{p^*}[f_i]$ (which are constant with respect to λ_i).[8]

Then, we can ask what values of $E_{\tilde{p}}[f_i] - E_{p^*}[f_i]$ would let us satisfy eq. (7). Consider the following relationship:

$$(E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \times D \approx \gamma \,\mathrm{sgn}\, \tilde{\lambda}_i \tag{27}$$

If we substitute this into eq. (25) and ignore the last term on the right again, this gives us exactly eq. (7). We refer to the value $(E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \times D$ as the *discount* of a feature. It can be thought of as representing how many times less the feature occurs in the test data as opposed to the training data, if the test data were normalized to be the same size as the training data. Discounts for n-grams have been studied extensively in the past, *e.g.*, (Good, 1953; Church and Gale, 1991; Chen and Goodman, 1998), and tend not to vary much across different training set sizes.

[8]It is possible to express the normalization factors $Z_\Lambda(x)$ as just additional features and parameters in a model (or one can omit them completely in unnormalized models). In this case, we simply have

$$\mathcal{H}(p^*, p_\Lambda) - \mathcal{H}(\tilde{p}, p_\Lambda) = \sum_{i=1}^{F} \lambda_i (E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \tag{26}$$

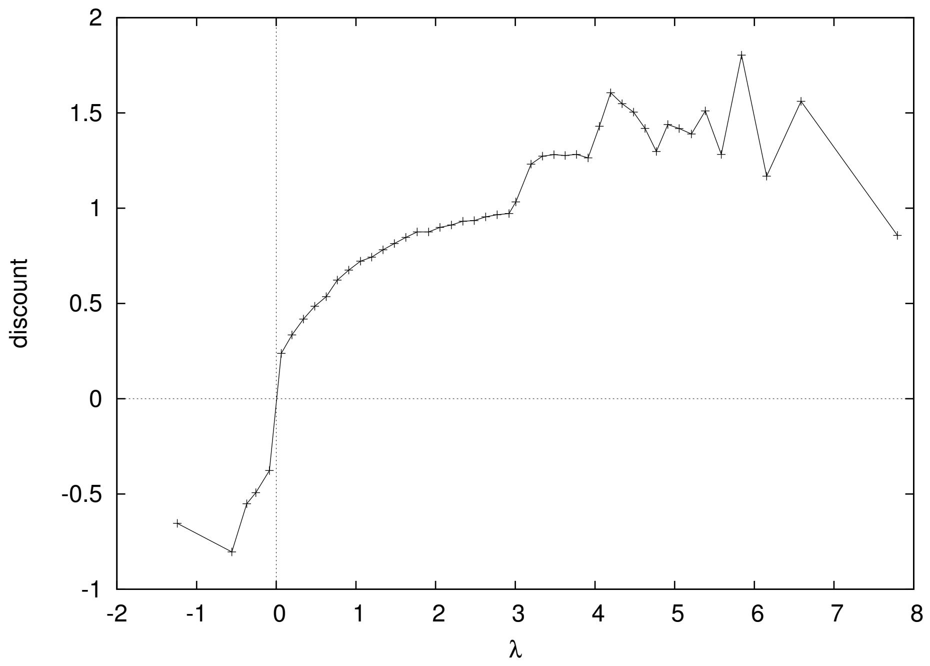

Figure 4: Smoothed graph of discount versus $\tilde{\lambda}_i$ for all features in word trigram model built on 30k sentence training set from domain *E*. Each smoothed point represents the average of at least 1024 raw data points.

Then, we can empirically check how well eq. (27) holds for real-life regularized n-gram models. Initially, we construct a total of ten n-gram models on domains *A* (letter sequences) and *E* (WSJ data, 21k word vocabulary). Using the same regularization as before, we build four letter 5-gram models on domain *A* on training sets ranging in size from 100 words to 30k words, and six models (either trigram or 5-gram) on domain *E* on training sets ranging from 100 sentences to 30k sentences. We create large development test sets (45k words for domain *A* and 70k sentences for domain *E*) to better estimate $E_{p^*}[f_i]$.

In Figure 3, we graph the discount $(E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \times D$ versus $\tilde{\lambda}_i$ for each feature in a single model with $\tilde{\lambda}_i \neq 0$ (letter 5-gram model, 100 words training). As the data appears very noisy and it is difficult to discern any patterns that might exist, we smooth the data. We sort the points by $\tilde{\lambda}_i$ values and partition the points into buckets containing at least k points with k=512 or 1024.[9] We average all of the points in each bucket to get a "smoothed" data point, and plot this single point for each bucket. In Figure 4, we plot the smoothed data for a single model (trigram model, domain *E*, 30k sentences training). While the data is quite smooth in some regions, for large $\tilde{\lambda}_i$ the variation in discounts is quite high and the graph is bumpy even after smoothing. In Figure 5, we plot the smoothed data for all ten models in the range $\tilde{\lambda}_i \in [-1, 4]$.

At first glance, it doesn't appear that discounts are constant on average for $\tilde{\lambda}_i > 0$ (or $\tilde{\lambda}_i < 0$) as in eq. (27). Instead, the smoothed discounts appear to be increasing and sublinear in $\tilde{\lambda}_i$ for $\tilde{\lambda}_i > 0$.[10] However, a closer examination reveals a more nuanced story. First, let us examine how

[9]We also enforce a minimum width for each bucket in $\tilde{\lambda}_i$.

[10]This suggests using the term $\tilde{\lambda}_i^a$ to help model $E_{\tilde{p}}[f_i] - E_{p^*}[f_i]$ for $a \in (0, 1)$, which from eq. (25) would translate to the term $\tilde{\lambda}_i^{a+1}$ in performance prediction. However, we evaluate the statistics $\frac{1}{D}\sum_{i=1}^{F} |\tilde{\lambda}_i|^{\frac{4}{3}}$ and $\frac{1}{D}\sum_{i=1}^{F} |\tilde{\lambda}_i|^{\frac{3}{2}}$ in Section 3.1, and neither helps.

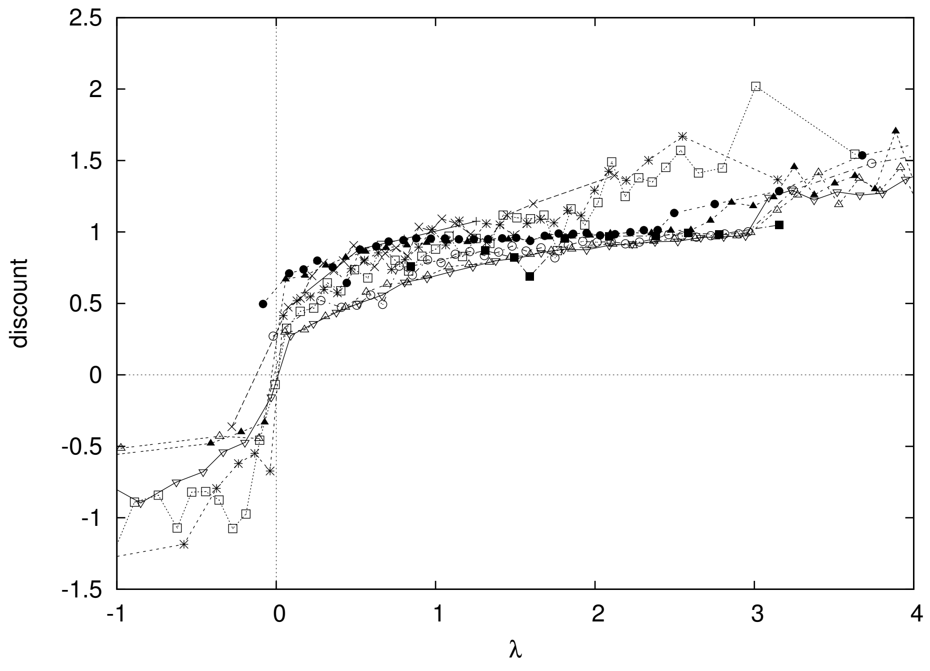

Figure 5: Smoothed graph of discount versus $\tilde{\lambda}_i$ for all features in ten different models built on domains *A* and *E*. Each smoothed point represents the average of at least 512 raw data points.

much different ranges of $\tilde{\lambda}_i$ contribute to the overall value of $\sum_{i=1}^{F} \tilde{\lambda}_i (E_{\tilde{p}}[f_i] - E_{p^*}[f_i])$ in eq. (25). In Figures 6 and 7, we plot histograms of this relationship for two different models. What we find is that the great majority of the mass is concentrated in a relatively small range of $\tilde{\lambda}_i$, the range $\tilde{\lambda}_i \in [0, 4]$, and this holds for all ten models under consideration. Thus, the value of the first term on the right-hand side of eq. (25) is mostly determined by the discounts for $\tilde{\lambda}_i$ in this limited range, which we refer to as the *critical region*. Looking at Figure 5, we note that at a very rough level, feature discounts are somewhat flat for $\tilde{\lambda}_i \in [0, 4]$ and have average values in the neighborhood of $\gamma = 0.938$ as required by eq. (27). However, while this relationship holds at a coarse level, clearly the relationship is not very precise.

Then, we can ask: how closely does eq. (27) have to hold in order to achieve the performance prediction accuracy that we actually achieve? In Section 3.1, we find that the average absolute prediction error of eq. (7) is about 0.03 nats. Let us say we have

$$(E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \times D \approx (\gamma + \epsilon)\, \mathrm{sgn}\, \tilde{\lambda}_i \tag{28}$$

for some error term ϵ. Substituting this into eq. (25) (and ignoring the last term as before), this gives us

$$\mathcal{H}(p^*, p_{\tilde{\Lambda}}) \approx \mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}}) + \frac{\gamma}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i| + \frac{\epsilon}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i| \tag{29}$$

The last term can be viewed as corresponding to prediction error, and its sensitivity to ϵ depends on the size $\frac{1}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i|$ of the model. If $\frac{1}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i|$ is low, then the average discount can be quite far from $\gamma = 0.938$ without yielding large absolute error (where we are mostly concerned with the average discount in the critical region). For example, to attain an absolute prediction error of 0.03 nats with a model of size 0.3 nats/event (the approximate size of the smallest model of the ten), then

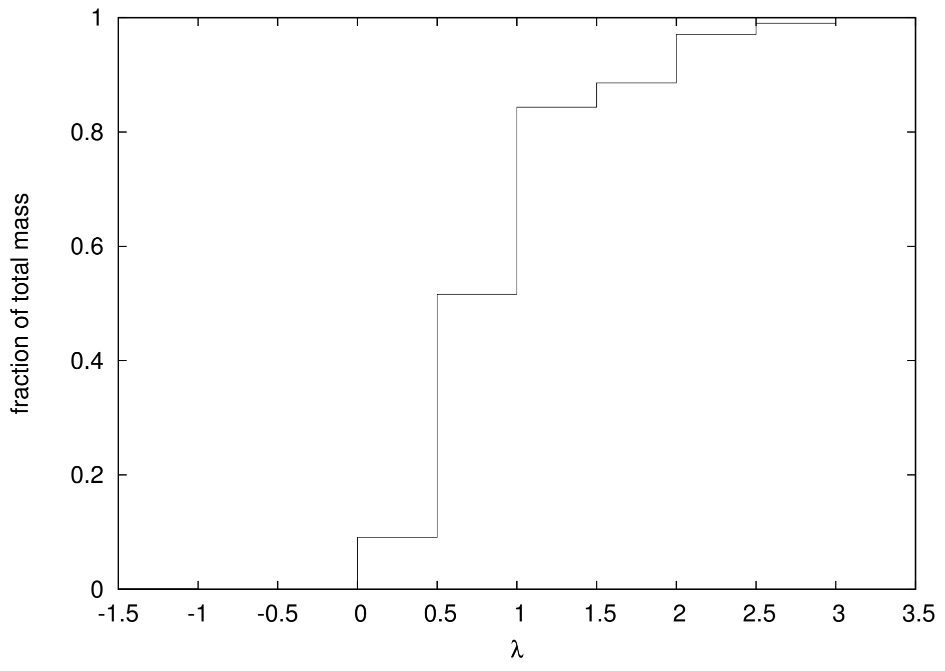

Figure 6: Histogram of total fraction of $\sum_{i=1}^{F} \tilde{\lambda}_i (E_{\tilde{p}}[f_i] - E_{p^*}[f_i])$ mass as function of $\tilde{\lambda}_i$ for letter 5-gram model built on 100 word training set from domain *A*; *i.e.*, this is a graph of $\sum_{i:\tilde{\lambda}_i<\lambda} \tilde{\lambda}_i (E_{\tilde{p}}[f_i] - E_{p^*}[f_i])$ as function of λ. The $\tilde{\lambda}_i$ are bucketed in buckets of width 0.5.

$|\epsilon|$ can be as large as 0.1. While difficult to discern from Figure 5, it is plausible that the average discount for each model in the critical region doesn't stray from the target value of $\gamma = 0.938$ more than this amount.[11]

On the other hand, when $\frac{1}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i|$ is high (corresponding to points to the right in Figure 1), then the average discount in the critical region must be quite close to $\gamma = 0.938$ in order to get good agreement. To investigate this issue further, we examine the ten models from Section 3.1 with the highest $\frac{1}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i|$. These correspond to very sparse models, models built on large vocabularies (domains *E*, *F*, and *G*) with very small training sets and large n-gram orders. We plot the smoothed discounts as a function of $\tilde{\lambda}_i$ for these models in Figure 8. Notably, we find that for these models, the great majority (90%–95%) of their n-grams occur only once in the training data. The average discounts for 1-count features range from 0.911 to 0.949 across models. That is, very sparse models tend to have similar statistics, resulting in similar average discounts.

To recap, the reason that eq. (7) gives low absolute prediction error is different for non-sparse and sparse models. Looking at the first term on the right-hand side of eq. (25), for non-sparse models (*i.e.*, models with low $\frac{1}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i|$), we find there is significant variation in average discounts across models. However, because $\frac{1}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i|$ is low, this results in fairly low absolute error. In contrast, we find that sparse models are all similar statistically, being dominated by single-count n-grams with features whose average discount is quite close to $\gamma = 0.938$. Consequently, their overall average discounts are also near this value, resulting in low absolute prediction error.

[11] In this discussion, we have been assuming that discounts remain constant with respect to $\tilde{\lambda}_i$ (on average), when this is clearly not the case in Figure 5. However, the basic argument that for small models, variations in average discount do not impact absolute prediction error much still holds.

