A Survey of Smoothing Techniques for Maximum Entropy Models

Stanley F. Chen and Ronald Rosenfeld

Abstract—
In certain contexts, maximum entropy modeling can be viewed as maximum likelihood training for exponential models, and like other maximum likelihood methods is prone to overfitting of training data. Several smoothing methods for maximum entropy models have been proposed to address this problem, but previous results do not make it clear how these smoothing methods compare with smoothing methods for other types of related models. In this work, we survey previous work in maximum entropy smoothing and compare the performance of several of these algorithms with conventional techniques for smoothing n-gram language models. Because of the mature body of research in n-gram model smoothing and the close connection between maximum entropy and conventional n-gram models, this domain is well-suited to gauge the performance of maximum entropy smoothing methods. Over a large number of data sets, we find that *fuzzy maximum entropy* smoothing [1] performs as well as or better than all other algorithms under consideration. We contrast this method with previous n-gram smoothing methods to explain its superior performance.

I. Introduction

Maximum entropy (ME) modeling has been successfully applied to a wide range of domains, including language modeling as well as many other natural language tasks [2], [3], [4], [5]. For many problems, this type of modeling can be viewed as maximum likelihood (ML) training for exponential models, and like other maximum likelihood methods is prone to overfitting of training data. While several smoothing methods for maximum entropy models have been proposed to address this problem [1], [6], [7], [5], previous results do not make it clear how these smoothing methods compare with smoothing methods for other types of related models.

However, there has a been great deal of research in smoothing n-gram language models, and it can be shown that maximum entropy n-gram models are closely related to conventional n-gram models. Consequently, this domain is well-suited to gauging the performance of maximum entropy smoothing methods relative to other smoothing techniques.

In this work, we survey previous work in maximum entropy smoothing and compare the performance of several of these algorithms with conventional techniques for smoothing n-gram language models. Evaluating the perplexity of each method over a large number of data sets, we find that *fuzzy maximum entropy* smoothing [1] performs as well as or better than all other algorithms under consideration. This method can be viewed as relaxing the requirement of exact constraint satisfaction in the ME framework by using a quadratic penalty for inexact constraint satisfaction. Equivalently, this method can also be viewed as applying a Gaussian prior on model parameters and selecting maximum *a posteriori* instead of maximum likelihood parameter values. While simple and efficient, this method exhibits all of the behaviors that have been observed by Chen and Goodman to be beneficial for n-gram smoothing [8].

In the remainder of this section, we present an introduction to maximum entropy modeling and discuss why smoothing ME models is necessary. In Section II, we introduce n-gram language models and summarize previous work on smoothing these models. We list the desirable properties of smoothing algorithms observed by Chen and Goodman. In Section III, we introduce maximum entropy n-gram models and discuss their relationship with conventional n-gram models. In Section IV, we survey previous work in smoothing maximum entropy models including the fuzzy maximum entropy technique. In Section V, we contrast fuzzy maximum entropy with smoothing algorithms for conventional n-gram models and show that it satisfies all of the criteria of Chen and Goodman. In Section VI, we present results of experiments comparing a number of maximum entropy and conventional smoothing techniques on n-gram language modeling, evaluating models through both perplexity and speech recognition word error rate. Finally, in Section VII we discuss our conclusions.

A. Maximum Entropy Modeling

Consider the task of estimating a probability distribution $q(x)$ over a finite set Ω given some training data set $X = \{x_1, \ldots, x_N\}$. Intuitively, our task is to find a distribution $q(x)$ similar to the empirical distribution $\tilde{p}(x)$ given by the training data

$$\tilde{p}(x) = \frac{c_X(x)}{N}$$

where $c_X(x)$ denotes the number of times x occurs in X and where N is the size of the training set. In the extreme case, we can take $q(x)$ to be identical to $\tilde{p}(x)$, but this will typically lead to overfitting to the training data. Instead, it would be better to require that $q(x)$ match only those properties of $\tilde{p}(x)$ that we deem to be significant and that can be reliably estimated from the training data.

