Pruning Exponential Language Models

Stanley F. Chen, Abhinav Sethy, Bhuvana Ramabhadran

IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598 USA
{stanchen,asethy,bhuvana}@us.ibm.com

***Abstract*—Language model pruning is an essential technology for speech applications running on resource-constrained devices, and many pruning algorithms have been developed for conventional word n-gram models. However, while exponential language models can give superior performance, there has been little work on the pruning of these models. In this paper, we propose several pruning algorithms for general exponential language models. We show that our best algorithm applied to an exponential n-gram model outperforms existing n-gram model pruning algorithms by up to 0.4% absolute in speech recognition word-error rate on Wall Street Journal and Broadcast News data sets. In addition, we show that Model M, an exponential class-based language model, retains its performance improvement over conventional word n-gram models when pruned to equal size, with gains of up to 2.5% absolute in word-error rate.**

I. Introduction

Due to the availability of increasingly large text corpora and the limited amount of memory in computing devices, language model pruning has become an indispensable technology. While pruning for conventional word n-gram models has received a great deal of attention, there has been little work on pruning for exponential (or *maximum entropy*) language models. However, recent work has shown that exponential language models such as Model M can achieve superior performance [1].

In this paper, we show how many existing n-gram model pruning algorithms can be viewed as attempting to optimize estimated test set perplexity, and discuss how ideas from these techniques can be adapted to exponential models. We present several methods for estimating the change in test set perplexity resulting from the removal of an n-gram, and evaluate the associated pruning algorithms in word-error rate for both exponential n-gram models and Model M. For exponential n-gram models, we find that our methods compare favorably to state-of-the-art pruning methods for conventional n-gram models. In addition, we find that Model M retains its performance gains relative to word n-gram models when pruned to equal size, with gains increasing with heavier pruning.

In the remainder of this section, we provide an introduction to exponential n-gram models and Model M. In Section II, we discuss key pruning algorithms from the literature and show how to adapt these to exponential models. In Section III, we present experiments and follow with related work in Section IV and conclusions in Section V.

An exponential model $p_\Lambda(y|x)$ contains a set of features $\{f_i(x,y)\}$ and equal number of parameters $\Lambda = \{\lambda_i\}$ where

$$p_\Lambda(y|x) = \frac{\exp(\sum_i \lambda_i f_i(x,y))}{Z_\Lambda(x)} \tag{1}$$

and where the $Z_\Lambda(x) = \sum_{y'} \exp(\sum_i \lambda_i f_i(x,y'))$ are normalization terms. In an exponential language model, we take y to be the current word and x to be a number of preceding words. A word n-gram model can be expressed as an exponential model in the following manner: Let $f_\theta(\cdot)$ denote a binary n-gram feature such that $f_\theta(x,y) = 1$ iff xy "ends" in the n-gram θ. Then, an *exponential n-gram model* has a feature $f_\theta(\cdot)$ for each m-gram occurring in the training data for $m \leq n$. Such models describe the same space of conditional models as conventional n-gram models do, and can be thought of as an alternate parameterization of the same model space [2]. One advantage of this representation is that smoothing can be done simply and effectively via $\ell_1 + \ell_2^2$ regularization [3], [1] where the parameters Λ are chosen to optimize

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

where $\mathrm{PP}_{\mathrm{trn}}$ is training set perplexity, D is the number of words in the training set, and α and σ are regularization hyperparameters. Unpruned n-gram models regularized this way have about the same performance as conventional n-gram models smoothed with modified Kneser-Ney smoothing [4].

Model M is a class-based n-gram model composed of two separate exponential models, one for predicting classes and one for predicting words. Let $p_{\mathrm{ng}}(y|\theta)$ denote an exponential n-gram model and let $p_{\mathrm{ng}}(y|\theta_1, \theta_2)$ denote a model containing all features in $p_{\mathrm{ng}}(y|\theta_1)$ and $p_{\mathrm{ng}}(y|\theta_2)$. If we assume that every word w is mapped to a single word class, the trigram version of Model M is defined as

$$\begin{aligned} p_M(w_j|w_{j-2}w_{j-1}) \equiv\, & p_{\mathrm{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \times \\ & p_{\mathrm{ng}}(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \tag{3}$$

where c_j is the word class of word w_j. Model M has achieved among the largest word-error rate improvements over word n-gram models ever reported, with gains as high as 3% absolute as compared to a Katz-smoothed trigram model [5].