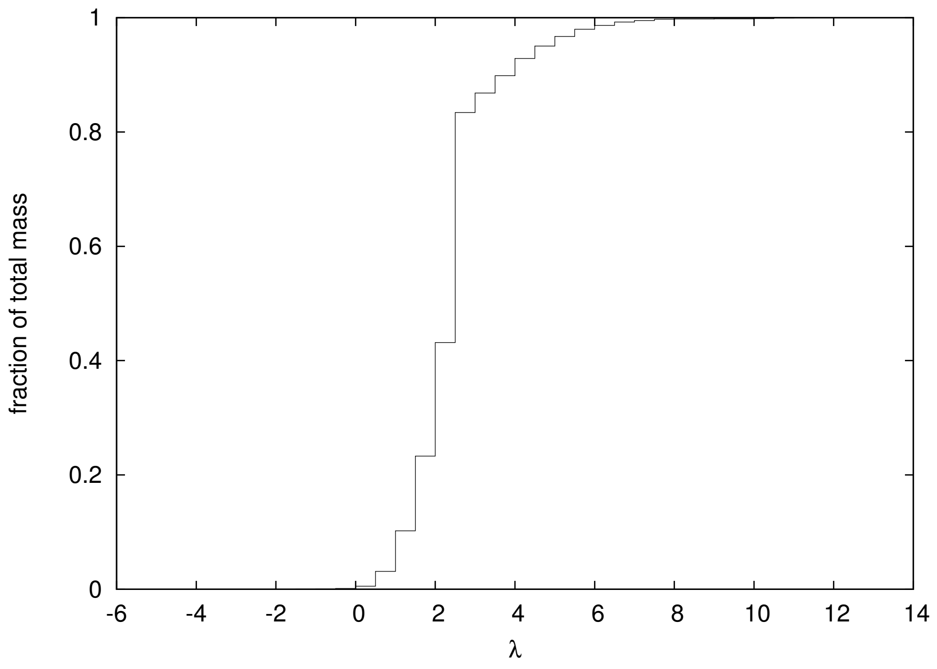

Figure 7: Like Figure 6, except for word trigram model built on 30k sentence training set from domain *E*.

So far, we have ignored the last term on the right-hand side of eq. (25). We can compute this term for various models to determine its significance. For the first ten models considered in this section, these values range from -0.025 nats to -0.258 nats (while the other term on the right-hand side range from 0.330 to 4.694 nats). For the ten sparse models, these values range from -0.131 to -0.283 nats (while the other term range from 5.045 to 6.397 nats). While this normalization correction term is a relatively small factor in the overall performance, some of these values are much larger than the average prediction error of eq. (7), and thus this term plays a significant role in performance prediction.

For exponential n-gram models, it is possible to add a single feature for each n-gram history and choose corresponding parameters λ_i to exactly imitate the effect of normalization (Chen and Rosenfeld, 2000). In this case, the second term on the right-hand side of eq. (25) becomes zero, and we can examine the impact of normalization features on the sum in the first term in the same way that we have been examining the effect of regular features. In Figure 9, we plot smoothed discounts versus $\tilde{\lambda}_i$ for each normalization feature in a single model. Notice that most of these $\tilde{\lambda}_i$ are negative and that the average discount is *positive* unlike for regular features with $\tilde{\lambda}_i < 0$. Thus, in eq. (25), the normalization features are a correction term in the opposite direction of the regular features. We also note that in an n-gram model, (almost all) n'-grams for $n' < n$ have the same counts as histories and as backoff events. That is, many factors $E_{\tilde{p}}[f_i] - E_{p^*}[f_i]$ in the first sum in eq. (25) will occur with both normalization features and regular features. This suggests that the larger the average discounts for regular features, the larger the average discounts for normalization features as well. Since most $\tilde{\lambda}_i$ for normalization features are negative rather than positive, these features will tend to improve agreement with eq. (7) by helping to correct for overly high or overly low average discounts among regular features.

In summary, the fact that eq. (7) predicts performance as well as it does is due to a combination

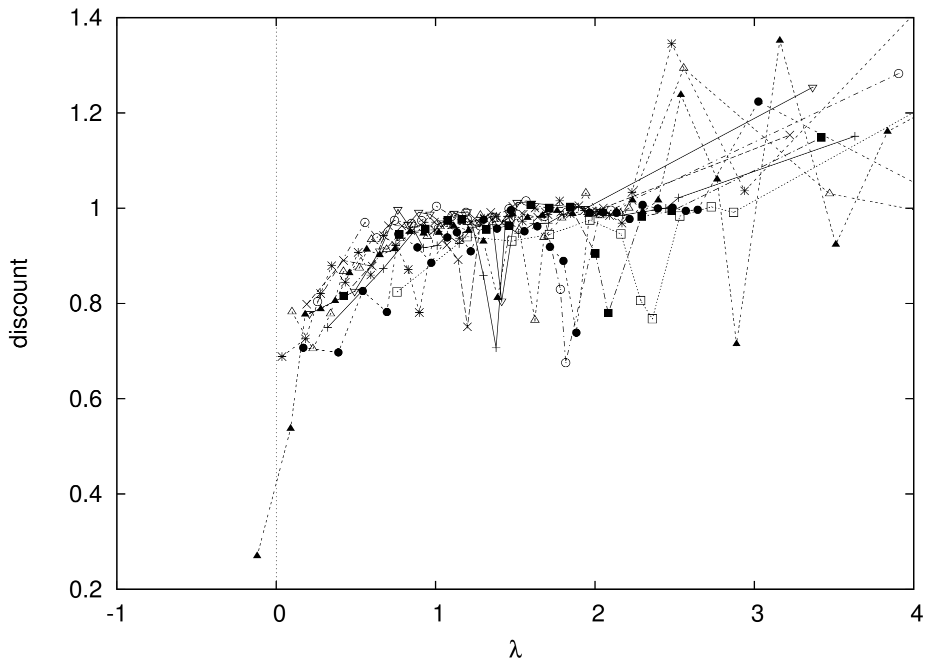

Figure 8: Smoothed graph of discount versus $\tilde{\lambda}_i$ for all features in ten different sparse models. Each smoothed point represents the average of at least 256 raw data points.

of factors. We see that prediction is not at all perfect, but for several reasons, the absolute error in cross-entropy tends to be quite small. One of the key factors why eq. (7) works for n-gram language models is that discounts remains in the general neighborhood of 0.9 or so on average even when varying many model properties as seen in Figure 5. However, we note that this property will not hold for all exponential models. For example, the Good-Turing estimate (Good, 1953) gives us a way to estimate the average discount for a set of features on a given data set, and we can use this knowledge to select a feature set and training set size so that discounts are near zero on average (*e.g.*, by making sure all features have multiple training counts). In this case, training and test performance should be nearly equal by eq. (25). If the model also has a large $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$, then eq. (7) will yield a large absolute prediction error. Thus, while eq. (7) works well for n-gram language models, this equation will not apply to all exponential models.

3.3 The Effect of Regularization on Performance Prediction

In Section 3, we have so far only considered $\ell_1 + \ell_2^2$ regularization with the hyperparameter settings $(\alpha = 0.5, \sigma^2 = 6)$. Here, we consider how performance prediction is affected as we vary regularization settings. In Figure 10, we graph optimism for the evaluation set against $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for each of our models under ℓ_1 regularization with $\alpha = 0.9$; Figure 11 is the same except for ℓ_2^2 regularization with $\sigma^2 = 2.5$. (These are the best hyperparameter settings found for ℓ_1 and ℓ_2^2 regularization in Section 2.3.) For ℓ_1 regularization with $\alpha = 0.9$, the optimal γ on the development set is 1.007 and the root mean squared prediction error on the evaluation set is 0.054 nats. For ℓ_2^2 regularization with $\sigma^2 = 2.5$, the optimal γ is 0.882 and the root mean squared error is 0.139 nats. For reference, the root mean squared error we achieve for $\ell_1 + \ell_2^2$ regularization in Section 3.1 is 0.043 nats. Thus, we see that depending on the type of regularization and the hyperparameter values, there can be a

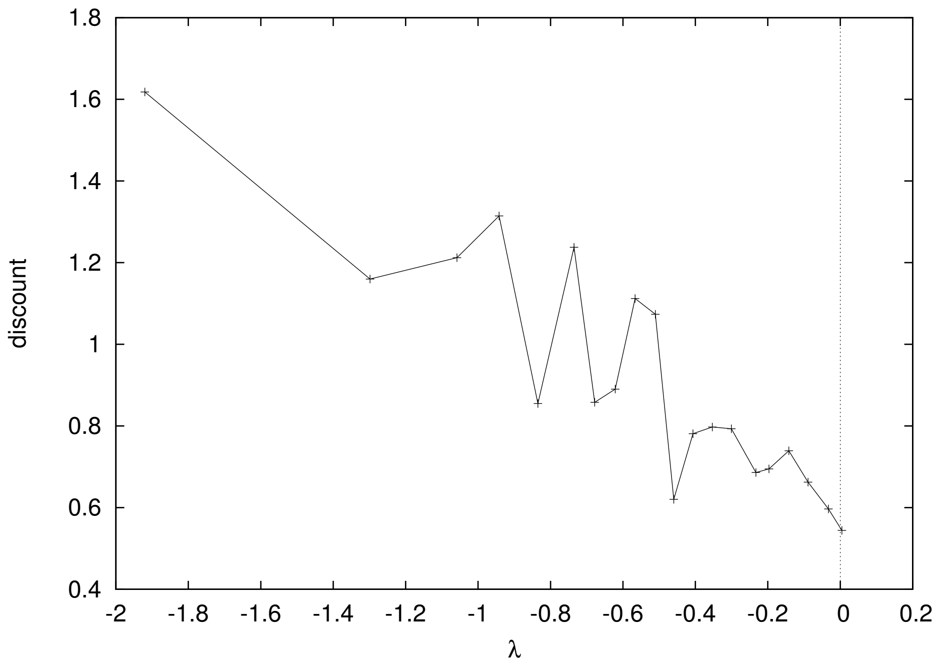

Figure 9: Smoothed graph of discount versus $\tilde{\lambda}_i$ for normalization features in word trigram model built on 30k sentence training set from domain *E*. Each smoothed point represents the average of at least 1024 raw data points.

significant difference in how well we can predict performance using eq. (7). In particular, we find substantially worse correlation with ℓ_2^2 regularization.

In retrospect, we find that choosing hyperparameters carefully in Section 2 was important in doing well in performance prediction. While hyperparameters were chosen to optimize test performance rather than prediction accuracy, we find that the chosen hyperparameters are beneficial for the latter task as well. It may be possible to discover hyperparameters that yield even better prediction accuracy, but at the cost of test performance.

Another point is that we see that the optimal γ value in eq. (7) depends on the type of regularization and the hyperparameter values. Indirectly, this shows that eq. (7) provides accurate performance prediction only for the *regularized* parameter estimates $\tilde{\lambda}_i$ (using the given hyperparameters) and not for arbitrary parameter values λ_i. If parameters are chosen using different hyperparameters (or using some other scheme entirely), then using $\gamma = 0.938$ may not provide very accurate prediction. On the other hand, the γ values we find for different hyperparameters are not hugely different, so it may be the case that eq. (7) is still somewhat accurate for Λ "near" the regularized estimates.

4 N-Gram Models and Backoff Features

While being able to predict n-gram model performance well is interesting, it is unclear why this is useful. In this section, we use performance prediction to explain why backoff features in n-gram models improve performance, and use this analysis to motivate a general heuristic for model design. In an exponential n-gram model, one has features of the form given in eq. (11) for each n'-gram ω in the training data for $n' \leq n$. We refer to features corresponding to n'-grams for n' strictly

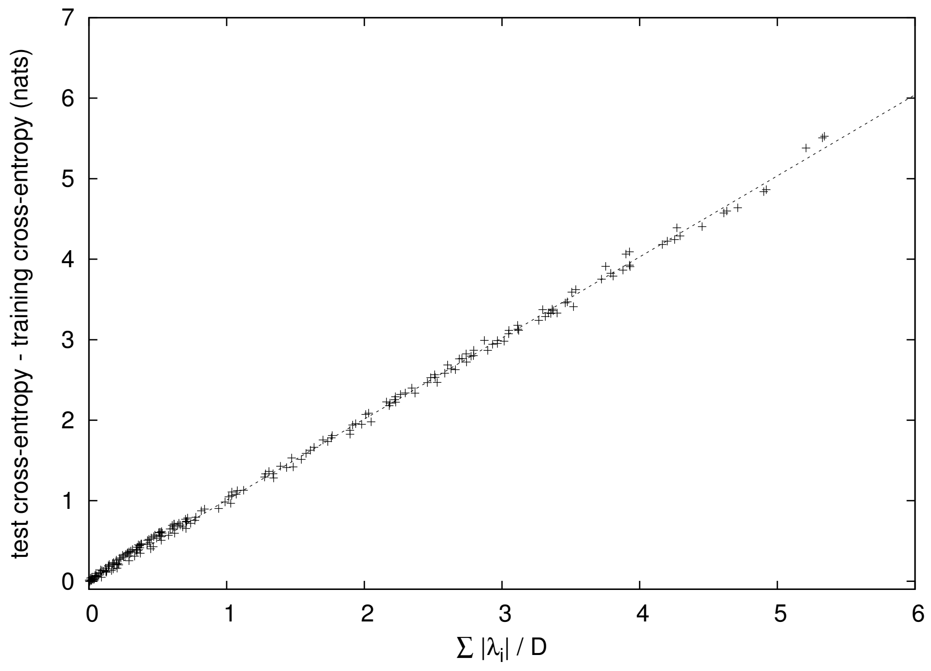

Figure 10: Graph of optimism on evaluation data *vs.* $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for various n-gram models under ℓ_1 regularization, $\alpha = 0.9$. The line represents the predicted optimism according to eq. (7) with $\gamma = 1.007$.

	H_{eval}	H_{pred}	H_{train}	$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$
no backoff	2.681	2.724	2.341	0.408
2g backoff only	2.528	2.513	2.248	0.282
1g+2g backoff	2.514	2.474	2.241	0.249

Table 6: Various statistics for trigram models built in domain *A* on a training set of 1k words. H_{eval} is $\mathcal{H}(p^*, p_{\tilde{\Lambda}})$, the cross-entropy of the evaluation data; H_{pred} is the predicted test cross-entropy according to eq. (7); and H_{train} is $\mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}})$, the training cross-entropy. All H values are in nats.

less than n as *backoff* features. We can use the Akaike Information Criterion and eq. (6) to predict whether backoff features improve test performance. First, we note that the maximum likelihood training set cross-entropy $\mathcal{H}(\tilde{p}, p_{\hat{\Lambda}})$ is the same whether backoff features are present or not. Since a backoff model has more parameters F, AIC predicts that backoff features hurt performance.

In fact, it is well-known that backoff features help performance a great deal (Jelinek and Mercer, 1980), and we analyze this phenomenon using eq. (7). We present statistics in Table 6 for various trigram models built on the same data set.[12] 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 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.

[12]We note that the prediction error from eq. (7) in Table 6 is quite small for n-gram models without backoff features. This is evidence that eq. (7) is accurate for more than just pure n-gram models.