For example, consider $x = (w_1, w_2)$ where w_1 and w_2 are English words, and let the training data X be the list of consecutive word pairs, or *bigrams*, that occur in some large corpus of English text. Thus, the task is estimating the frequency of English bigrams. Consider a bigram that does not occur in the training data, say PIG DOG. We have $\tilde{p}(\text{PIG DOG}) = 0$, but intuitively we want $q(\text{PIG DOG}) > 0$ since this bigram has *some* chance of occurring. This is an example of a property of $\tilde{p}(x)$ that we do not deem significant and thus do not want to match exactly with

$q(x)$. However, let us assume that we observe that the word THE occurs with frequency 0.05 in the training data, *i.e.*,

$$\sum_{w_2} \tilde{p}(\text{THE } w_2) = \sum_{w_1} \tilde{p}(w_1 \text{ THE}) = 0.05 \quad .$$

Because of the abundance of the word THE, this is presumably an accurate estimate of this frequency and it seems reasonable to require that our selected distribution $q(x)$ satisfies the analogous constraints

$$\sum_{w_2} q(\text{THE } w_2) = \sum_{w_1} q(w_1 \text{ THE}) = 0.05 \quad . \tag{1}$$

More generally, we can select a number of nonnegative random variables or *features* $\vec{f} = \{f_1(x), \ldots, f_F(x)\}$ and require that the expected value of each feature over the model $q(x)$ is equal to that of the empirical distribution $\tilde{p}(x)$:

$$\sum_x q(x) f_i(x) = \sum_x \tilde{p}(x) f_i(x), \quad i = 1, \ldots, F \quad . \tag{2}$$

The constraints represented in (1) can be expressed with two such features,

$$f_j(w_1, w_2) = \begin{cases} 1 & \text{if } w_j = \text{THE} \\ 0 & \text{otherwise} \end{cases}$$

for $j = 1, 2$.

The constraints given by (2) do not generally specify a unique model $q(x)$, but a set of models $Q_{\vec{f}}$. The *maximum entropy* principle states that we should select the model $q(x) \in Q_{\vec{f}}$ with the largest entropy $H(q) = -\sum_x q(x) \log q(x)$ [9]. Intuitively, models with high entropy are more uniform and correspond to assuming less about the world. The maximum entropy model can be interpreted as the model that assumes only the knowledge that is represented by the features derived from the training data, and nothing else.

The maximum entropy paradigm has many elegant properties [2], [3]. The maximum entropy model is unique and can be shown to be an exponential model of the form

$$q_{\text{ME}}(x) = \frac{1}{Z_\Lambda} \exp\Big(\sum_{i=1}^{F} \lambda_i f_i(x)\Big) \tag{3}$$

where $Z_\Lambda = \sum_x \exp(\sum_{i=1}^{F} \lambda_i f_i(x))$ is a normalization factor and $\Lambda = (\lambda_1, \ldots, \lambda_F)$ are the parameters of the model. Furthermore, the maximum entropy model is also the maximum likelihood model in the class of exponential models given by (3).[1] Finally, the log-likelihood of the training data is concave in the model parameters Λ, and thus it is relatively easy to find the unique maximum entropy/maximum likelihood model using algorithms such as generalized iterative scaling [10] or improved iterative scaling [3].

While models with high entropy tend to be rather uniform or smooth and we may only constrain properties of $q(x)$ we consider significant, a maximum entropy model can still overfit training data, even when the number of constraints is small. For example, consider constraints on the frequency of the word MATEO and the bigram SAN MATEO, and assume that the word MATEO occurs only after the word SAN in the training data. Then, we will have

$$\sum_{w_1} q(w_1 \text{ MATEO}) = \sum_{w_1} \tilde{p}(w_1 \text{ MATEO}) = \tilde{p}(\text{SAN MATEO})$$

and

$$q(\text{SAN MATEO}) = \tilde{p}(\text{SAN MATEO})$$

which implies $q(w_1 \text{ MATEO}) = 0$ for all $w_1 \neq \text{SAN}$. Intuitively, we want $q(x) > 0$ for all $x \in \Omega$ since all bigrams have some chance of occurring. Zero probabilities lead to infinite loss in log-loss objective functions and can lead to poor performance in many applications, *e.g.*, when $q(x)$ represents a language model to be used in speech recognition. Thus, it is desirable to *smooth* maximum entropy models, or adjust parameter values away from their maximum likelihood estimates.

II. Smoothing N-Gram Language Models

While there has been relatively little work in smoothing maximum entropy models, there has been a great deal of work in smoothing n-gram language models. A *language model* is a probability distribution $q(s)$ over word sequences s that models how often each sequence s occurs as a sentence. Language models have many applications, including speech recognition, machine translation, and spelling correction [11], [12], [13].