II. Pruning for exponential language models

In this section, we present a general framework for analyzing pruning algorithms. A natural goal in designing a pruning algorithm is to maximize test set performance given a fixed language model size. For computational reasons, we attempt to optimize the *perplexity* of test data rather than an application-specific measure such as word-error rate. These two quantities are well-correlated when considering models of a single type, *e.g.*, [4]. However, since test data is by its nature "unseen", we cannot directly measure test set perplexity and must instead estimate it. Note that if we have the counts of the n-grams in a test set, it is straightforward to compute its perplexity. Furthermore, note that the probabilities given by a smoothed language model can be interpreted as estimates of n-gram frequencies in unseen data, and thus we can use the corresponding counts to compute test set perplexities.

While generally not framed in this way, many conventional n-gram pruning algorithms can also be viewed as attempting to optimize test set perplexity. In particular, several algorithms attempt to minimize the Kullback-Leibler distance between the original model and the pruned model [6], [7]:

$$D(p_{\text{orig}} \parallel p_{\text{prune}}) = \sum_{x,y} p(x) p_{\text{orig}}(y|x) \log \frac{p_{\text{orig}}(y|x)}{p_{\text{prune}}(y|x)} \quad (4)$$

Test set log perplexity is equivalent to the cross-entropy between the empirical test set distribution p_{tst} and the final model p_{prune}, which differs from $D(p_{\text{tst}} \parallel p_{\text{prune}})$ by a constant. As mentioned earlier, p_{orig} can be viewed as an estimate of p_{tst}, so minimizing $D(p_{\text{orig}} \parallel p_{\text{prune}})$ is essentially equivalent to minimizing test set perplexity.

In the following sections, we discuss how various existing conventional n-gram pruning algorithms correspond to successively better approximations to eq. (4), and use the isomorphism between conventional and exponential n-gram models to show how these ideas can be applied to exponential language models as well.

A. Weighted difference pruning

One of the earliest and simplest algorithms for pruning conventional n-gram models is *weighted difference* pruning [8]. As noted in [9], smoothed conventional n-gram models can generally be expressed as

$$p_s(w_j|w_{j-n+1}^{j-1}) = \begin{cases} q(w_j|w_{j-n+1}^{j-1}) & \text{if } c_{\text{trn}}(w_{j-n+1}^{j}) > 0 \\ \alpha(w_{j-n+1}^{j-1}) \times & \\ \quad p_s(w_j|w_{j-n+2}^{j-1}) & \text{otherwise} \end{cases} \quad (5)$$

where $w_j^k \equiv w_j w_{j+1} \cdots w_k$ and where lower-order distributions of $p_s(\cdot)$ are defined analogously. The values $q(\cdot|\cdot)$ can be viewed as the parameters of the model, similar to the $\lambda_i' s$ in an exponential model. The *back-off factors* $\alpha(\cdot)$ force each distribution to sum to 1, and play the same role as the normalization constants $Z_\Lambda(x)$ in exponential models.

Weighted difference pruning makes the approximation that when pruning an n-gram θ, the $q(\cdot)$ parameter associated with θ is deleted and all other $q(\cdot)$ and $\alpha(\cdot)$ are left unchanged. For simplicity, we describe this algorithm for the case of bigram models. Substituting eq. (5) into eq. (4) and simplifying, the impact of removing a single n-gram $w_{j-1}w_j$ is

$$D(p_{\text{orig}} \parallel p_{\text{prune}}) \approx p(w_{j-1}) p_{\text{orig}}(w_j|w_{j-1}) \log \frac{p_s(w_j|w_{j-1})}{\alpha(w_{j-1}) p_s(w_j)}$$

Approximating $p(w_{j-1}) p_{\text{orig}}(w_j|w_{j-1})$ scaled by D with the Good-Turing discounted count $c_{\text{disc}}(w_{j-1}w_j)$, we get

$$\text{score}_{\text{WD}}(w_{j-1}w_j) = c_{\text{disc}}(w_{j-1}w_j) \log \frac{p_s(w_j|w_{j-1})}{\alpha(w_{j-1}) p_s(w_j)} \quad (6)$$

This score is computed for each n-gram relative to the full model, and n-grams with scores below a threshold are pruned.

To see how we can construct an analogous pruning algorithm for exponential language models, let us map the conventional n-gram parameters to exponential model parameter space and find the analog to eq. (6). Using the equivalence relations from [2], we have

$$p_s(w_j|w_{j-1}) = \frac{\exp(\lambda_{w_j} + \lambda_{w_{j-1}w_j})}{Z_\Lambda(w_{j-1})} \quad (7)$$

$$\alpha(w_{j-1}) p_s(w_j) = \frac{\exp(\lambda_{w_j})}{Z_\Lambda(w_{j-1})} \quad (8)$$

Substituting into eq. (6), we get

$$\text{score}_{\text{simple}}(w_{j-1}w_j) = c_{\text{disc}}(w_{j-1}w_j) \lambda_{w_{j-1}w_j} \quad (9)$$

By generalizing this equation to apply to arbitrary features rather than just n-grams, this gives us a pruning algorithm (*simple*) that can be applied to arbitrary exponential language models. Thus, we have shown how an existing algorithm for conventional n-gram models can be adapted to exponential language models, and we repeat this exercise for more complex algorithms below.