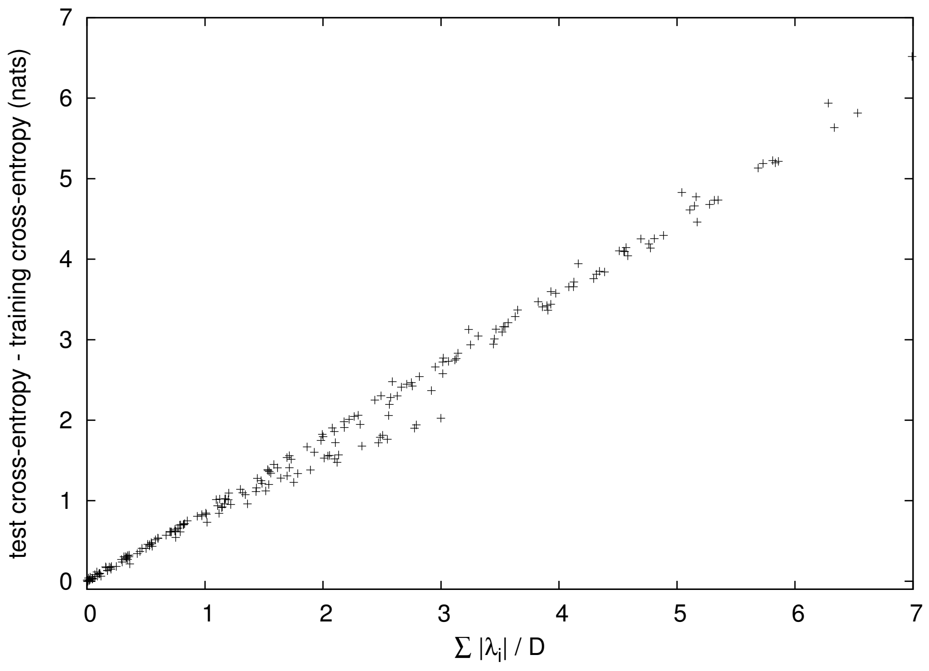

Figure 11: Graph of optimism on evaluation data *vs.* $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for various n-gram models under ℓ_2^2 regularization, $\sigma^2 = 2.5$.

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 (regularized) parameter values $\tilde{\lambda}_1 = 2$ and $\tilde{\lambda}_2 = 4$. From eq. (4), this model has the form

$$p_{\tilde{\Lambda}}(y|x) = \frac{\exp(2f_1(x,y) + 4f_2(x,y))}{Z_\Lambda(x)} \tag{30}$$

Now, consider creating a new feature $f_3(x, y) = f_1(x, y) + f_2(x, y)$ and setting our parameters as follows: $\lambda_1^{\text{new}} = 0$, $\lambda_2^{\text{new}} = 2$, and $\lambda_3^{\text{new}} = 2$. Substituting into eq. (4), we have

$$\begin{aligned} p_{\Lambda^{\text{new}}}(y|x) &= \frac{\exp(0f_1(x,y) + 2f_2(x,y) + 2f_3(x,y))}{Z_\Lambda(x)} &(31)\\ &= \frac{\exp(2f_1(x,y) + 4f_2(x,y))}{Z_\Lambda(x)} &(32)\end{aligned}$$

Notice that the distribution this model describes does not change, and thus neither will its training performance: the parameter mass in λ_3^{new} effectively boosts the parameters of $f_1(x, y)$ and $f_2(x, y)$ by the same amount so the resulting effect of these features is the same.[13] However, the (unscaled) size $\sum_{i=1}^{F}|\lambda_i|$ of the model has been reduced from 2+4=6 to 2+2=4, and consequently by eq. (7) we predict that test performance will improve.[14]

In general, these new parameter values Λ^{new} will not be the regularized parameter estimates for the expanded feature set. In other words, the actual regularized parameter estimates $\tilde{\Lambda}^{\text{new}}$ will have

[13]If we require feature values to be binary, then we need f_1 and f_2 to be non-overlapping; *i.e.*, there is no (x, y) such that $f_1(x, y) \neq 0$ and $f_2(x, y) \neq 0$.

[14]While we have noted that eq. (7) will not be as accurate for parameter values that are not the regularized estimates, we argue in Section 3.3 that eq. (7) should still give reasonable predictions for nearby values.

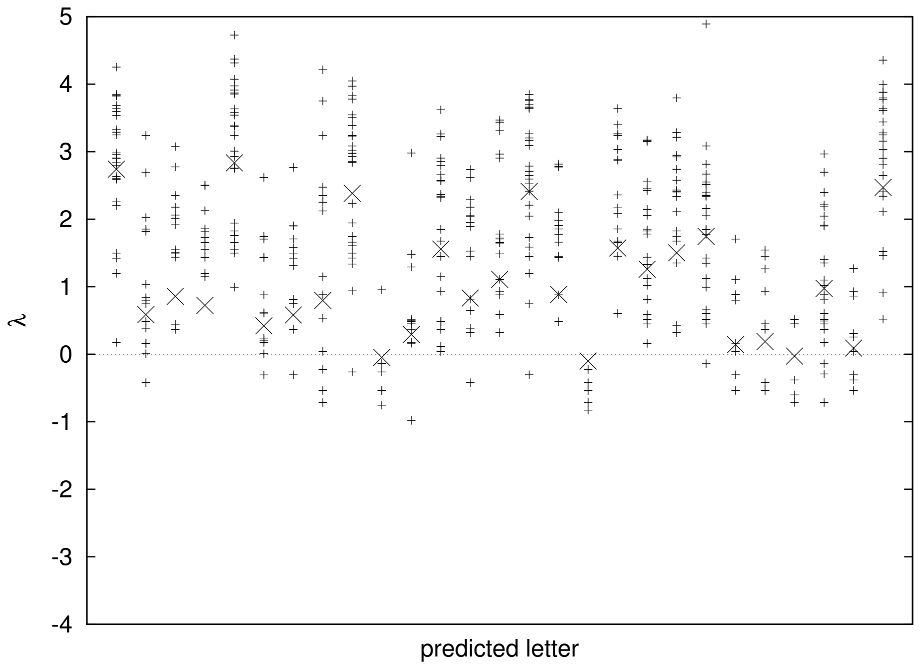

Figure 12: Nonzero $\tilde{\lambda}_i$ values for bigram features grouped by predicted letter (*i.e.*, the second letter in each bigram), for bigram model without unigram backoff features built on domain *A* on training set of 1k words. The large '$\times$' marks represent the average $\tilde{\lambda}_i$ in each column; the average includes history words for which no feature exists or for which $\tilde{\lambda}_i = 0$.

a better score according to the $\ell_1 + \ell_2^2$ regularization objective function given in eq. (10) than Λ^{new}. We note that the right-hand side of eq. (10) is similar to that of eq. (7) and posit that improving the former objective function will also tend to improve the latter; *i.e.*, it is likely that the predicted performance of $\tilde{\Lambda}^{\text{new}}$ will be even better than that of Λ^{new}.

Hence, if we can add "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). Whenever we find a group of features with similar $\tilde{\lambda}_i$ values, adding a new feature that is the sum of these features will tend to shrink the overall $\sum_{i=1}^{F} |\tilde{\lambda}_i|$. The higher the magnitude of the original $\tilde{\lambda}_i$, the larger the gain in predicted performance.

Given this perspective, we can explain why backoff features improve n-gram model performance. For simplicity, let us consider a bigram model, one without unigram backoff features. For a given word w_j, it seems likely that probabilities of the form $p(w_j|w_{j-1})$ are correlated across different w_{j-1}, and thus so are the $\tilde{\lambda}_i$ for the corresponding bigram features. For example, if a word has a high unigram probability, it will also tend to have high bigram probabilities. In Figure 12, 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 w_j, and the large '$\times$' mark in each column is the average value of $\tilde{\lambda}_i$ over all history words w_{j-1}. From the graph, we see that the average $\tilde{\lambda}_i$ for each word w_j is often quite different from zero, which suggests that we can create features of the form

$$f_{w_j}(x, y) = \sum_{w_{j-1}} f_{w_{j-1}w_j}(x, y) \tag{33}$$

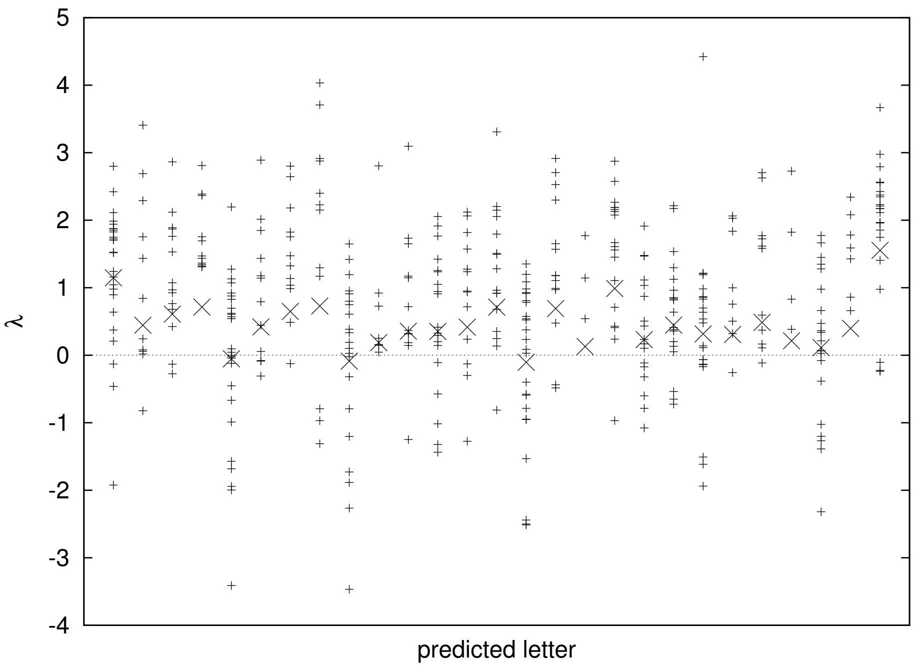

Figure 13: Like Figure 12, but for model with unigram backoff features.

to reduce the overall size of the model (using the notation from eq. (11)).[15] In fact, these features are exactly unigram backoff features. In Figure 13, we plot the nonzero $\tilde{\lambda}_i$ values for all bigram features after adding unigram backoff features. We see that the average $\tilde{\lambda}_i$ values are much closer to zero, implying that the 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.*, we can use bigram features to shrink trigram feature parameters, whose parameters can in turn be shrunk by adding unigram features. As shown in Table 6, both training cross-entropy and model size can be reduced by this technique.

Thus, our analysis suggests the following general technique for improving the (cross-entropy) performance of an exponential model:

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

As noted earlier, if we have the constraint that all features must be binary-valued, then features in the same group cannot overlap. In the next section, we show how we can apply this heuristic to design a novel class-based language model.[16]

5 Class-Based Language Models

In this section, we analyze and evaluate several class-based language models with two main goals in mind. First, we show that eq. (7) can accurately predict performance for class-based models, not just

[15]Features for bigrams that do not occur in the training data will not be included in a model. However, for the purposes of this sum, we pretend they do exist.

[16]One possible way to apply Heuristic 1 is to first train a model and then to group together features based solely on $\tilde{\lambda}_i$ values. We posit that choosing feature groupings in hindsight will not improve performance because $E_{\tilde{p}}[f_i] - E_{p^*}[f_i]$ will tend to be larger on average for such features.

	H_{eval}	H_{pred}	H_{train}	$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$
word n-gram	4.649	4.672	3.354	1.405
model **S**	5.458	5.458	5.430	0.029
model **M**	4.536	4.544	3.296	1.330
model **L**	4.547	4.536	3.145	1.483

Table 7: Various statistics for word and class trigram models built on 100k sentences of WSJ training data. H_{eval} is $\mathcal{H}(p^*, p_{\tilde{\Lambda}})$, the cross-entropy of the evaluation data; H_{pred} is the predicted test cross-entropy according to eq. (7); and H_{train} is $\mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}})$, the training cross-entropy. All H values are in nats. Class models are built with 50 word classes.

for word n-gram models. Secondly, 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.

In class-based modeling, each word w_j in a sentence $w_1 \cdots w_l$ is labeled with a class c_j. Then, we have

$$p(w_1 \cdots w_l) = \sum_{c_1 \cdots c_l} p(c_1 \cdots c_{l+1}, w_1 \cdots w_l) \tag{34}$$

$$= \sum_{c_1 \cdots c_l} \prod_{j=1}^{l+1} p(c_j | c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) \prod_{j=1}^{l} p(w_j | c_1 \cdots c_j, w_1 \cdots w_{j-1}) \tag{35}$$

where c_{l+1} is taken to be a distinguished end-of-sentence token. Here, we assume *hard* classing is employed; *i.e.*, each word w is always mapped to the same class $c(w)$ regardless of context. In this case, we need not perform the sum in the preceding equation; we need only consider the class sequence $c(w_1) \ldots c(w_l)$. The parameterization of the models used to predict c_j and w_j determine the nature of the model, and there are a wide variety of possible choices (Goodman, 2001).

To discuss the possible space of class-based language models, we introduce some notation. We use the notation $p_{\text{ng}}(y|\omega)$ to express an exponential n-gram model as defined in Section 2, where we have features for each suffix of each ωy occurring in the training set. For example, the model $p_{\text{ng}}(w_j | w_{j-1} c_j)$ has features for each n-gram occurring in the training set with one of the following forms: w_j, $c_j w_j$, or $w_{j-1} c_j w_j$. We use the notation $p_{\text{ng}}(y|\omega_1, \ldots, \omega_k)$ to denote a model containing all features in the models $p_{\text{ng}}(y|\omega_1)$, $\ldots$, $p_{\text{ng}}(y|\omega_k)$. For example, $p_{\text{ng}}(c_j | c_{j-2} c_{j-1}, w_{j-2} w_{j-1})$ contains features for n-grams of the forms c_j, $c_{j-1} c_j$, $c_{j-2} c_{j-1} c_j$, $w_{j-1} c_j$, and $w_{j-2} w_{j-1} c_j$.

We say that a model is a class-based n-gram model for some n if the prediction of each word and of each class conditions on words and/or classes at most $n-1$ positions back. Then, we construct the training set events (x, y) for a class-based n-gram model as follows: We assume our raw data consists of a sequence of sentences of the form $w_1 \cdots w_l$ labeled with classes $c_1 \cdots c_l$. (With hard classing, we just take $c_j = c(w_j)$.) For each sentence, we generate $l+1$ class prediction events where $y = c_j$ and $x = c_{j-n+1} \cdots c_{j-1} w_{j-n+1} \cdots w_{j-1}$ for $j = 1, \ldots, l+1$ and l word prediction events where $y = w_j$ and $x = c_{j-n+1} \cdots c_j w_{j-n+1} \cdots w_{j-1}$ for $j = 1, \ldots, l$. We say that an n-gram over words and classes occurs in the training data if the referenced words and classes occur in their associated positions in some event in the training data.

We can define a class-based n-gram model by choosing parameterizations for the distributions $p(c_j | c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1})$ and $p(w_j | c_1 \cdots c_j, w_1 \cdots w_{j-1})$ in eq. (35) above. For example, the most widely-used class-based n-gram model is the one introduced by Brown et al. (1992), which takes

$$\begin{aligned} p(c_j|c_1\cdots c_{j-1}, w_1\cdots w_{j-1}) &= p_{\text{ng}}(c_j|c_{j-2}c_{j-1}) \\ p(w_j|c_1\cdots c_j, w_1\cdots w_{j-1}) &= p_{\text{ng}}(w_j|c_j) \end{aligned} \tag{36}$$

We refer to this model as the IBM class model. (In the original work, non-exponential n-gram models are used.) Clearly, there is a very wide range of class-based n-gram models that one could consider.

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. Then, Heuristic 1 tells us how to take advantage of this observation to improve the performance of a model.