For a word sequence $s = w_1 \cdots w_l$, we can express its probability $\Pr(s)$ as

$$\begin{aligned} \Pr(s) &= \Pr(w_1) \times \Pr(w_2|w_1) \times \cdots \\ &\quad \times \Pr(w_l|w_1 \cdots w_{l-1}) \times \Pr(\text{END}|w_1 \cdots w_l) \\ &= \prod_{i=1}^{l+1} \Pr(w_i|w_1 \cdots w_{i-1}) \end{aligned}$$

where the token $w_{l+1} = \text{END}$ signals the end of the sentence. The most widely-used language models, by far, are n-gram language models. In an n-gram model, we make the approximation that the identity of a word depends only on past words through the identity of the last $n-1$ words, giving us

$$\Pr(s) = \prod_{i=1}^{l+1} \Pr(w_i|w_1 \cdots w_{i-1}) \approx \prod_{i=1}^{l+1} \Pr(w_i|w_{i-(n-1)}^{i-1})$$

where the notation w_i^j denotes the sequence $w_i \cdots w_j$ and where $w_{-n+2}, \ldots, w_0$ are all taken to be some distinguished beginning-of-sentence token.

[1] These properties hold when constraining feature expectations to be equal to those found in a training set. When constraining expectations to alternate values, the maximum entropy model will not be the maximum likelihood model, and the ME model will not exist if the constraints are inconsistent.

The maximum likelihood estimate $q_{\mathrm{ML}}(w_i|w_{i-(n-1)}^{i-1})$ of the probabilities $\Pr(w_i|w_{i-(n-1)}^{i-1})$ over some training data X can be calculated by simply counting how often the token w_i follows the *history* or context $w_{i-(n-1)}^{i-1}$ and dividing by the total number of times the history occurs, *i.e.*,

$$q_{\mathrm{ML}}(w_i|w_{i-(n-1)}^{i-1}) = \frac{c_X(w_{i-(n-1)}^{i})}{c_X(w_{i-(n-1)}^{i-1})} = \frac{c_X(w_{i-(n-1)}^{i})}{\sum_{w_i} c_X(w_{i-(n-1)}^{i})} \quad .$$

However, maximum likelihood estimation of these probabilities typically leads to overfitting, and instead it is desirable to use smoothed estimates of these values. For example, one simple smoothing technique is to linearly interpolate the maximum likelihood estimate of the n-gram probability $q_{\mathrm{ML}}(w_i|w_{i-(n-1)}^{i-1})$ with an estimate of the $(n-1)$-gram probability $\Pr(w_i|w_{i-(n-2)}^{i-1})$ [14], [15]:

$$q_{\mathrm{int}}(w_i|w_{i-(n-1)}^{i-1}) = \lambda\, q_{\mathrm{ML}}(w_i|w_{i-(n-1)}^{i-1}) + (1-\lambda)\, q_{\mathrm{int}}(w_i|w_{i-(n-2)}^{i-1}), \quad 0 \le \lambda \le 1 \quad . \tag{4}$$

The lower-order estimate can be defined analogously, and the recursion can end with a unigram or uniform distribution. Since the lower-order distributions are less sparsely estimated from the training data, their interpolation generally reduces overfitting. A large number of other smoothing methods for n-gram models have been proposed, *e.g.*, [16], [8], [14], [17], [18], [19].

We present a brief overview of past work in n-gram model smoothing. One basic observation is that the maximum likelihood estimate of the probability of an n-gram that does not occur in the training data is zero and is thus too low, and consequently the ML probabilities of n-grams with nonzero counts are generally too high. This dichotomy motivates the following framework for expressing smoothing methods, which can be used to express most existing smoothing techniques [18]:

$$q_{\mathrm{sm}}(w_i|w_{i-(n-1)}^{i-1}) = \begin{cases} \alpha(w_i|w_{i-(n-1)}^{i-1}) \\ \quad \text{if } c_X(w_{i-(n-1)}^{i}) > 0 \\ \gamma(w_{i-(n-1)}^{i-1}) q_{\mathrm{sm}}(w_i|w_{i-(n-2)}^{i-1}) \\ \quad \text{if } c_X(w_{i-(n-1)}^{i}) = 0 \end{cases} \quad . \tag{5}$$