B. Relative entropy pruning

Relative entropy pruning [7] is the most popular pruning algorithm in practice, and improves upon weighted difference pruning by also accounting for the effect on the back-off factor $\alpha(w_{j-1})$ when pruning the bigram $w_{j-1}w_j$. Specifically, $\alpha(w_{j-1})$ is adjusted so that the model is again normalized. Then, $D(p_{\text{orig}} \parallel p_{\text{prune}})$ can be computed exactly and efficiently when pruning a single n-gram from the original model.

A natural way to extend these ideas to exponential models is to account for the change in the normalization constants $Z_\Lambda(x)$ when pruning a feature. Our first method is motivated by the empirical relationship from [1] relating training and test set performance for exponential language models:

$$\log \text{PP}_{\text{tst}} \approx \log \text{PP}_{\text{trn}} + \frac{\gamma}{D} \sum_i |\lambda_i| \quad (10)$$

where $\gamma = 0.938$ works well when regularizing with ($\alpha = 0.5, \sigma^2 = 6$). Note that eq. (2) is a generalization of eq. (10), so the former equation can also be used as a proxy for estimated test set performance.

Let us compute the change in $\log \text{PP}_{\text{trn}}$ resulting from removing a single n-gram θ. Plugging eq. (1) into the definition of perplexity, for binary features we have that

$$\log \text{PP}_{\text{trn}} = \frac{1}{D} \left(-\sum_\theta c_{\text{trn}}(\theta) \lambda_\theta + \sum_x c_{\text{trn}}(x) \log Z_\Lambda(x) \right)$$

$$D \log \frac{\text{PP}_{\text{trn}}^{\text{new}}}{\text{PP}_{\text{trn}}^{\text{old}}} = c_{\text{trn}}(\theta) \lambda_\theta + \sum_{x: f_\theta \text{ active}} c_{\text{trn}}(x) \log \frac{Z_\Lambda^{\text{new}}(x)}{Z_\Lambda^{\text{old}}(x)}$$

Let $s(x,y) \equiv \exp(\sum_i \lambda_i f_i(x,y))$ and let y_θ be the last word in θ. Then, the last term in the previous equation can be approximated as

$$\sum_{x:f_\theta \text{ active}} c_{\text{trn}}(x) \frac{s^{\text{new}}(x, y_\theta) - s^{\text{old}}(x, y_\theta)}{Z_\Lambda^{\text{old}}(x)} \tag{11}$$

$$= D \cdot E_{p_\Lambda}[f_\theta] \cdot (e^{-\lambda_\theta} - 1) \tag{12}$$

When using $\ell_1 + \ell_2^2$ regularization, $E_{p_\Lambda}[f_\theta]$ can be computed simply using the relationship

$$E_{p_\Lambda}[f_\theta] = \frac{1}{D}\left(c_{\text{trn}}(\theta) - \alpha\,\text{sgn}(\lambda_\theta) - \frac{\lambda_\theta}{\sigma^2}\right) \tag{13}$$

Putting everything together, we get

$$\text{score}_{\text{train}}(\theta) = c_{\text{trn}}(\theta)\lambda_\theta + D \cdot E_{p_\Lambda}[f_\theta] \cdot (e^{-\lambda_\theta} - 1) - \alpha|\lambda_\theta| - \frac{\lambda_\theta^2}{2\sigma^2} \tag{14}$$

Our second method for approximating the effect of renormalization, *norm* pruning, is inspired by ideas from [10]. We make the assumption that $p_\Lambda(y_\theta|x)$ is constant across all x for which a feature θ is active. Then, we have

$$E_{p_\Lambda}[f_\theta] = \frac{1}{D} p_\Lambda(y_\theta|x) \sum_{x:f_\theta \text{ active}} c_{\text{trn}}(x) \equiv \frac{1}{D} p_\Lambda(y_\theta|x) c_{\text{hist}}(f_\theta)$$

where $c_{\text{hist}}(f_\theta)$ denotes the number of training event histories where f_θ is active. Using eq. (13), we get

$$p_{\text{avg}}(f_\theta) \equiv p_\Lambda(y_\theta|x) = \frac{c_{\text{trn}}(\theta) - \alpha\,\text{sgn}(\lambda_\theta) - \frac{\lambda_\theta}{\sigma^2}}{c_{\text{hist}}(f_\theta)} \tag{15}$$

for x where f_θ is active.