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

$$p(w_j|w_{j-2}w_{j-1}) = \sum_{c_j} p(w_j, c_j|w_{j-2}w_{j-1}) \tag{37}$$

$$= \sum_{c_j} p(c_j|w_{j-2}w_{j-1})p(w_j|w_{j-2}w_{j-1}c_j) \tag{38}$$

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 word 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 features with similar $\tilde{\lambda}_i$. From Heuristic 1, we can make new features

$$f_{w_{j-2}w_{j-1}c_j}(x, y) = \sum_{w_j \in c_j} f_{w_{j-2}w_{j-1}w_j}(x, y) \tag{39}$$

These are exactly the features in our class prediction model (while the features on the right belong to the word prediction model). 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

$$f_{c_{j-2}c_{j-1}c_j}(x, y) = \sum_{w_{j-2}\in c_{j-2}, w_{j-1}\in c_{j-1}} f_{w_{j-2}w_{j-1}c_j}(x, y) \tag{40}$$

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 this model as model **M**, for "medium-sized" model:

$$\begin{aligned} p(c_j|c_1\cdots c_{j-1}, w_1\cdots w_{j-1}) &= p_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \\ p(w_j|c_1\cdots c_j, w_1\cdots w_{j-1}) &= p_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \tag{41}$$

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

For contrast, we consider two other exponential class-based n-gram models. Model **S** is the IBM class model parameterized using exponential models as given in eq. (36). We note that neither component distribution has many parameters as compared to models that directly condition on past words, so we refer to this model as the "small" model. Model **L**, the "large" model, is defined as

$$\begin{aligned} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(c_j|w_{j-2}c_{j-2}w_{j-1}c_{j-1}) \\ p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(w_j|w_{j-2}c_{j-2}w_{j-1}c_{j-1}c_j) \end{aligned} \qquad (42)$$

For both class prediction and word prediction, we condition on all previous classes and words up to two positions back using the most natural backoff order (backing off from nearest to farthest, first class then word). Notice that the class prediction model is a 5-gram model and the word prediction model is a 6-gram model, while in model **M**, the class prediction model consists of features from two trigram models and the word prediction model is a 4-gram model. Thus, we expect model **L** to be significantly larger than model **M**. It is straightforward to extend each of these models from being class trigram models (conditioning on up to two word positions back) to being class 4-gram models (conditioning on up to three word positions back) or higher.

To give an idea of whether these models behave as we expect them to in terms of size and training performance, in Table 7 we provide statistics for these models (as well as for a baseline word n-gram model) built on 100k training sentences with 50 classes. (However, we will later see that how the class models compare to the baseline and to each other will vary greatly as we vary training set size, n-gram order, and the number of classes.) As expected, model **S** has the smallest size $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$, **M** is next smallest, and **L** is the largest. On the other hand, model **L** has the lowest training cross-entropy, **M** the next lowest, and **S** the worst. As compared to the baseline word n-gram model, model **S** performs much worse on the test set. We can explain this from the perspective of eq. (7): its training performance is so poor that even its small size cannot make up for it. This is an example of a model that is too small to model the training data well; we discuss this model further in Section 8.2. On the other hand, while model **L** is larger than the baseline, it models the training data substantially better, and thus achieves better test performance. Finally, we see that model **M** is both smaller than the baseline and has a lower training cross-entropy. This is similar to the behavior we found when adding backoff features to word n-gram models in Section 4, and hints at the potential of Heuristic 1 for improving model performance.

5.1 Predicting Test Set Performance for Class-Based Models

In this section, we evaluate whether eq. (7) can accurately predict test performance for the class-based models **S**, **M**, and **L**. For these experiments, we use the WSJ data and 21k word vocabulary from domain *E*, and consider training set sizes of 1k, 10k, 100k, and 900k sentences. To create word classes, we use the algorithm of Brown et al. (1992) on the largest training set. We create three different classings containing 50, 150, and 500 classes. For each training set size and number of classes (except for the largest training set with model **L** due to resource constraints), we build each of our three class models using $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$, and we build both 3-gram and 4-gram versions of each model. In all three models, word prediction for a word w_j is conditioned on the class c_j in the same position. We constrain that $p(w_j|c_j, ...) = 0$ if $c(w_j) \neq c_j$ as would be expected. Conceptually, we can implement this by pinning λ_i to $-\infty$ for all features of the form $f_{c_j w_j}(x, y), c(w_j) \neq c_j$.

In Figure 14, we plot optimism (*i.e.*, test minus training cross-entropy) versus $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for these models (66 in total) on our WSJ evaluation set. The large points correspond to our class n-gram models, while the small points replicate the points for word n-gram models from Figure 1.

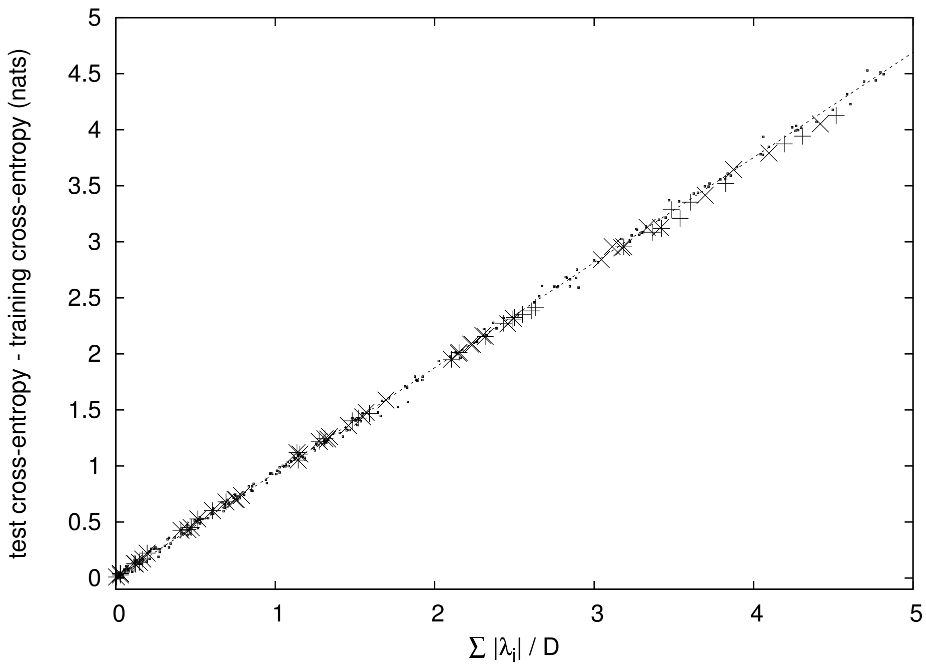

Figure 14: Graph of optimism on evaluation data *vs.* $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for various models. The large points correspond to models **S**, **M**, and **L** for WSJ data over different training set sizes, n-gram orders, and numbers of classes. The small points represent regular word n-gram models over several domains, and are identical to the points from Figure 1. The line represents the predicted optimism according to eq. (7) with $\gamma = 0.938$.

Remarkably, eq. (7) appears to accurately predict performance for many types of class n-gram models using the same $\gamma = 0.938$ value we found for word n-gram models. The mean absolute prediction error is 0.029 nats; the root mean squared error is 0.040 nats; the median absolute error is 0.021 nats; and the maximum absolute error is 0.111 nats. These errors are comparable to those found for word n-gram models in Section 3.1.

It is interesting that eq. (7) works for class-based models despite their differences with word models. The most notable difference is that class models are composed of two submodels, one for word prediction and one for class prediction. However, note that for hard classing, we can remove the sum from eq. (35) and get

$$\log p(w_1 \cdots w_l) = \sum_{j=1}^{l+1} \log p(c_j | c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) + \sum_{j=1}^{l} \log p(w_j | c_1 \cdots c_j, w_1 \cdots w_{j-1}) \tag{43}$$

where $c_j = c(w_j)$. That is, the cross-entropy of data (which is just a scaled log-likelihood) can be decomposed into the sum of the cross-entropy of the word data and the cross-entropy of the class data. It is not surprising that eq. (7) holds separately for the class prediction model predicting the class data and the word prediction model predicting the word data, since each of these component models are basically n-gram models. Summing, this explains why eq. (7) holds for the whole class model.

In fact, one of the component models in the class models is not a pure n-gram model, namely the class prediction model $p_{\text{ng}}(c_j | c_{j-2}c_{j-1}, w_{j-2}w_{j-1})$ in model **M**. This model contains the union

	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**
exponential word n-gram				
3g	**486.6**	268.0	164.3	118.7
4g	491.5	**266.8**	**159.8**	**110.6**
conventional IBM class model				
3g, 50c	428.4	316.2	307.3	307.0
3g, 150c	**381.1**	259.4	232.1	223.1
3g, 500c	389.0	229.5	185.3	164.7
4g, 50c	442.8	320.1	295.3	285.9
4g, 150c	382.5	260.3	226.4	205.5
4g, 500c	388.1	**226.9**	**179.7**	**152.9**
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
model **S**, exponential IBM class model				
3g, 50c	391.5	320.1	309.7	310.1
3g, 150c	**360.9**	259.7	232.6	224.9
3g, 500c	386.6	233.2	184.5	164.9
4g, 50c	399.5	317.2	294.2	287.6
4g, 150c	363.3	258.3	222.4	203.8
4g, 500c	388.5	**230.5**	**177.3**	**150.2**
model **M**, medium model				
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
model **L**, large model				
3g, 50c	352.9	215.6	147.3	
3g, 150c	**351.3**	210.0	145.2	
3g, 500c	393.9	216.2	145.7	
4g, 50c	353.2	214.6	144.7	
4g, 150c	**351.3**	**208.3**	**142.0**	
4g, 500c	396.8	214.5	142.4	

Table 8: WSJ perplexity results. The best performance for each training set size for each model type is highlighted in bold.

of features found in two separate n-gram models. Referring back to Section 3.2, we hypothesize that the distribution of discounts with respect to $\tilde{\lambda}_i$ in this model is very similar to that for a pure n-gram model, and so eq. (7) still applies.

5.2 Perplexity Comparison of Various Class-Based Models

In this section and the next, we compare how our novel class-based model, model **M**, stacks up against state-of-the-art class-based models, both exponential and non-exponential, in terms of perplexity and speech recognition word-error rate. We use the same training sets and models as in the last section, but construct new development and evaluation sets from the Wall Street Journal CSR Corpus (Doddington, 1992; Paul and Baker, 1992). We extract all of the "verbalized punctuation" data from the training and test portions of this corpus. From this, we randomly select 2439 utterances (46888 words), each generated from a unique prompt ID, as our evaluation set. From the remaining verbalized punctuation data, we select 977 utterances (18279 words) as our development set. The development set is used to tune interpolation weights for interpolated models and to tune the acoustic weight used in the speech recognition experiments.[17]

We compare the following model types: conventional (*i.e.*, non-exponential) word n-gram models; exponential word n-gram models; conventional IBM class n-gram models; conventional IBM class n-gram models interpolated with conventional word n-gram models (Brown et al., 1992); model **S** (*i.e.*, an exponential IBM class model); model **M**; and model **L**. We build both 3-gram and

[17]Unlike in the performance prediction experiments, we no longer satisfy the assumption that the training and test data come from the same underlying distribution, as is the typical case in practice.

4-gram models of each type for each of the training set sizes of 1k, 10k, 100k, and 900k sentences (except for model **L** on the largest training set size). For the class models, we do runs using 50, 150, and 500 classes. All conventional (word and class) n-gram models are smoothed with modified Kneser-Ney smoothing (Chen and Goodman, 1998), except for the models $p(w_j|c_j)$ in the IBM class models. These models are unsmoothed except that words with no count are given a count of 1 (with the appropriate class). All exponential models are regularized with $\ell_1 + \ell_2^2$ regularization with the hyperparameters $(\alpha = 0.5, \sigma^2 = 6)$. *Note:* Because word classes are constructed using only the largest training set, results for word models and class models are not directly comparable except for the largest training set. Results for different class models are comparable across all data sets. Note that the conventional class model interpolated with the conventional word model represents that most popular state-of-the-art class-based model in the literature; it is also the only model here that uses the development set to tune interpolation weights. Interpolation weights are chosen to optimize the perplexity of the development set.

We display the perplexities of these models on the evaluation set in Table 8. As expected, the performances of conventional and exponential word n-gram models are quite similar, as are the performances of the conventional and exponential IBM class models except on the smallest training set. We attribute this difference to the primitive smoothing used in the conventional model for $p(w_j|c_j)$. As mentioned above, word and class model results are incomparable except for the largest training set. On this training set, the IBM class models are much worse than the word n-gram models, but the interpolated class model is slightly better as is consistent with previous results, *e.g.*, (Brown et al., 1992; Martin et al., 1995).

Next, we compare the other class models with the state-of-the-art interpolated class model. As expected, the interpolated IBM class model outperforms the IBM class model alone, both conventional and exponential, across the board. If we compare the best performance for model **L** at each training set size with that of the interpolated class model, we see that the interpolated model appears slightly better. Model **M** performs best of all (even without interpolating with a word n-gram model), outperforming the interpolated model on 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 the performance of these models in applications, and we investigate speech recognition performance in the next section.

The most widely-used baseline models in the language modeling literature are conventional word trigram models. On the largest training set, model **M** achieves an 18% reduction in perplexity (99.1 *vs.* 121.5) with respect to this baseline.[18]

5.3 Word-Error Rate Comparison of Various Class-Based Models

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. 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 and then LDA is performed to yield 40-dimensional feature vectors. 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 then pruned to contain a total of about

[18]The most widely-used smoothing method in the literature for baseline n-gram models is Katz smoothing (Katz, 1987), not modified Kneser-Ney smoothing as used here. We report results for Katz smoothing in Table 12.

	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%**
exponential word n-gram				
3g	**34.6%**	**30.4%**	25.7%	22.5%
4g	**34.6%**	30.7%	**25.6%**	**22.2%**
conventional IBM class model				
3g, 50c	32.5%	30.4%	30.0%	30.1%
3g, 150c	31.7%	29.0%	27.8%	27.4%
3g, 500c	32.3%	28.6%	26.0%	24.4%
4g, 50c	32.5%	30.4%	29.5%	29.3%
4g, 150c	**31.6%**	28.9%	27.5%	26.5%
4g, 500c	32.4%	**28.5%**	**25.8%**	**23.9%**
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
model **S**, exponential IBM class model				
3g, 50c	31.6%	30.3%	29.9%	30.1%
3g, 150c	**31.2%**	28.8%	27.6%	27.3%
3g, 500c	32.2%	**28.5%**	25.8%	24.3%
4g, 50c	31.7%	30.1%	29.4%	29.1%
4g, 150c	31.3%	28.8%	27.3%	26.1%
4g, 500c	32.4%	**28.5%**	**25.6%**	**23.6%**
model **M**, medium model				
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%
model **L**, large model				
3g, 50c	**31.0%**	27.3%	23.9%	
3g, 150c	31.1%	**27.2%**	23.9%	
3g, 500c	32.5%	27.8%	24.1%	
4g, 50c	**31.0%**	27.4%	23.8%	
4g, 150c	**31.0%**	27.3%	**23.6%**	
4g, 500c	32.4%	28.0%	23.9%	

Table 9: 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.