That is, if an n-gram $w_{i-(n-1)}^{i}$ occurs in the training data, the estimate $\alpha(w_i|w_{i-(n-1)}^{i-1})$ is used; this estimate is generally a discounted version of the maximum likelihood estimate. Otherwise, we back off to a scaled version of the $(n-1)$-gram distribution $q_{\mathrm{sm}}(w_i|w_{i-(n-2)}^{i-1})$, where the lower-order distribution is typically defined analogously to the higher-order distribution. The scaling factor $\gamma(w_{i-(n-1)}^{i-1})$ is chosen to assure that each conditional distribution sums to 1. The algorithm described by (4) can be placed in this framework with the following relations:

$$\begin{aligned} \alpha(w_i|w_{i-(n-1)}^{i-1}) &= q_{\mathrm{int}}(w_i|w_{i-(n-1)}^{i-1}) \\ \gamma(w_{i-(n-1)}^{i-1}) &= 1-\lambda \\ q_{\mathrm{sm}}(w_i|w_{i-(n-2)}^{i-1}) &= q_{\mathrm{int}}(w_i|w_{i-(n-2)}^{i-1}) \quad . \end{aligned}$$

There are three primary distinctions between smoothing algorithms: whether an algorithm is *interpolated* or *backed-off*, what type of discounting is applied to the ML estimate to calculate $\alpha(w_i|w_{i-(n-1)}^{i-1})$, and how lower-order distributions are computed.

In interpolated models, the probability estimate $\alpha(w_i|w_{i-(n-1)}^{i-1})$ of an n-gram $w_{i-(n-1)}^{i}$ with nonzero count depends on the probability assigned to the corresponding $(n-1)$-gram $w_{i-(n-2)}^{i}$, as in (4). In backed-off models, the probability estimate of an n-gram with nonzero count is determined while ignoring information from lower-order distributions. Interpolated models include Jelinek-Mercer smoothing [14] and Witten-Bell smoothing [16]; backed-off models include Katz smoothing [17], absolute discounting [19], and Kneser-Ney smoothing [18].

To describe the different types of discounting, we write $\alpha(w_i|w_{i-(n-1)}^{i-1})$ as

$$\alpha(w_i|w_{i-(n-1)}^{i-1}) = \frac{c_X(w_{i-(n-1)}^{i}) - d(w_{i-(n-1)}^{i})}{c_X(w_{i-(n-1)}^{i-1})} + \beta(w_{i-(n-1)}^{i})$$

where $d(w_{i-(n-1)}^{i})$ can be viewed as the discount in count space from the ML estimate and where $\beta(w_{i-(n-1)}^{i})$ is the contribution from lower-order distributions. The value $\beta(w_{i-(n-1)}^{i})$ is zero for backed-off models and typically $\gamma(w_{i-(n-1)}^{i-1}) q_{\mathrm{sm}}(w_i|w_{i-(n-2)}^{i-1})$ for interpolated models. In *linear discounting*, the discount $d(w_{i-(n-1)}^{i})$ is taken to be proportional to the original count $c_X(w_{i-(n-1)}^{i})$, as in (4) where the discount is $(1-\lambda)\cdot c_X(w_{i-(n-1)}^{i})$. In *absolute discounting*, $d(w_{i-(n-1)}^{i})$ is taken to be a constant $0 \le D \le 1$. In Good-Turing discounting, the discount is calculated using the Good-Turing estimate [20], a theoretically motivated discount that has been shown to be accurate in non-sparse data situations [17], [21]. A brief description of the Good-Turing estimate is given in Section IV-B. Jelinek-Mercer smoothing and Witten-Bell smoothing use linear discounting, Kneser-Ney smoothing uses absolute discounting, and Katz smoothing and Church-Gale smoothing [21] use Good-Turing discounting.