Then, using eq. (1), we can compute how $p_\Lambda(y_\theta|x)$ changes for these events if the feature f_θ is pruned:

$$p_{\text{prune}}(f_\theta) = \frac{p_{\text{avg}}(f_\theta) e^{-\lambda_\theta}}{1 + p_{\text{avg}}(f_\theta)(e^{-\lambda_\theta} - 1)} \tag{16}$$

The denominator reflects the change in the normalization constant. Thus, for each event (x, y_θ) where f_θ is active, the loss in log likelihood for the pruned model is $-\log \frac{p_{\text{prune}}(f_\theta)}{p_{\text{avg}}(f_\theta)}$. For each event (x, y) where f_θ is active and $y \neq y_\theta$, there is a gain in log likelihood of

$$\log(1 + p_{\text{avg}}(f_\theta)(e^{-\lambda_\theta} - 1)) = \log \frac{1 - p_{\text{prune}}(f_\theta)}{1 - p_{\text{avg}}(f_\theta)} \tag{17}$$

due to the adjusted normalization constant. Then, we need only estimate the counts of these two types of events. The number of events of the first type is $c_{\text{tst}}(\theta)$, and a natural estimate is the marginal count $D \cdot E_{p_\Lambda}[f_\theta]$ of that feature (which can be computed efficiently using eq. (13)). For events of the second type, we use the estimate $c_{\text{hist}}(f_\theta) - c_{\text{tst}}(\theta)$. Putting it all together and scaling by D, we get

$$\text{score}_{\text{norm}}(\theta) = c_{\text{hist}}(f_\theta)[p_{\text{avg}}(f_\theta) \log \frac{p_{\text{avg}}(f_\theta)}{p_{\text{prune}}(f_\theta)} + (1 - p_{\text{avg}}(f_\theta)) \log \frac{1 - p_{\text{avg}}(f_\theta)}{1 - p_{\text{prune}}(f_\theta)}] \tag{18}$$

This is essentially identical to the gain of adding a binary feature to a joint exponential model reported in [10].

TABLE I
EXPONENTIAL LANGUAGE MODEL PRUNING ALGORITHMS.

name	score(θ)
count cutoffs	$c_{\text{trn}}(\theta)$
λ cutoffs	$\lvert\lambda_\theta\rvert$
simple	eq. (9) with $c_{\text{disc}}(\theta) = D \cdot E_{p_\Lambda}[f_\theta]$.
train	eq. (14) with $\alpha = 0.5$, $\sigma^2 = 6$.
perfpred	eq. (14) with $\alpha = 0.938$, $\sigma^2 = \infty$.
norm	eq. (18)
iter.grow.rand	Iterative random growing and *norm* pruning.

C. Revised Kneser pruning

Revised Kneser pruning [11] is specific to n-gram models smoothed with Kneser-Ney smoothing and variants [9], and improves upon relative entropy pruning in several ways. One of the characteristics of Kneser-Ney smoothing is that marginal constraints of the following form are (approximately) satisfied:

$$\sum_{w_{j-1}} p_{\text{trn}}(w_{j-1}) p_s(w_j|w_{j-1}) \approx p_{\text{trn}}(w_j) \tag{19}$$

To maintain this constraint when pruning a bigram $w_{j-1}w_j$, its "count" should be given to the unigram w_j. That is, $q(w_j)$ should generally be boosted when we prune $w_{j-1}w_j$, and it is shown that eq. (4) can still be computed efficiently when reestimating back-off n-gram probabilities so as to maintain marginal constraints. For exponential models, the analogous parameter retraining (*e.g.*, via iterative scaling) is expensive in the general case, and cannot be repeated for each feature. Instead, we retrain parameters only once after pruning is complete; this can be interpreted as resmoothing the model using eq. (2) given the final feature set.

Another improvement of revised Kneser pruning is the use of sequential pruning. Rather than computing the score of all n-grams relative to the full model, after each n-gram is pruned the model is updated, and the adjusted model is used to compute the scores of succeeding n-grams. While it is impractical to update an exponential model after each feature is pruned, we can do so after each *batch* of features; we refer to this as *iterative* pruning.

Finally, [11] proposes a method for growing n-gram models that can be applied before pruning. Ideally, one would like to add a feature to a model based on its estimated gain, but this gain is expensive to compute for general exponential models. Instead, we take a randomized approach: for any existing feature in the model, we consider any extension or truncation by one token that occurs in the training set, and add that feature with probability p_{keep}. In the algorithm *iter.norm.rand*, we alternate randomized growing and *norm* pruning stages (and parameter retraining).