350k n-grams using the algorithm of Stolcke (1998). The silence token is treated as a transparent word with probability 0.1. The word-error rate of the first-pass decoding is 27.7% and the oracle error rate of the generated lattices is 9.6%.

We choose the acoustic weight for each model to optimize the lattice rescoring error rate of that model on the development set; we do a Powell search to find the best weight with a granularity of 0.005. The minimum weight selected is 0.055, and the maximum weight selected is 0.085. Generally, the better the language model, the lower the optimal acoustic weight. In rescoring, we also treat the silence token as a transparent word with probability 0.1.

In Table 9, we display the word-error rates for various models. We focus on the performance of model **M**, which achieved the best perplexities in the last section and which achieves the best word-error rates here. If we compare the best performance of model **M** at each training set size 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 modest perplexity gains of model **M** over the interpolated model; as has been observed earlier, perplexity is 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 conventional word trigram model by 1.8% absolute. We situate these results with respect to other results reported for class-based language models in Section 7.2.

One natural question to ask is that if perplexity is not an accurate predictor of word-error rate and application performance is our ultimate goal, why does it make sense to predict and optimize test set perplexity (or equivalently, cross-entropy)? We note that while improved perplexity does

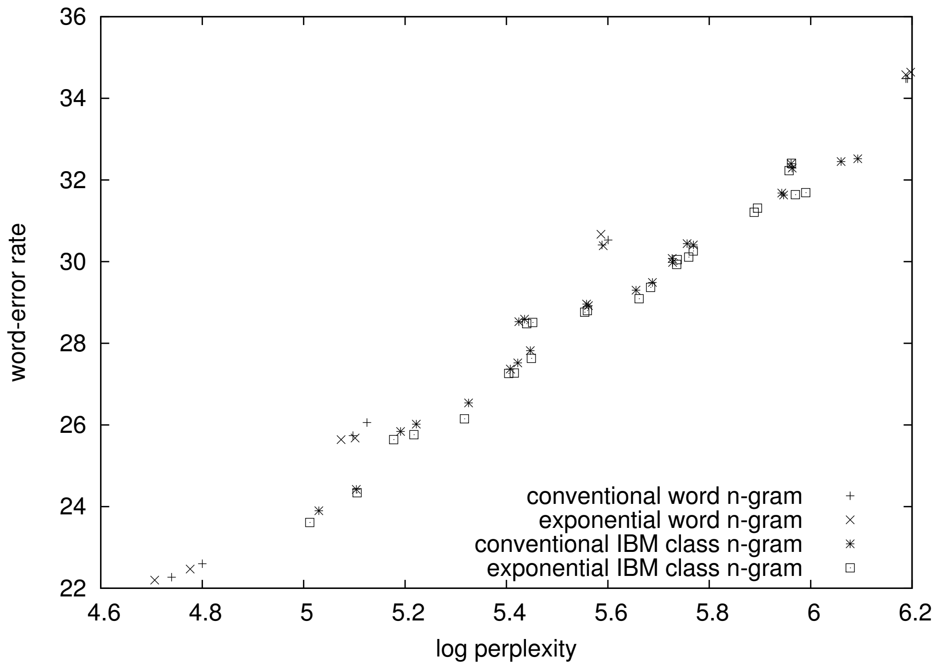

Figure 15: Graph of word-error rate *vs.* perplexity on WSJ data set for conventional and exponential versions of word n-gram models and IBM class models for different training set sizes, n-gram orders, and numbers of classes.

not necessarily lead to improved word-error rate when comparing models of different types, word-error rate is generally monotonic in perplexity when considering different models of the *same* type, *e.g.*, (Chen and Goodman, 1998). In Figures 15 and 16, we plot word-error rate *vs.* perplexity for all of the evaluated models. For a given model type, word-error rate is largely monotonic in perplexity; however, some model types tend to have better word-error rates than others given the same perplexity. For example, in Figure 15, we see that the exponential versions of the IBM class model tend to have slightly better word-error rates for the same perplexity as compared to the conventional versions. In Figure 16, we see that models **M** and **L** tend to have significantly better word-error rates for the same perplexity as compared to the interpolated IBM class model.

6 Domain Adaptation

In this section, we introduce another heuristic for improving exponential models and show how this heuristic can be used to motivate minimum discrimination information (MDI) models for domain adaptation (Della Pietra et al., 1992). We show that eq. (7) can accurately predict cross-entropy performance for MDI models and compare this method against other adaptation techniques in both perplexity and word-error rate.

To motivate this new heuristic, we reexamine eq. (25). We note that if $E_{\tilde{p}}[f_i] - E_{p^*}[f_i] = 0$ for a feature f_i, then the feature does not affect the difference between test and training cross-entropy (ignoring its impact on the last term). If we can somehow add features with $E_{\tilde{p}}[f_i] - E_{p^*}[f_i] \approx 0$ to a model such that training performance remains unchanged while the remaining features shrink in size, then test performance should be improved.

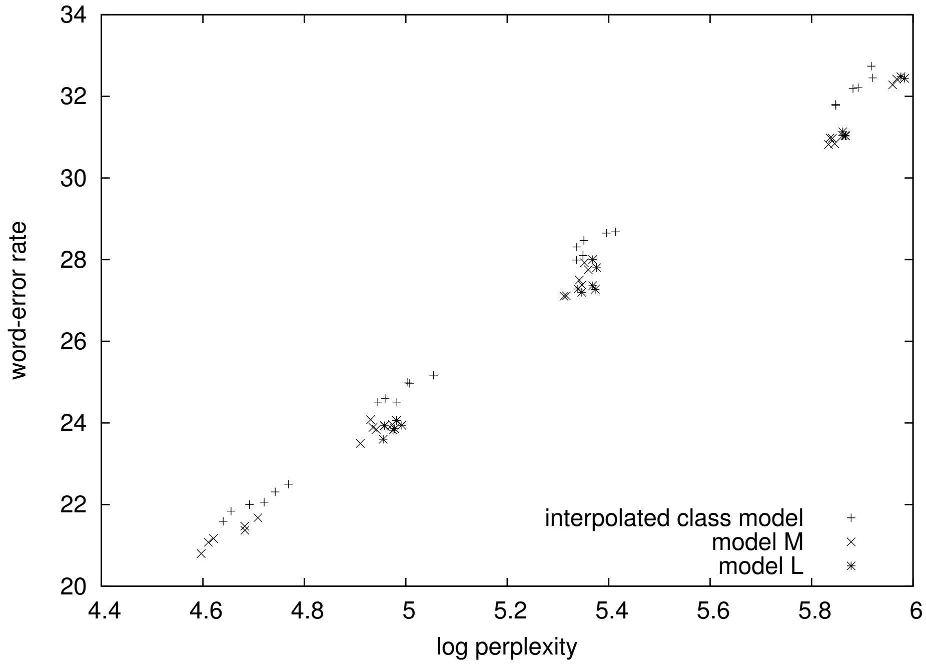

Figure 16: Graph of word-error rate *vs.* perplexity on WSJ data set for interpolated IBM class model and models **M** and **L** for different training set sizes, n-gram orders, and numbers of classes.

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 $p'_{\Lambda^{\text{new}}}$ as follows:

$$p'_{\Lambda^{\text{new}}}(y|x) = q(y|x)\frac{\exp(\sum_{i=1}^{F}\lambda_i^{\text{new}} f_i(x,y))}{Z_{\Lambda^{\text{new}}}(x)} \tag{44}$$

Then, let us choose Λ^{new} such that $p'_{\Lambda^{\text{new}}}(y|x) = p_{\tilde{\Lambda}}(y|x)$ 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 $p'_{\Lambda^{\text{new}}}$ to be less than that of $p_{\tilde{\Lambda}}$. Since they describe the same distribution, their training cross-entropy will be the same. If eq. (7) still holds for this scenario, then we expect $p'_{\Lambda^{\text{new}}}$ to have better test performance than $p_{\tilde{\Lambda}}$.

For this analysis to hold, we need to show that $E_{\tilde{p}}[f_i^q] - E_{p^*}[f_i^q] \approx 0$ for features f_i^q in q (assuming q is an exponential model), so that it is safe to ignore these features in eqs. (25) and (7). We note that q is derived from an independent data set and by assumption, the training and test set for p come from the same distribution. It follows that we expect $E_{\tilde{p}}[f_i^q] - E_{p^*}[f_i^q]$ to be zero for all features in q. We empirically validate that eq. (7) holds for these types of models later in this section. As in Section 4, in general, the parameter values Λ^{new} will not be the regularized parameter estimates for p', and we hypothesize that optimizing the regularization objective function for p' will improve test performance further.

In summary, this analysis suggests the following method for improving the performance of an exponential model:

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

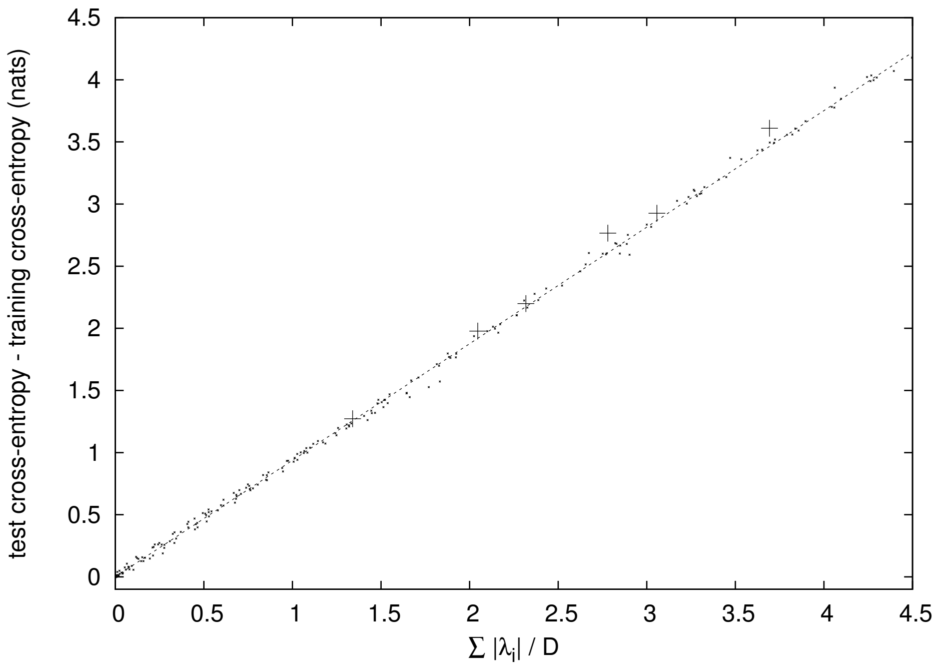

Figure 17: Graph of optimism on evaluation data *vs.* $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for various models. The large points correspond to MDI models over different n-gram orders and in-domain training set sizes. The small points represent regular word n-gram models over several domains, and are identical to the points from Figure 1. The line represents the predicted optimism according to eq. (7) with $\gamma = 0.938$.

While we do not define what it means for a distribution to be similar except by whether it empirically shrinks the size of the original model and/or improves test performance, this heuristic can still be used effectively in practice.

For example, 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 task 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 then 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.

To evaluate whether eq. (7) holds for exponential language models with prior distributions, we use the same WSJ training and evaluation sets from domain *E* as in Section 5.1. Our out-of-domain data is the 100k sentence Broadcast News training set from domain *F*. We consider three different training set sizes for our in-domain (WSJ) data: 1k, 10k, and 100k sentences. First, we build an exponential n-gram model on the Broadcast News data using the regularization hyperparameters found in Section 2.2. Then, we use this Broadcast News model as the prior model $q(y|x)$ in eq. (44) when building an exponential n-gram model on the in-domain data (again, using the same regularization hyperparameters). We repeat the same process for both trigram models and 4-gram models.

In Figure 17, we plot test minus training cross-entropy versus $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for these models on our WSJ evaluation set. The large points correspond to the adapted n-gram models, while the small

	H_{eval}	H_{pred}	H_{train}	$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$
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 10: Various statistics for trigram models built on WSJ training data. The first column is the size of the (in-domain) training set in sentences. For the MDI models, an n-gram model built on 100k sentences of Broadcast News data is used as a prior. H_{eval} is $\mathcal{H}(p^*, p_{\tilde{\Lambda}})$, the cross-entropy of the evaluation data; H_{pred} is the predicted test cross-entropy according to eq. (7); and H_{train} is $\mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}})$, the training cross-entropy. All H values are in nats.

points replicate the points for word n-gram models from Figure 1. As expected, eq. (7) appears to work quite well for MDI models using the same $\gamma = 0.938$ value as before. The mean absolute prediction error is 0.077 nats; the root mean squared error is 0.095 nats; the median absolute error is 0.058 nats; and the maximum absolute error is 0.159 nats. These errors are somewhat larger than those found for word n-gram models in Section 3.1, but the correlation between optimism and model size is still very good.

In Table 10, 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 $\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$ and have better training cross-entropy than the unadapted models built on the same data. By eq. (7), it seems likely that the adapted models will have better test performance and we verify this in the next section.

6.1 Comparison of Various Domain Adaptation Methods

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 training sets as before, but we use the development and evaluation data sets from Section 5.2. As before, the development set is used to tune interpolation weights for interpolated models and to tune the acoustic weights used in the speech recognition experiments. Interpolation weights are chosen to optimize the perplexity of the development set and acoustic weights are chosen to optimize word-error rate. We use the same speech recognition system and lattice rescoring setup as in Section 5.3.

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 (Jelinek et al., 1991). In count merging, the 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 (Iyer et al., 1997; Bacchiani et al., 2006). The in-domain data set may be replicated several times to more heavily weight this data. Finally, we also consider the baseline of not using the out-of-domain data at all, *i.e.*, just building an n-gram model on the in-domain data.

In Table 11, we display perplexity and word-error rates for all of the models under consideration, for both trigram and 4-gram models and with varying amounts of in-domain training data. In all

	in-domain training (sents.)		
	1k	10k	100k
in-domain data only			
3g	488.4	270.6	168.2
4g	**486.8**	**267.4**	**163.6**
count merging			
3g	503.1	290.9	170.7
4g	**497.1**	**284.9**	**165.3**
interpolated model			
3g	328.3	234.8	162.6
4g	**325.3**	**230.8**	**157.6**
MDI model			
3g	296.3	218.7	157.0
4g	**293.7**	**215.8**	**152.5**

	in-domain training (sents.)		
	1k	10k	100k
in-domain data only			
3g	**34.5%**	30.5%	26.1%
4g	**34.5%**	**30.4%**	**25.7%**
count merging			
3g	30.4%	28.3%	**25.2%**
4g	**30.0%**	**28.0%**	25.3%
interpolated model			
3g	**30.3%**	28.5%	25.8%
4g	**30.3%**	**28.4%**	**25.2%**
MDI model			
3g	30.0%	28.0%	**24.9%**
4g	**29.6%**	**27.9%**	**24.9%**

Table 11: WSJ perplexity and lattice rescoring results for domain adaptation models. The first three models are composed of modified Kneser-Ney-smoothed n-gram models; the last model is an exponential model using $\ell_1 + \ell_2^2$ regularization. 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.

experiments, 100k sentences of out-of-domain (Broadcast News) data is used. The last model 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 the word-error rate on the evaluation set and improves the perplexity only slightly. For example, with three copies of the in-domain data, the trigram model perplexity improves from 503.1 to 498.8 when using 1k sentences of in-domain data and from 290.9 to 280.0 when using 10k sentences.