The final major distinction between smoothing algorithms is how the lower-order probability estimates are calculated. While most smoothing methods define the lower-order model $q_{\mathrm{sm}}(w_i|w_{i-(n-2)}^{i-1})$ analogously to the higher-order model $q_{\mathrm{sm}}(w_i|w_{i-(n-1)}^{i-1})$, in Kneser-Ney smoothing a different approach is taken. The $(n-1)$-gram model is chosen to satisfy certain constraints derived from the training data, namely

$$\sum_{w_{i-(n-1)}} c_X(w_{i-(n-1)}^{i-1}) q_{\mathrm{sm}}(w_i|w_{i-(n-1)}^{i-1}) = c_X(w_{i-(n-2)}^{i}) \tag{6}$$

for all $(n-1)$-grams $w_{i-(n-2)}^{i}$. This constraint can be rephrased as: The expected number of times $w_{i-(n-2)}^{i}$ occurs in the training data given the model $q_{\mathrm{sm}}(w_i|w_{i-(n-1)}^{i-1})$

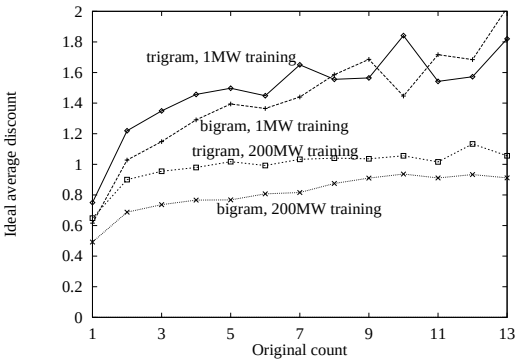


Fig. 1. Ideal average discount for n-grams with given count in training data for 1 million word training set and 200 million word training set, bigram and trigram models

and the history frequencies $c_X(w_{i-(n-1)}^{i-1})$ should equal the actual number of times it occurs. Kneser-Ney smoothing can be applied recursively to lower-order distributions, in which case the constraints (6) are not satisfied exactly. Instead, the right-hand side of the constraints are discounted with absolute discounting.

Chen and Goodman [8] provide an extensive comparison of all of the widely-used smoothing techniques. They evaluate each algorithm on a wide range of training sets through its *perplexity* on test data. The perplexity $\mathrm{PP}_q(X')$ of a model q on a test set X' is the reciprocal of the geometric average probability that the model assigns to each word in the test set. They also use the derivative measure *cross-entropy* $H_q(X') = \log_2 \mathrm{PP}_q(X')$, which can be interpreted as the average number of bits needed to code each word in the test set using the model q. Chen *et al.* [8], [22] also conducted experiments investigating how the cross-entropy of a language model is related to its performance when used in a speech recognition system. They found a strong linear correlation between cross-entropy and recognition word error rate when comparing models that only differ in smoothing.

In terms of perplexity, Chen and Goodman found that Kneser-Ney smoothing and variations consistently outperform all other algorithms. More specifically, they present four main conclusions:

- The primary reason for the superior performance of Kneser-Ney smoothing and variations is the novel manner in which lower-order probability estimates are calculated.
- Absolute discounting is superior to linear discounting. For n-grams with a given count r in the training data, they calculate the average discount in count space $d(w_{i-(n-1)}^{i})$ from the ML estimate that would cause the expected number of these n-grams in a test set to be equal to their actual number (assuming $\beta(w_{i-(n-1)}^{i}) = 0$). This ideal average discount is displayed in Figure 1 for counts $r \leq 13$ for two training sets for bigram and trigram models. From this graph, we see why a fixed discount works well. While Good-Turing discounting is actually better than absolute discounting at predicting the *average* discount, it has yet to be used in such a way as to predict the correct discounts in individual distributions well.
- Interpolated models outperform backed-off models when considering performance on just n-grams with low counts in the training data. This is because lower-order models provide valuable information for estimating the probabilities of n-grams with low counts.
- Adding free parameters to an algorithm and optimizing these parameters on held-out data can improve the performance of an algorithm.

Based on these observations, Chen and Goodman propose an algorithm named *modified Kneser-Ney smoothing* that is found to outperform all other methods considered. It is an interpolated variation of Kneser-Ney smoothing with an augmented version of absolute discounting. Instead of using a single discount D for all n-grams, three separate discounts D_1, D_2, and D_{3+} are used for n-grams with one count, two counts, and three or more counts, respectively. This is motivated by the observation that the ideal discount for one-counts and two-counts is substantially smaller than the ideal discount of larger counts, as shown in Figure 1.