III. Experiments

We present results on two different corpora: Wall Street Journal and Broadcast News. For the Wall Street Journal runs, we use the same data sets and methodology as in [1]. We use a training set of 23M words and a vocabulary of 21k words. The development and evaluation sets are 18k and 47k words, respectively. The acoustic model is a cross-word quinphone system built from 50h of Broadcast News data and contains

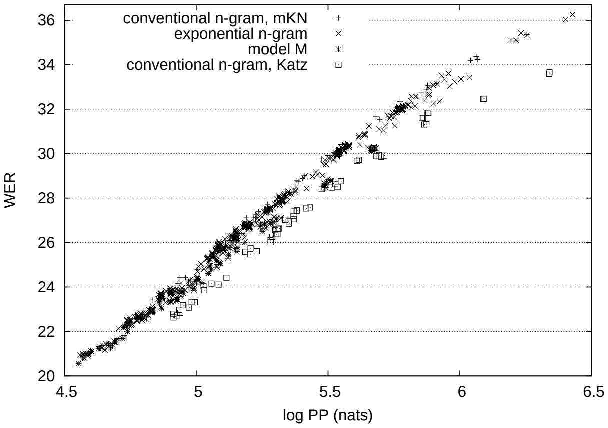

Fig. 1. Comparison between test set perplexity and word-error rate for pruned models on Wall Street Journal data.

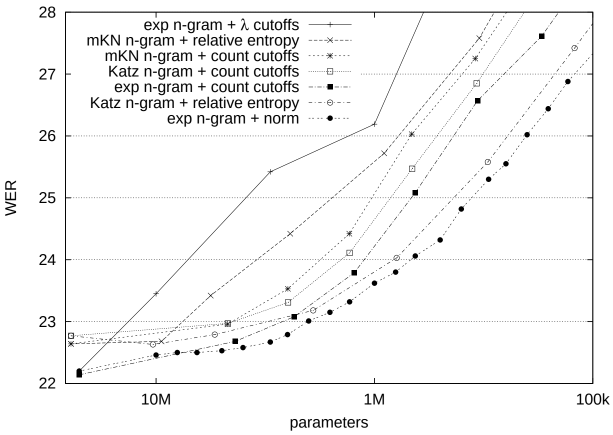

Fig. 2. Test set word-error rates for various pruning algorithms for conventional and exponential 4-gram models, 23MW WSJ training set.

2176 context-dependent states and 50k Gaussians. We use lattice rescoring to evaluate each language model, and choose the acoustic weight for each model to optimize the word-error rate of that model on the development set.

The speech recognition setup for the Broadcast News experiments is based on the 2007 IBM GALE speech transcription system. The acoustic model was discriminatively trained on 430h of Broadcast News audio and contains 6000 context-dependent states and 250k Gaussians. The language model training text is 130MW of 1996 CSR Hub4 language model data, and a vocabulary of 84k words is used. The evaluation set is the 2.5h RT04 evaluation set containing 45k words.

In all experiments, the 4-gram versions of models are used. For count cutoffs, we use the same cutoff values across all n-gram levels. Unless otherwise noted, all of the evaluated pruning algorithms have the following form: we assign a score to each feature, and prune all features whose score falls below a given threshold. All exponential models are trained with $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$ as recommended in [1], and models are retrained in the same way after pruning. The exponential language model pruning algorithms we evaluate are summarized in Table I.

We take the size of a model to be the number of nonzero parameters. However, we do not count normalization parameters $Z_\Lambda(\cdot)$ nor the analogous back-off parameters $\alpha(\cdot)$ since normalization constants may be computed on the fly for exponential models. We do not count word unigram features (nor try to prune them) since these are mandatory in ARPA n-gram model format. For Model M, we *do* count one parameter for each word specifying the class membership of that word.

We first evaluate our algorithms on the Wall Street Journal data before presenting results with the larger Broadcast News training set. While most previous work has focused on perplexity, we will present word-error rate results almost exclusively. In Figure 1, we plot log perplexity versus word-error rate for many of the pruned Wall Street Journal language models that we evaluated. Some types of models systematically achieve better word-error rates for the same perplexity than others. Most notably, Katz-smoothed n-gram models appear to outperform modified-Kneser-Ney-smoothed n-gram models by about 1% absolute at the same perplexity value. Thus, perplexity results can be misleading when comparing pruning algorithms. Also, we report performance differences as *absolute* word-error rates rather than relative differences, as these tend to vary less across different baseline word-error rates [1].