Looking first at perplexity, we see that count merging does even worse than the baseline of not using any out-of-domain training data. Both interpolated and MDI models are substantially better than the baseline, with MDI outperforming 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, all three adaptation techniques do significantly better than the baseline. Despite the poor showing in perplexity, count merging does at least as well as interpolation across all training sets. MDI models perform best of all, outperforming interpolation on each training set by 0.3–0.7% absolute, and outperforming count merging by 0.1–0.4% absolute.[19]

7 Related Work

Here, we discuss related word in performance prediction, class-based language modeling, and domain adaptation.

7.1 Performance Prediction

We group performance prediction methods into two categories: *non-data-splitting* methods and *data-splitting* methods. In non-data-splitting methods, test performance is directly estimated from

[19]We note that the in-domain data contains verbalized punctuation while in the Broadcast News data, punctuation is deleted. Thus, results on data sets where this mismatch is not present may be somewhat different. On the other hand, the fact that MDI works well under these conditions is evidence of its robustness.

training set performance and/or other statistics of a model. Data-splitting methods involve partitioning training data into two parts, a truncated training set and a surrogate test set. The performance on the surrogate test set of a model estimated from the truncated training set (perhaps averaged over several splits) is taken as a proxy for the true test performance of that model.

The most popular non-data-splitting methods for predicting test set cross-entropy (or likelihood) are the Akaike Information Criterion (AIC) and variants such as AIC$_c$, quasi-AIC (QAIC), and QAIC$_c$ (Akaike, 1973; Akaike, 1974; Hurvich and Tsai, 1989; Lebreton et al., 1992); other related methods include Mallows' C_p for least squares regression and the Takeuchi Information Criterion (TIC) (Mallows, 1973; Takeuchi, 1976). As seen in eq. (6), AIC can be used to predict the test cross-entropy of a model from the number of model parameters and its training cross-entropy (using maximum likelihood parameter estimates). However, as noted earlier, maximum likelihood parameter estimates do not generally perform well in language modeling. In Section 3.1, we considered performance prediction formulae with the same form as AIC and AIC$_c$ (except using regularized parameter estimates), and neither performed nearly as well as eq. (7); *e.g.*, see Table 4 and Figure 2.

There are many techniques for bounding test set classification error including the Occam's Razor bound (Blumer et al., 1987; McAllester, 1999), PAC-Bayes bound (McAllester, 1999), and the sample compression bound (Littlestone and Warmuth, 1986; Floyd and Warmuth, 1995). These methods derive theoretical guarantees that the true error rate of a classifier will be below (or above) some value with a certain probability. Langford (2005) evaluates these techniques over many data sets; while the bounds can sometimes be fairly tight, in many data sets the bounds are quite loose.

When learning an element from a set of target classifiers, the Vapnik-Chervonenkis (VC) dimension of the set can be used to bound the true error rate relative to the training error rate with some probability (Vapnik and Chervonenkis, 1971; Vapnik, 1998); this technique has been used to compute error bounds for many types of classifiers. Extensions of this method include methods that bound the true error rate based on the fat-shattering dimension of a set of target classifiers, *e.g.*, (Bartlett, 1998), and methods that bound error based on the training set margins of a classifier, *e.g.*, (Schapire et al., 1998). Bartlett (1998) shows that for a neural network with small weights and small training set squared error, the true error depends on the size of its weights rather than the number of weights; this finding is similar in spirit to eq. (7).

One key difference between our work and other work is that the vast majority of previous non-data-splitting methods use theoretical analysis to derive their predictions of test performance. Here, we developed eq. (7) on a purely empirical basis.

Because of the connection between performance prediction and model selection, we briefly discuss Bayesian methods for model selection. In the Bayesian paradigm, one determines a prior distribution $p(p_\Lambda)$ over model structures and/or parameters. From this, we can compute the posterior probability of a model given training data:

$$p(p_\Lambda|\mathcal{D}) \propto p_\Lambda(\mathcal{D})p(p_\Lambda) \tag{45}$$

Then, one can select the single most probable model or model class as in maximum *a posteriori* (MAP) estimation.[20] The most popular model selection methods of this type are probably the Bayesian Information Criterion (Schwarz, 1978) and the Minimum Description Length principle (Rissanen, 1978). We note that the right hand side in eq. (7) has the same form as the ℓ_1 regularization objective function in eq. (8) and that ℓ_1 regularization is an instance of MAP estimation. Thus, eq. (7) suggests the use of a particular prior distribution in Bayesian estimation. However, we note

[20] Another approach is Bayesian Model Averaging in which a model is produced by interpolating candidate models weighted by their posterior probability (Leamer, 1978).

	training set (sents.)			
	1k	10k	100k	900k
3g	579.3	317.1	196.7	137.5
4g	592.6	325.6	202.4	136.7

	training set (sents.)			
	1k	10k	100k	900k
3g	35.5%	30.7%	26.2%	22.7%
4g	35.6%	30.9%	26.3%	22.7%

Table 12: WSJ perplexity and lattice rescoring word-error rate results for Katz-smoothed word n-gram models.

that the goal of eq. (7) is accurate performance prediction, while the point of MAP estimation is to find the most *likely* model. As these two criteria are very different, it is unclear why we should expect that a "prior" that works well for one objective is also suitable for the other.

In practice, the most accurate methods for performance prediction in most contexts are data-splitting methods (Guyon et al., 2006). These techniques include the hold-out (or split-sample) method; leave-one-out and k-fold cross-validation; and bootstrapping (Allen, 1974; Stone, 1974; Geisser, 1975; Stone, 1977; Craven and Wahba, 1979; Stone, 1979; Efron, 1983; Efron, 1993; Kohavi, 1995; Shao, 1997). In the hold-out method, a single split of the training data is performed and performance on the held-out set is taken as an estimate of test performance. In the other methods, performance is averaged over multiple data splits. While accurate, the methods that involve multiple splits can be computationally expensive since models have to be rebuilt for each split. More importantly, unlike non-data-splitting methods, these methods do not lend themselves well to providing insight into model design as discussed in Section 8.2.

7.2 Class-Based Language Models

Since many previous papers use Katz-smoothed word n-gram models as baselines (Katz, 1987), we compute perplexities and word-error rates for Katz-smoothed models on our data sets from Section 5.2 and display them in Table 12. Another issue is that as we built word classes using only our largest data set, our class model results are comparable with our word model results only for this data set. To address this, we also build word classes on our 100k sentence (2.6M word) training set, and compute perplexities and word-error rates for model **M** with these new classes and this training set using the best settings found previously (4-gram model, 150 classes). This results in a perplexity of 143.5 and a word-error rate of 24.1% (as compared to 135.7 and 23.5% using the original classes). The perplexity and word-error rate of model **M** using our 900k sentence (23M word) training set (from Tables 8 and 9) are 99.1 and 20.8%, respectively. This translates to reductions in perplexity of 27% and 28% over a Katz-smoothed word trigram baseline for our 100k-sentence and 900k-sentence training sets, respectively, and corresponding reductions in word-error rate of 2.1% and 1.9% absolute.

In the following sections, we survey the most closely-related class-based language models in the literature. To situate the quality of our results, we also survey the best perplexity and speech recognition word-error rate results reported for class-based language models relative to conventional word n-gram model baselines. While many different data sets are used in previous work so that most results are not directly comparable, we find that our results are very competitive with the existing results in the literature. The largest reported reductions in perplexity we found over a Katz-smoothed trigram model are 53% for a SuperARV language model (Wang and Harper, 2002) and about 25% for the model *fullibmpredict* proposed by Goodman (2001). The largest reported decrease in speech recognition word-error rate we found over a Katz-smoothed trigram model is 2.2% absolute for a multi-class composite language model (Yamamoto et al., 2003); almost all other improvements are

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, Katz				
3g	**24.2%**	**20.2%**	**16.1%**	13.3%
4g	24.3%	20.3%	16.2%	**13.2%**
conventional word n-gram, modified KN				
3g	23.6%	19.4%	**15.3%**	12.7%
4g	**23.4%**	**19.3%**	**15.3%**	**12.5%**
interpolated IBM class model				
3g, 50c	20.9%	17.7%	14.8%	12.6%
3g, 150c	20.6%	17.1%	14.2%	12.3%
3g, 500c	21.3%	17.1%	**13.9%**	12.1%
4g, 50c	21.1%	17.5%	14.7%	12.2%
4g, 150c	**20.5%**	17.3%	14.2%	11.9%
4g, 500c	21.2%	**17.0%**	**13.9%**	**11.8%**
model **M**, medium model				
3g, 50c	**19.9%**	16.7%	13.8%	11.8%
3g, 150c	20.0%	**16.2%**	**13.1%**	11.7%
3g, 500c	21.4%	17.2%	13.5%	11.5%
4g, 50c	20.0%	16.6%	13.6%	11.4%
4g, 150c	20.0%	16.6%	13.2%	11.3%
4g, 500c	21.7%	17.3%	13.4%	**11.2%**

Table 13: WSJ lattice rescoring results with improved acoustic models; 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 30 errors.

considerably smaller. Before we continue, we first analyze whether it makes more sense to compare absolute word-error rate reductions or relative ones.

7.2.1 Comparing Relative and Absolute Word-Error Rate Differences Between Language Models

In the language modeling literature, some papers report relative word-error rate improvements while other papers report absolute word-error rate improvements. In our discussion of related work, we report all gains as absolute word-error rate differences, as absolute word-error rate improvements for a language model tend to be relatively stable regardless of the baseline word-error rate, whereas relative word-error rate improvements are very sensitive to the baseline word-error rate. To demonstrate this effect, we construct an additional acoustic model using about nine hours of training data from the Wall Street Journal CSR Corpus (disjoint from our evaluation set built from this corpus). Because of the small amount of acoustic data we have from this source, our acoustic training set includes our original development set, and we create a new development set (900 utterances, 17342 words) and new evaluation set (1539 utterances, 29546 words) by randomly splitting our original evaluation set into two parts. The development set is used to tune the acoustic weight used with each language model. The new acoustic model is a cross-word triphone system with 386 context-dependent states and a total of 28487 Gaussians; the front end and lattice generation language model is the same as used in Section 5.3. The word-error rate of the first-pass decoding is 16.7% and the oracle error rate of the generated lattices is 7.0%. Despite the smaller amount of acoustic training data and smaller size, this acoustic model has substantially better performance than our Broadcast

News acoustic model since it is trained on data from the same source as the test data.

We display lattice rescoring word-error rates for several language models in Table 13. These results are directly comparable to those in Tables 9 and 12 except for the change in acoustic model; the language models being evaluated are identical. Note that all error rates have shifted down around 10% absolute; this suggests that absolute word-error rate differences between models are about the same, while relative word-error rates differences are much changed. For example, the word-error rate difference between the best and worst modified Kneser-Ney-smoothed word n-gram models is 12.2% absolute in Table 9 (34.5% *vs.* 22.3%), while in Table 13 this difference is 11.1% absolute (23.6% *vs.* 12.5%). Translated to relative reductions in word-error rate, the reduction changes from 35% to 47%. To give another example, the difference between model **M** on the largest training set and a Katz-smoothed word trigram model is 1.9% absolute and 2.1% absolute for the two acoustic models, while the relative reductions in error rate are 8% and 16%; *i.e.*, the relative reduction doubles when the baseline word-error rate is approximately halved. Clearly, relative error rate reductions depend strongly on the baseline error rate, so absolute error rate improvements are a better representation of language model performance.

7.2.2 Class-Based Language Models with Separate Models for Class and Word Prediction

We survey related work by category. The most common type of class-based language model describes a joint distribution over word and class sequences, where this distribution is decomposed into separate conditional models for word prediction and class prediction as in eq. (35). In *soft* clustering, a word may belong to several different classes. In *hard* clustering, each word w is always mapped to the same class $c(w)$ regardless of context and we need not perform the sum in eq. (35); we need only consider the class sequence $c(w_1) \ldots c(w_l)$.

The most widely-used class model is the model we refer to as the IBM class model (Brown et al., 1992; Ney et al., 1994). For this model, we assume

$$\begin{aligned} p(c_j | c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) &\approx p(c_j | c_{j-2} c_{j-1}) \\ p(w_j | c_1 \cdots c_j, w_1 \cdots w_{j-1}) &\approx p(w_j | c_j) \end{aligned} \tag{46}$$

where the distributions $p(c_j | c_{j-2} c_{j-1})$ and $p(w_j | c_j)$ are parameterized as conventional n-gram models. (In our discussion of related work, we describe only the trigram versions of each model.) In the original formulation of this model, hard clustering is used. While this model can be superior to word n-gram models for small training sets, it performs poorly with larger training sets unless interpolated with a word n-gram model.

Martin et al. (1995) implement the IBM class model and compare different algorithms for inducing word classes. On Wall Street Journal training sets ranging in size from 1M words to 38M words, they achieve perplexity gains of up to 12% with an interpolated class model over a baseline word trigram model. Niesler et al. (1998) propose a variant of the IBM class model with soft clustering and variable-length class n-grams in the class prediction model. Using a 37M word WSJ training set, they achieve gains over a baseline Katz-smoothed trigram model of up to 13% in perplexity and 1.1% absolute in N-best list rescoring word-error rate with an interpolated class n-gram model. Samuelsson and Reichl (1999) propose an interpolated IBM class model where classes are taken to be part-of-speech ambiguity classes and another model where a word n-gram model is modified to backoff to an IBM class model. Using a training corpus of 37M words of WSJ data, a perplexity improvement of 4% is achieved over a Katz-smoothed trigram baseline as well as a 0.7% absolute reduction in word-error rate.

Another variant of IBM class models are *multi-class composite* n-gram models (Yamamoto and Sagisaka, 1999; Deligne, 2000; Deligne and Sagisaka, 2000; Isogai et al., 2001; 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, rather than using a single clustering across all word positions. In experiments on the ATR spoken language database (Takezawa et al., 1998), Yamamoto et al. (2003) report a reduction in perplexity of 9% and an increase in word accuracy of 2.2% absolute over a baseline Katz-smoothed trigram model.

Instead of making the assumptions made by the IBM class model in eq. (46), one can make the weaker assumption that word predictions and class predictions depend only on words and classes up to two positions back (for trigram models), *i.e.*,

$$\begin{aligned} p(c_j|c_1\cdots c_{j-1},w_1\cdots w_{j-1}) &\approx p(c_j|c_{j-2}c_{j-1},w_{j-2}w_{j-1}) \\ p(w_j|c_1\cdots c_j,w_1\cdots w_{j-1}) &\approx p(w_j|c_{j-2}c_{j-1}c_j,w_{j-2}w_{j-1}) \end{aligned} \tag{47}$$

Many parameterizations of $p(c_j|c_{j-2}c_{j-1},w_{j-2}w_{j-1})$ and $p(w_j|c_{j-2}c_{j-1}c_j,w_{j-2}w_{j-1})$ have been proposed. Our models **M** and **L** fall in this category.

Heeman and Allen (1997) describe a model where these two distributions are parameterized as decision-tree language models (Bahl et al., 1989) and the word classes are taken to be part-of-speech tags. In preliminary experiments on WSJ data using a training set of 79k words, a 15% reduction in perplexity over a Katz-smoothed trigram baseline is achieved (Heeman, 1998).

The most closely-related class-based language model to model **M** is one of the models described by Goodman (2001). This paper compares many different class-based language models and the best performing individual model is called *fullibmpredict* and has the following form for trigram models:

$$\begin{aligned} p(c_j|c_{j-2}c_{j-1},w_{j-2}w_{j-1}) &= \lambda\,p(c_j|w_{j-2}w_{j-1})+(1-\lambda)\,p(c_j|c_{j-2}c_{j-1}) \\ p(w_j|c_{j-2}c_{j-1}c_j,w_{j-2}w_{j-1}) &= \mu\,p(w_j|w_{j-2}w_{j-1}c_j)+(1-\mu)\,p(w_j|c_{j-2}c_{j-1}c_j) \end{aligned} \tag{48}$$

where each component model is a conventional n-gram model. This is similar to model **M** except that linear interpolation is used to combine information from word and class histories, whereas we combine word history features and class history features within a single exponential model. In addition, there is no analog to the final term in eq. (48) in our model. One of our contrast models, model **L**, is identical to another of the models described in this paper, *indexpredict*, except that exponential n-gram models are used for the component models instead of conventional n-gram models. (Another difference is that *fullibmpredict* and *indexpredict* are multi-class models, while our models use a single clustering for all word positions.)

Goodman (2001) runs experiments over a range of training set sizes using the North American Business news corpus (Stern, 1996). The largest reduction in perplexity achieved by a single class-based model over the Katz-smoothed trigram model baseline is about 25%, by the model *fullibmpredict* on a training set of 1M words. Speech recognition word-error rate results are reported for the largest training set (284M words) using N-best list rescoring. The best result for an individual class-based model is an 0.5% absolute reduction in word-error rate (over a baseline error rate of 9.8%).

The SuperARV language model (Wang and Harper, 2002; Wang et al., 2002; Wang et al., 2004) is another class-based model in this category, where classes are based on *abstract role values* as given by a Constraint Dependency Grammar (Harper and Helzerman, 1995). The distributions in eq. (47) are n-gram models that back off to a variety of mixed word/class histories in a specific order. Using a Wall Street Journal training set of about 930k words, a reduction in perplexity of about 29% is achieved over a Katz-smoothed trigram model baseline. Using a training set of about 37M words, a perplexity reduction of up to 53% is achieved as well as a decrease in word-error rate of up to 1.0% absolute.

7.2.3 Class-Based Language Models that Jointly Predict a Word and Class as a Single Unit

Models in this category also describe a joint distribution over word and class sequences, but individual words and classes are predicted jointly as a single unit:

$$\begin{aligned} p(w_1 \cdots w_l) &= \sum_{c_1 \cdots c_l} p(c_1 \cdots c_{l+1}, w_1 \cdots w_l) && (49) \\ &= \sum_{c_1 \cdots c_l} \prod_{j=1}^{l+1} p((c_j, w_j) | c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) && (50) \end{aligned}$$

Galescu and Ringger (1999) describe a model of this type where part-of-speech tags are used as classes; the model is a conventional n-gram model except over word/tag pairs instead of words. Using a 5M word WSJ training set, a reduction in perplexity of about 5% is achieved over a Witten-Bell-smoothed trigram baseline (Witten and Bell, 1991).

Cui et al. (2007) also predict a word and its part-of-speech tag as a single unit, but instead of using a conventional n-gram model for prediction, they use an exponential model that includes features for many different types of word/tag histories. Using a WSJ training set of 1M words, they achieve perplexity reductions of up to 15% over a modified Kneser-Ney-smoothed trigram model. With a 3M word Switchboard training set, a word-error rate reduction of up to 0.4% absolute is achieved over a modified Kneser-Ney-smoothed 4-gram model.

7.2.4 Class-Based Language Models Consisting Only of a Word Prediction Model

In the final category of class-based language models we discuss, there is no class prediction model. Instead, class information is used only in histories:

$$\begin{aligned} p(w_1 \cdots w_l) &= \prod_{j=1}^{l+1} p(w_j | w_1 \cdots w_{j-1}) && (51) \\ &= \prod_{j=1}^{l+1} p(w_j | f(w_1 \cdots w_{j-1})) && (52) \end{aligned}$$

Typically, hard clustering is used, and the function f includes information about the class assignments of words in the history.

Dupont and Rosenfeld (1997) propose *lattice-based* language models. Given a hierarchical word clustering, a lattice of n-gram models of the form $p(w_j | c_{j-n+1} \ldots c_{j-1})$ is constructed for different n-gram orders and for classes at different levels in the class hierarchy. Then, the model is a word n-gram model where backoff occurs along the edges in the lattice to the models at each node. A perplexity reduction of 6% is achieved with a 2.5M word Switchboard training set over a baseline word trigram model.

Blasig (1999) describes a class-based model called a *category/word varigram* model. This model is similar to a word n-gram model except backoff can be performed to arbitrary sequences of words and word classes. Using a 39M word WSJ training set, a reduction of about 10% in perplexity and about 0.8% absolute in word-error rate is achieved as compared to a baseline word varigram model (Kneser, 1996).

Whittaker and Woodland (2001) introduce *one-sided* class models of the form $p(w_j|c_{j-2}c_{j-1})$ (for trigram models). This model is parameterized as a conventional n-gram model. When interpolated with a word n-gram model, one-sided class models yield about the same performance as interpolated IBM class models in both perplexity and word-error rate, but require less computation for clustering (Whittaker and Woodland, 2003).

Zitouni et al. (2003) propose *hierarchical* class n-gram models. This model is like a word n-gram model except instead of backing off to successively shorter word histories, backoff is performed by successively abstracting each word position to a more general class in a class hierarchy, from farthest word to nearest. Zitouni (2007) reports results on Wall Street Journal data with a 56M word training set. A perplexity decrease of 6% is achieved over a baseline Katz-smoothed trigram model; a word-error rate decrease of 0.9% to 1.2% absolute is achieved, depending on the vocabulary size. A perplexity decrease of about 2% is achieved using a Switchboard training set of 3.5M words with a CALLHOME test set. Wang and Vergyri (2006) extend this technique by using part-of-speech information in building the class hierarchy. On an Arabic CALLHOME corpus with a training set of about 180k words, their "Model II" achieves a perplexity decrease of about 8% over a baseline word trigram model. In single-pass N-best list rescoring, gains of up to 1.2% absolute are achieved over the baseline; when combined with a word n-gram model and a factored language model (Vergyri et al., 2004), gains increase to 1.7% absolute.[21]

7.3 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 discussion of techniques using word class, topic, syntax, or semantic information, an extensive survey of existing techniques for language model adaptation is provided by 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, and it has been used in a vast number of papers since then. Another popular method is count merging; this has been motivated as an instance of MAP adaptation (Federico, 1996; Masataki et al., 1997; Chen and Huang, 1999). With respect to perplexity, linear interpolation tends to perform well while count merging does not. In terms of word-error rate, both methods improve over the baseline of not using out-of-domain data. Compared to each other, Iyer et al. (1997) found linear interpolation to give better speech recognition performance while Bacchiani et al. (2006) found count merging to be superior.

Della Pietra et al. (1992) introduce the idea of minimum discrimination information distributions. Given some prior model $q(y|x)$ (*e.g.*, representing out-of-domain data), 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 for each linear constraint with $q(y|x)$ as its prior as in eq. (44). While this method has been used many times for language model adaptation, *e.g.*, (Kneser et al., 1997; Federico, 1999), MDI models have not performed as well as linear interpolation in perplexity or word-error rate in previous work (Rao et al., 1995; Rao et al., 1997). One of the issues present when using this technique is how to select the feature set (or equivalently, the linear constraints) specifying the model. With a small amount of adaptation data, one should intuitively use a small feature set, *e.g.*, just containing

[21]Wang and Vergyri (2006) also report *all-pass* rescoring results in which a language model is used in multiple decoding passes, *e.g.*, for acoustic model adaptation. In this case, language modeling gains can be magnified across the multiple passes.

unigram features. However, the use of regularization can obviate the need for intelligent feature selection. For example, 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.

In its original formulation, MDI models do not have a parameter analogous to the mixture weight in interpolation or the in-domain replication factor in count merging. On one hand, having a tunable parameter can improve test performance when one's test data does not come from the same distribution as one's training data. On the other hand, this simplifies the implementation of MDI models and we find that MDI is still able to outperform the other methods despite this difference.

8 Discussion

8.1 Regularization and Performance Prediction

This paper makes contributions in several areas. First, we demonstrate the effectiveness of $\ell_1 + \ell_2^2$ regularization (Kazama and Tsujii, 2003) in language modeling. We show that this type of regularization can outperform ℓ_1 and ℓ_2^2 regularization alone by up to several percent in perplexity for n-gram models. In addition, we show that a single hyperparameter setting, $(\alpha = 0.5, \sigma^2 = 6)$, works well for n-gram models across varying domains, vocabulary sizes, training set sizes, and n-gram orders. As compared to optimizing hyperparameter values for each individual model, we lose less than 0.01 nats (about 1% in perplexity) on average as compared to using this single setting. As optimizing regularization hyperparameters can be expensive, this finding has significant practical benefit. In addition to being effective for word n-gram models, we find that $\ell_1 + \ell_2^2$ regularization appears to be effective for class-based models and domain adaptation models as well. While ℓ_1 regularization and ℓ_2^2 regularization are used very widely, $\ell_1 + \ell_2^2$ regularization is much less popular, and these results suggest that further investigation of this regularization scheme is warranted.

Next, we show that for several types of exponential language models, it is possible to accurately predict the cross-entropy of test data using the simple relationship given in eq. (7). When using $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$, the value $\gamma = 0.938$ works well across varying domains, vocabulary sizes, training set sizes, and n-gram orders, yielding a mean absolute error of about 0.03 nats (3% in perplexity). We evaluate over 300 language models in total, including word n-gram models, class n-gram models, and n-gram models with prior distributions. We show that the optimal value of γ depends on the regularization hyperparameters, and that we get better prediction accuracy using $\ell_1 + \ell_2^2$ regularization as opposed to ℓ_1 or ℓ_2^2 regularization alone.

We also provide an explanation for why this relationship holds. We show in eq. (25) that the difference in test and training performance can be expressed as a simple function of $\tilde{\Lambda}$ and the differences in feature expectations between the training and test data. We find that eq. (7) is not a perfect relationship, but that the absolute error tends to be quite small due to a combination of factors. Sparse models tend to be dominated by single-count n-grams whose average discount is fairly close to $\gamma = 0.938$. For non-sparse models, the average discount varies more, but because this is effectively multiplied by the size of the model $\frac{1}{D}\sum_{i=1}^{F} |\tilde{\lambda}_i|$ which by assumption is small, the overall absolute error is small. While eq. (7) holds across several types of exponential language models, it won't hold for all exponential models. For example, performance prediction will be poor for extremely non-sparse models with large $\frac{1}{D}\sum_{i=1}^{F} |\tilde{\lambda}_i|$.

One interesting connection is that to optimize test performance in parameter estimation, we have found that $\ell_1 + \ell_2^2$ regularization works best, while to optimize performance prediction accuracy (*e.g.*, for model selection), we have found that an expression with the same form as the ℓ_1 regularization objective function works best. It would have been quite elegant if these two criteria were the same; then, the process of parameter estimation could seemingly be viewed as directly optimizing test performance. However, this would not actually be the case, as our performance prediction formula is accurate only for the regularized parameter estimates and not for arbitrary parameter values.

While there has been a great deal of work in performance prediction, the great majority of work on non-data-splitting methods has focused on finding theoretically-motivated approximations or probabilistic bounds on test performance. In contrast, we developed eq. (7) on a purely empirical basis, and there has been little, if any, existing work that has shown comparable performance prediction accuracy over such a large number of models and data sets. In addition, there has been little, if any, previous work on performance prediction for language modeling.[22] The most closely-related performance prediction method to ours is probably AIC (Akaike, 1973). However, AIC is based on maximum-likelihood parameter estimates, which perform poorly in language modeling, while our method works with regularized parameter estimates. Empirically, we find our method to be much more accurate than AIC-like measures for prediction, and unlike AIC, we correctly predict that backoff features improve the performance of n-gram models.

While eq. (7) performs well as compared to other non-data-splitting methods for performance prediction, the prediction error can be several percent in perplexity, which means we cannot reliably rank models that are close in quality. In addition, data-splitting methods are the most accurate performance prediction methods in many contexts (Guyon et al., 2006). Finally, in speech recognition and many other applications of language modeling, an external test set is typically provided, which means we can measure the relevant test set performance directly rather than estimate it from training statistics. Thus, in practice, eq. (7) is not terribly useful for the task of model selection, one of the main applications of performance prediction. However, what eq. (7) gives us is insight into *model design*.

8.2 Model Design

The task of model selection deals primarily with selecting between candidate models *once they have been built*. However, it can be expensive to implement various models, and thus it is desirable to be able to select between models at the *model design* stage. Being able to intelligently compare models (without implementation) requires that we know which aspects of a model impact test performance, and this is exactly what eq. (7) (and other criteria for non-data-splitting performance prediction) tell us. While we may not be able to accurately guess what the training cross-entropy or size $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ of a model is, at least we have a framework for thinking about these issues, and this is certainly better than trying to guess evaluation performance in a vacuum.

Intuitively, simpler models should perform better on test data given equivalent training performance, and model structure (as opposed to its parameter values) is an important aspect of the complexity of a model. Accordingly, there have been many methods for model selection that measure the size of a model in terms of the number of features or parameters in the model, *e.g.*, (Akaike, 1973; Rissanen, 1978; Schwarz, 1978). Surprisingly, for exponential language models, the number of model parameters seems to matter not at all; all that matters are the magnitudes of the parameter values, as evidenced by both eq. (25) and the accuracy of eq. (7). Consequently, one can improve ex-

[22] Here, we refer to predicting test set performance from training set performance and other model statistics. However, there has been a good deal of work on predicting test set speech recognition word-error rate from test set perplexity and other statistics, *e.g.*, (Klakow and Peters, 2002).