In Figure 2, we compare the performance of existing pruning algorithms for conventional and exponential n-gram models against *norm* pruning, our primary baseline among the algorithms we propose. (In most graphs, the key will list algorithms in the same order as their performance.) For each algorithm, we evaluate a variety of pruning thresholds to produce a curve of word-error rate versus model size, with the points to the far left corresponding to unpruned models.

Among the pruning algorithms for conventional n-gram models, we find that both relative entropy pruning and count cutoffs perform badly with Kneser-Ney-smoothed models, as consistent with previous findings in the literature [11]. In contrast, Katz smoothing, despite its relatively poor unpruned behavior, does better, with relative entropy pruning outperforming count cutoffs. Indeed, relative entropy pruning with Katz-smoothed models represents the start of the art for conventional n-gram pruning in terms of word-error rate, except when using little or no pruning. (We do not evaluate revised Kneser pruning here, as no word-error rate gains have been found as compared to the preceding algorithm [11].) However, exponential n-gram models with *norm* pruning do even better across a wide range of pruning thresholds. Count cutoffs are somewhat worse while λ cutoffs perform very poorly with exponential language models.

For algorithms that are close in word-error rate, performance differences are clearer when plotting performance relative to some baseline algorithm. We do this for a number of algorithms relative to *norm* pruning in Figure 3, where the line $y = 0$ corresponds to the performance of *norm* pruning. Linear interpolation is used to estimate error rates for model sizes that we do not have direct measurements for. In this graph, we can see that *norm* pruning outperforms relative entropy pruning with Katz smoothing by 0.1–0.4% absolute with modest to heavy pruning. Among the other pruning

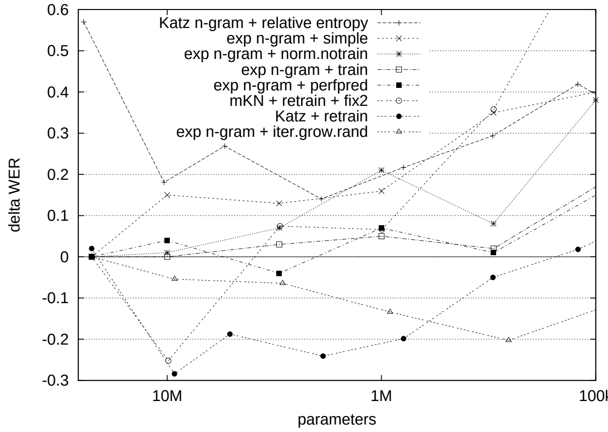

Fig. 3. Relative test set performance between various pruning algorithms and *norm* pruning, 23MW Wall Street Journal training set.

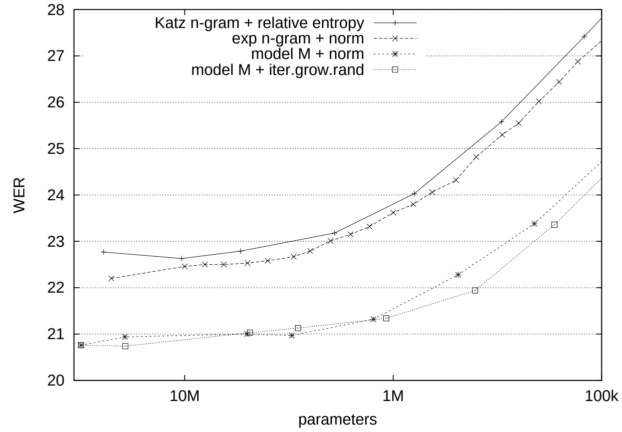

Fig. 5. Test set word-error rates for various pruning algorithms for Model M and word 4-gram model baselines, 23MW Wall Street Journal training set.

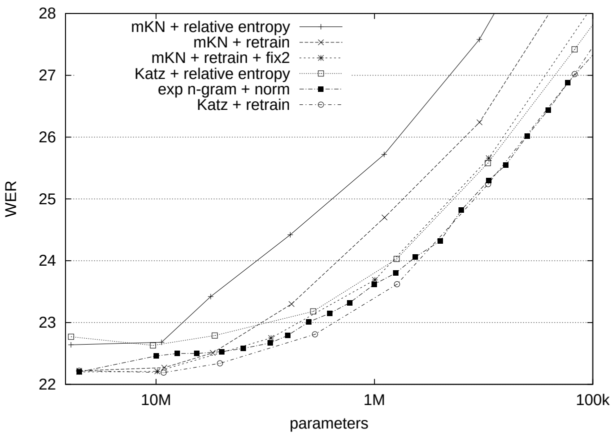

Fig. 4. Test set word-error rates for retrained relative-entropy-pruned conventional 4-gram models and baseline algorithms, 23MW WSJ training.