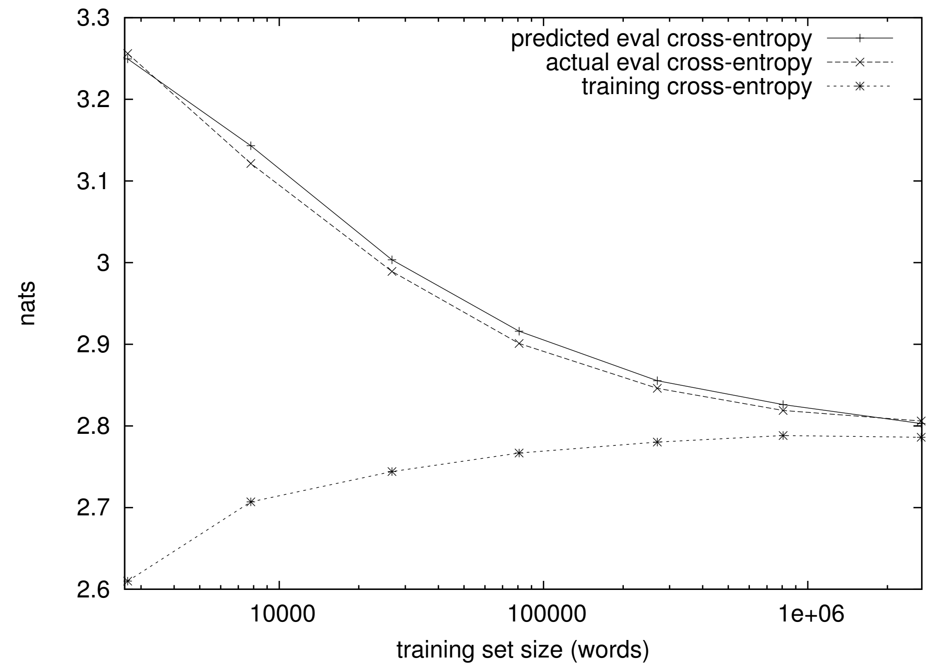

Figure 18: Graph of training set cross-entropy as well as predicted and actual evaluation set cross-entropy *vs.* training set size for bigram models from domain *C* (WSJ, 300 word vocabulary).

ponential language models by adding features (or a prior model) that reduce parameter values while maintaining training performance, and from this observation we develop Heuristics 1 and 2. These are general techniques that can be used to develop novel exponential models as well as explain the performance of existing ones.

We can also gain insight into other aspects of model design, *e.g.*, the impact of training set size on model design. In Figures 18 and 19, we plot how training set performance and evaluation set performance vary according to training set size for two different WSJ models: bigram models built on a 300 word vocabulary, and 5-gram models built on a 21k word vocabulary. The former represents a non-sparse model, while the latter is rather sparse.

One somewhat unintuitive point is that training performance generally *degrades* as training set sizes increase; the reason that evaluation performance improves is solely because the model size decreases. To understand the reason why, consider the case where the target distribution is a uniform unigram distribution. Clearly, the more training data there is, the more uniform the estimated model becomes, and the worse the training set performance is (as non-uniform models can achieve better performance on non-uniform data than a uniform model on uniform data). Intuitively, training cross-entropy increases with more training data because the model cannot overfit the data as much.

What this tells us is that for a model with a fixed number of parameters, evaluation performance will essentially stop improving once the training set size passes some point, and that this point will be reached when the model size per event $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ becomes small enough. In practice, once $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ becomes less than a few hundredths of a nat, evaluation cross-entropy likely won't improve significantly with more training data. We see this effect occurring for the trigram version of model **S** with 50 classes in Table 8; performance saturates with a training set of 100k sentences. As shown in Table 7, the size of model **S** on this training set is only 0.029 nats/event. Once this

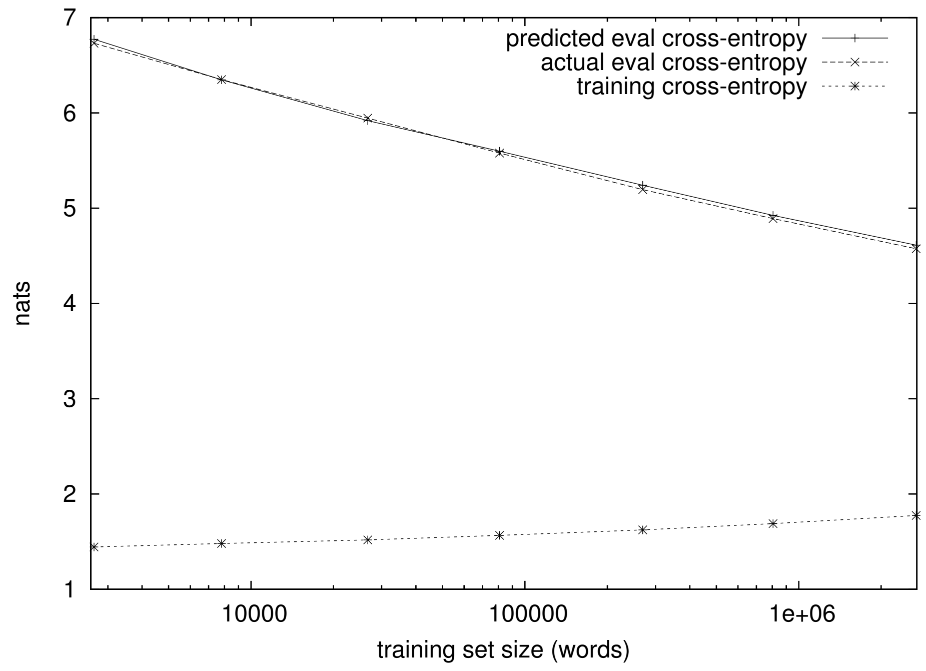

Figure 19: Graph of training set cross-entropy as well as predicted and actual evaluation set cross-entropy *vs.* training set size for 5-gram models from domain *E* (WSJ, 21k word vocabulary).

point is reached, to improve test performance it is necessary to improve training performance in some way, *e.g.*, by increasing the model size. For example, one can increase the model order n if using an n-gram model; the number of classes if using a class-based model; or one can have separate features for separate subdomains of the training data as in Section 6.

Not only can we derive general observations about model design from eq. (7), but we can also apply these observations to better understand various aspects of language modeling and improve language modeling performance. Using this relationship, we show how aspects of language modeling as diverse as backoff n-gram features, class-based models, and domain adaptation can all be explained as a simple tradeoff between training performance and model size. We can frame performance improvements in all of these areas as methods that decrease model size without degrading training set performance. Notably, using Heuristic 1, we developed a novel and simple class-based language model that achieves perplexity and speech recognition word-error rate improvements competitive with the best reported results for class-based models in the literature.

Clearly, there is much possible follow-on work. While we have shown that eq. (7) applies to several types of exponential language models, we have only partially characterized the types of models that it does *not* apply to. In addition, it would be interesting to see whether these ideas can be extended to non-exponential model types or to any other types of loss functions such as 0-1 loss. However, we note that eq. (25) is specific to exponential models under a log loss function, so this is unclear. On the other hand, there are likely many other types of exponential language models, as well as exponential models outside of language modeling, where eq. (7) holds. For all of these models, eq. (7) gives us a way of explaining their test performance in terms of training performance and model size, and we can use techniques like Heuristics 1 and 2 to improve them. All in all, eq. (7) provides a valuable and important new framework for characterizing, analyzing, and designing statistical models.

References

Hirotugu Akaike. 1973. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, pages 267–281.

Hirotugu Akaike. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

David M. Allen. 1974. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16(1):125–127.

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

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. 1989. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:1001–1008, July.

Peter L. Bartlett. 1998. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536.

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

Reinhard Blasig. 1999. Combination of words and word categories in varigram histories. In *Proceedings of ICASSP*, pages 529–532, Washington, DC, USA. IEEE Computer Society.

Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. 1987. Occam's razor. *Information Processing Letters*, 24(6):377–380.

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.

Kenneth P. Burnham and David R. Anderson. 2002. *Model Selection and Multi-Model Inference: A Practical Information-Theoretic Approach*. Springer, second edition.

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

Langzhou Chen and Taiyi Huang. 1999. An improved MAP method for language model adaptation. In *Proceedings of Eurospeech*, pages 1923–1926.

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.

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

Kenneth W. Church and William A. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54.

Peter Craven and Grace Wahba. 1979. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403.

Jia Cui, Yi Su, Keith Hall, and Frederick Jelinek. 2007. Investigating linguistic knowledge in a maximum entropy token-based language model. In *Proceedings of ASRU*, pages 171–176.

Sabine Deligne and Yoshinori Sagisaka. 2000. Statistical language modeling with a class-based n-multigram model. *Computer Speech and Language*, 14(3):261–279.

Sabine Deligne. 2000. Statistical language modeling with a class based n-multigram model. In *Proceedings of ICSLP*, volume 3, pages 119–122.

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.

George R. Doddington. 1992. CSR corpus development. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 363–366, Morristown, NJ, USA. Association for Computational Linguistics.

Miroslav Dudík and Robert E. Schapire. 2006. Maximum entropy distribution estimation with generalized regularization. In *Proceedings of the 19th Annual Conference on Learning Theory*.

Pierre Dupont and Ronald Rosenfeld. 1997. Lattice based language models. Technical Report CMU-CS-97-173, Carnegie Mellon University.

Bradley Efron. 1983. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association*, 78(382):316–331.

Bradley Efron. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, London.

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

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

Sally Floyd and Manfred Warmuth. 1995. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304.

Lucian Galescu and Eric Ringger. 1999. Augmenting words with linguistic information for n-gram language models. In *Proceedings of Eurospeech*.

Seymour Geisser. 1975. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70:320–328.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, volume I, pages 517–520, March.

I.J. Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264.

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

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proceedings of NAACL*.

David Graff. 1997. The 1996 Broadcast News speech and language-model corpus. In *Proceedings of the DARPA Speech Recognition Workshop*, pages 11–14.

Isabelle Guyon, Amir Saffari, Gideon Dror, and Joachim Buhmann. 2006. Performance prediction challenge. In *Proceedings of International Conference on Neural Networks (IJCNN06), IEEE World Congress on Computational Intelligence (WCCI06)*, pages 2958–2965, July.

Mary P. Harper and Randall A. Helzerman. 1995. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9:187–234.

Peter A. Heeman and James F. Allen. 1997. Incorporating POS tagging into language modeling. In *Proceedings of Eurospeech*, pages 2767–2770.

Peter A. Heeman. 1998. POS tags versus classes in language modeling. In *Proceedings of Sixth Workshop on Very Large Corpora*.

Arthur E. Hoerl and Robert W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67.

Clifford M. Hurvich and Chih-Ling Tsai. 1989. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, June.

Shuntaro Isogai, Katsuhiko Shirai, Hirofumi Yamamoto, and Yoshinori Sagisaka. 2001. Multi-class composite n-gram language model using multiple word clusters and word successions. In *Proceedings of Eurospeech*, pages 25–28.

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 and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, The Netherlands: North-Holland, May.

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.

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 *Proceedings of EMNLP*, pages 137–144.

Sanjeev Khudanpur. 1995. A method of maximum entropy estimation with relaxed constraints. In *1995 Johns Hopkins University Language Modeling Workshop Proceedings*, pages 1–17, Center for Language and Speech Processing, Johns Hopkins University, Baltimore.

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communications*, 38(1):19–28.

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

Reinhard Kneser. 1996. Statistical language modeling using a variable context length. In *Proceedings of ICSLP*, volume 1, pages 494–497, Philadelphia, October.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of IJCAI*, pages 1137–1145.

John Langford. 2005. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306.

Raymond Lau. 1994. Adaptive statistical language modelling. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Edward E. Leamer. 1978. *Specification Searches: Ad Hoc Inference with Nonexperimental Data*. John Wiley & Sons, Inc., New York.

Guy Lebanon and John Lafferty. 2001. Boosting and maximum likelihood for exponential models. Technical Report CMU-CS-01-144, Carnegie Mellon University.

Jean-Dominique Lebreton, Kenneth P. Burnham, Jean Clobert, and David R. Anderson. 1992. Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. *Ecological Monographs*, 62:67–118.

Nick Littlestone and Manfred K. Warmuth. 1986. Relating data compression and learnability. Technical report, University of California, Santa Cruz.

Colin L. Mallows. 1973. Some comments on C_p. *Technometrics*, 12:591–612.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2).

Sven Martin, Jörg Liermann, and Hermann Ney. 1995. Algorithms for bigram and trigram word clustering. In *Proceedings of Eurospeech*, pages 1253–1256.

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, page 783, Washington, DC, USA. IEEE Computer Society.

David A. McAllester. 1999. PAC-Bayesian model averaging. In *Proceedings of COLT*, pages 164–170, New York, NY, USA. ACM.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8:1–38.

Andrew Y. Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of ICML*, New York, NY, USA. ACM Press.

Thomas R. Niesler, Edward W. D. Whittaker, and Philip C. Woodland. 1998. Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *Proceedings of ICASSP*, pages 177–180.

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

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

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

Jorma Rissanen. 1978. Modeling by the shortest data description. *Automatica*, 14:465–471.

Christer Samuelsson and Wolfgang Reichl. 1999. A class-based language model for large-vocabulary speech recognition extracted from part-of-speech statistics. In *Proceedings of ICASSP*, pages 537–540, Washington, DC, USA. IEEE Computer Society.

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

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686.

Gideon Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

Jun Shao. 1997. An asymptotic theory for linear model selection. *Statistica Sinica*, 7:221–264.

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.

Richard M. Stern. 1996. Specification of the 1995 ARPA Hub 3 evaluation: Unlimited vocabulary NAB news baseline. In *Proceedings of the DARPA Speech Recognition Workshop*, pages 5–7, Harriman, NY, February.

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.

M. Stone. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36:111–147.

M. Stone. 1977. Asymptotics for and against cross-validation. *Biometrika*, 64(1):29–35.

M. Stone. 1979. Comments on model selection criteria of Akaike and Schwarz. *Journal of the Royal Statistical Society B*, 41:276–278.

Kei Takeuchi. 1976. Distribution of information statistics and validity criteria of models. *Mathematical Science*, 153:12–18. In Japanese.

Toshiyuki Takezawa, Tsuyoshi Morimoto, and Yoshinori Sagisaka. 1998. Speech and language databases for speech translation research in ATR. In *Proceedings of Oriential-COCOSDA Workshop*, pages 145–148.

Robert Tibshirani. 1994. Regression shrinkage and selection via the lasso. Technical report, University of Toronto.

Vladimir N. Vapnik and Alexey Y. Chervonenkis. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley, New York.

Dimitra Vergyri, Katrin Kirchhoff, Kevin Duh, and Andreas Stolcke. 2004. Morphology-based language modeling for Arabic speech recognition. In *Proceedings of ICSLP*, pages 2245–2248.

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

Wen Wang and Dimitra Vergyri. 2006. The use of word n-grams and parts of speech for hierarchical cluster language modeling. In *Proceedings of ICASSP*, pages 1057–1060.

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

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

Edward W. D. Whittaker and Philip C. Woodland. 2001. Efficient class-based language modelling for very large vocabularies. In *Proceedings of ICASSP*, volume 1, pages 545–548.

Edward W. D. Whittaker and Philip C. Woodland. 2003. Language modelling for Russian and English using words and classes. *Computer Speech and Language*, 17(1):87–104.

Peter M. Williams. 1995. Bayesian regularization and pruning using a Laplace prior. *Neural Comput.*, 7(1):117–143.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, July.

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

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

Imed Zitouni, Olivier Siohan, and Chin-Hui Lee. 2003. Hierarchical class n-gram language models: towards better estimation of unseen events in speech recognition. In *Proceedings of Eurospeech*, pages 237–240.

Imed Zitouni. 2007. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech and Language*, 21(1):88–104.