algorithms for exponential models, *simple* pruning is the worst of these as befits its name. The algorithms *train* and *perfpred* are very close to *norm*, within 0.1% absolute for most pruning thresholds, though they may be slightly worse with very high pruning. The algorithm *norm.notrain* is like *norm* pruning except we do not retrain parameters after pruning. We see that retraining can make a significant difference with heavier pruning. The algorithm *iter.grow.rand* (with 20 iterations and $p_{\text{keep}} = 0.1$) achieves modest performance gains of up to 0.2% absolute over *norm* pruning.

Note that the performance of a pruned model depends on two separate factors: which features are pruned, and how the resulting model is smoothed. Particularly, exponential models smoothed with $\ell_1 + \ell_2^2$ regularization tend to outperform smoothed conventional n-gram models, so performance gains from pruned exponential models may be due to better smoothing rather than better feature selection. To separate the impact of these two factors, we convert pruned conventional n-gram models into the equivalent exponential n-gram models and then retrain their parameters using $\ell_1 + \ell_2^2$ regularization. In this way, we can directly compare the quality of feature selection since the smoothing is identical. In Figures 3 and 4, we display the performance of relative entropy pruning with conventional n-gram models when followed by parameter retraining. Parameter retraining corrects one of the two flaws identified in [11] associated with applying relative entropy pruning to Kneser-Ney-smoothed models. The other flaw can be corrected by taking $p(x)$ in eq. (4) to be equal to its training set frequency; this is denoted as *fix2* in the graph labels. We see that both retraining and applying *fix2* improve relative entropy pruning with Kneser-Ney smoothing a great deal. However, parameter retraining with Katz smoothing yields even better performance, with parameter retraining improving performance by about 0.4–0.5% absolute and overall performance surpassing *norm* pruning by up to 0.3% absolute. These results suggest that the main contributor to better performance for pruned exponential n-gram models is superior smoothing rather than better feature selection.

As discussed in Section I, the class prediction model in Model M contains features from two different exponential n-gram models, and thus conventional n-gram model pruning algorithms cannot be applied. For the Wall Street Journal experiments, we use the enhanced word classing algorithm for Model M developed in [5]. For the Broadcast News experiments, we use bigram mutual information word clustering [12] to build word classes.

In Figure 5, we compare algorithms for pruning Model M against our main baselines for word n-gram model pruning. Despite its larger unpruned size, Model M consistently yields sizable gains versus word n-gram models, with gains of about 1.5% absolute with light pruning and gains of above 2.5% absolute at the right edge of the graph, with *iter.grow.rand* pruning doing slightly better than *norm* pruning.

Finally, in Figure 6, we display the performance of various pruning algorithms for word n-gram models and Model M using the 130MW Broadcast News training set. We see qualitatively similar results as before with this larger data set. Again, with modest to heavy pruning, Katz smoothing outperforms modified Kneser-Ney smoothing when using relative entropy pruning. The *norm* algorithm outperforms Katz smoothing with relative entropy pruning by about 0.3% absolute or more across a wide range of pruning thresholds, with Katz smooth-

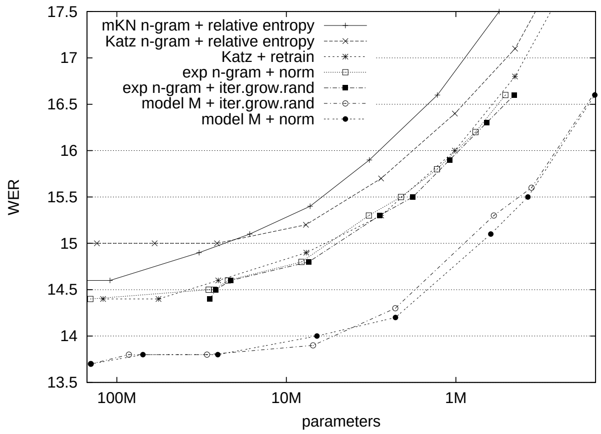

Fig. 6. Test set word-error rates for various pruning algorithms for conventional and exponential 4-gram models and 4-gram Model M, 130MW Broadcast News training set.

ing with retraining and *iter.norm.rand* pruning performing about the same. While gains for Model M are not as large as with the smaller training set, we still see consistent gains of 1% absolute as compared to Katz smoothing with relative entropy pruning, with gains of up to 1.5% absolute with heavier pruning.

IV. Related Work

Here, we discuss previous work on pruning for exponential language models. The use of *count cutoffs* is a common technique, *e.g.*, [13], [14]. Count cutoffs are compared with smoothing techniques for exponential n-gram models in [2].

The use of ℓ_1 or $\ell_1 + \ell_2^2$ regularization [3], [15] has been noted to produce *sparse* solutions; *i.e.*, many λ_i are set to 0 and can be trivially pruned. For example, 8% of the λ_i's in our unpruned Wall Street Journal exponential 4-gram model are zero due to regularization. While such techniques can be effective for light pruning, α needs to be increased for heavier pruning, likely leading to poor test set performance.

A related technique to model pruning is model *growing*, or *feature induction*. In [16], several criteria for inducing features in a maximum entropy language model are compared, including the feature gain computation described in [10], count cutoffs, and a criterion based on mutual information [14]. They found modest gains in perplexity with the first method as compared to the latter, but speech recognition experiments revealed basically no gain in word-error rate as compared to a baseline trigram model [17].

V. Discussion

There has been relatively little work on the pruning of general exponential language models, and this paper provides the first systematic comparison of many methods for pruning such models. We propose several novel pruning algorithms and show that these outperform simpler methods such as count cutoffs and λ cutoffs by a large margin. The method *norm* is efficient, easy to implement, and achieves excellent word-error rates. The method *iter.grow.rand*, while complex, can sometimes give slightly better performance.

On the other hand, there has been a great deal of work on pruning conventional n-gram models. The majority of results in the pruning literature have dealt with perplexity, but we show that this is a poor predictor of speech recognition performance due to systematic differences in word-error rates for different methods. While relative entropy pruning with Katz-smoothed models have been surpassed in perplexity [11], this algorithm has not been bettered in the literature in word-error rate, and thus can be viewed as the state of the art for conventional n-gram pruning. Here, we show that we can improve upon this baseline by resmoothing with $\ell_1 + \ell_2^2$ regularization or by using *norm* pruning with exponential n-gram models, with word-error rate gains of up to 0.4–0.5% absolute. We show that these gains are due primarily to better smoothing rather than better feature selection.

Finally, we apply our novel pruning algorithms to Model M and show that it retains its word-error rate gain over word n-gram models when pruned to the same number of parameters. We find that compared to a (pruned) Katz-smoothed 4-gram model, gains can be up to 2.5% absolute with heavy pruning, larger than previous gains reported in the literature for equal-size class-based models, *e.g.*, [18].

References

[1] S. F. Chen, "Performance prediction for exponential language models," IBM Research Division, Tech. Rep. RC 24671, October 2008.
[2] S. F. Chen and R. Rosenfeld, "A survey of smoothing techniques for maximum entropy models," *IEEE Trans. on Speech and Audio Processing*, vol. 8, no. 1, pp. 37–50, 2000.
[3] J. Kazama and J. Tsujii, "Evaluation and extension of maximum entropy models with inequality constraints," in *Proc. EMNLP*, 2003.
[4] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Harvard U., Tech. Rep. TR-10-98, 1998.
[5] S. F. Chen and S. M. Chu, "Enhanced word classing for model M," in *Proc. Interspeech*, 2010.
[6] R. Kneser, "Statistical language modeling using a variable context length," in *Proc. ICSLP*, vol. 1, October 1996, pp. 494–497.
[7] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, February 1998, pp. 270–274.
[8] K. Seymore and R. Rosenfeld, "Scalable backoff language models," in *Proc. ICSLP*, vol. 1, October 1996, pp. 232–235.
[9] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proc. ICASSP*, vol. I, May 1995, pp. 181–184.
[10] S. Della Pietra, V. Della Pietra, and J. Lafferty, "Inducing features of random fields," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, April 1997.
[11] V. Siivola, T. Hirsimki, and S. Virpioja, "On growing and pruning Kneser-Ney smoothed n-gram models," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 5, pp. 1617–1624, July 2007.
[12] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, December 1992.
[13] R. Lau, "Adaptive statistical language modelling," Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1994.
[14] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Comp. Speech and Lang.*, vol. 10, 1996.
[15] J. Goodman, "Exponential priors for maximum entropy models," in *Proc. NAACL*, 2004.
[16] A. Berger and H. Printz, "A comparison of criteria for maximum entropy/minimum divergence feature selection," in *Proc. EMNLP*, 1998.
[17] ——, "Recognition performance of a large-scale dependency grammar language model," in *Proc. ICSLP*, 1998.
[18] H. Yamamoto, S. Isogai, and Y. Sagisaka, "Multi-class composite n-gram language model," *Speech Comm.*, vol. 41, no. 2-3, 2003.