III. Maximum Entropy N-Gram Models

We can construct language models very similar to conventional n-gram models within the maximum entropy framework. The maximum entropy models described in Section I-A are joint models; to create the conditional distributions used in conventional n-gram models we use the framework introduced by Brown *et al.* [23]. Instead of estimating a joint distribution $q(x)$ over samples x, we estimate a conditional distribution $q(y|x)$ over samples (x, y). Instead of constraints as given by (2), we have constraints of the form

$$\sum_{x,y} \tilde{p}(x) q(y|x) f_i(x,y) = \sum_{x,y} \tilde{p}(x,y) f_i(x,y) \quad . \tag{7}$$

This can be interpreted as replacing $q(x,y)$ in the joint formulation with $\tilde{p}(x)q(y|x)$. That is, we assume that history frequencies $\tilde{p}(x)$ are taken from the training data, and we only estimate conditional probabilities. Conditional ME models share many of the same properties as joint models, including being maximum likelihood models, and have computational and performance advantages over joint models in language modeling [24], [5]. A conditional maximum entropy model has the form

$$q_{\mathrm{ME}}(y|x) = \frac{1}{Z_\Lambda(x)} \exp\Big(\sum_{i=1}^{F} \lambda_i f_i(x,y)\Big) \quad . \tag{8}$$

To construct a maximum entropy n-gram model, we take $x = w_{i-(n-1)}^{i-1}$ to be the history and $y = w_i$ to be the following word. For each m-gram $\theta = w_{i-(m-1)}^{i}$ with $m = 1, \ldots, n$ that occurs in the training data, we include a constraint that forces the conditional expectation of θ according to q to be the same as its frequency in the training data. The corresponding features $f_\theta(x,y)$ are

$$f_\theta(x,y) = \begin{cases} 1 & \text{if } (x,y) = w_{i-(n-1)}^{i} \text{ ends in } \theta \\ 0 & \text{otherwise} \end{cases} \quad .$$

Substituting these features into (7) and simplifying, we arrive at constraints of the form

$$\sum_{w_{i-(n-1)}^{i}:\text{suffix}(w_{i-(n-1)}^{i})=\theta} \tilde{p}(w_{i-(n-1)}^{i-1})q(w_i|w_{i-(n-1)}^{i-1}) = \tilde{p}(\theta) \quad . \tag{9}$$

In fact, the only solution to these constraints is $q(y|x) = q_{\text{ML}}(y|x)$. The maximum entropy model is identical to the maximum likelihood n-gram model, and consequently it will be beneficial to smooth the estimates of the model parameters $\Lambda = \{\lambda_\theta\}$.

Remarkably, the set of models given by (8) with n-gram features is basically identical to the set of models described by (5), which we used to express most existing smoothing algorithms for conventional n-gram models.[2] That is, smoothed maximum entropy n-gram models can be viewed as conventional backed-off n-gram models (though later we discuss how they also behave like interpolated models). To see this, consider any maximum entropy n-gram model $q_{\text{ME}}(w_i|w_{i-(n-1)}^{i-1})$ with parameters Λ. To express this model as a backed-off model, let us define a set of m-gram models $q_{\text{ME}}(w_i|w_{i-(m-1)}^{i-1})$ for $m = 1, \ldots, n$ as in (8), where each m-gram model only contains features corresponding to word sequences up to length m, and where all models inherit the parameters Λ of the original model. To express the original model in terms of (5), we take

$$\begin{aligned} \alpha(w_i|w_{i-(m-1)}^{i-1}) &= q_{\text{ME}}(w_i|w_{i-(m-1)}^{i-1}) \\ \gamma(w_i|w_{i-(m-1)}^{i-1}) &= \frac{Z_\Lambda(w_{i-(m-2)}^{i-1})}{Z_\Lambda(w_{i-(m-1)}^{i-1})} \quad . \end{aligned}$$

Recursively expanding (5) using these relations will yield (8) for the original model.

Because smoothing λ_θ estimates in maximum entropy n-gram models and smoothing conventional n-gram models both consider the same class of models, we can discuss both classes using the same concepts and it seems likely that we can transfer knowledge gleaned from smoothing conventional n-gram models to smoothing maximum entropy models. Furthermore, this equivalence has the pragmatic advantage that software tools developed for conventional n-gram models can also be utilized for maximum entropy n-gram models. For example, maximum entropy n-gram models can be expressed efficiently in the standard ARPA format for conventional n-gram models [25].

IV. Smoothing Maximum Entropy Models

In this section, we survey previous work in maximum entropy model smoothing, including constraint exclusion, Good-Turing discounting, fuzzy maximum entropy, and fat constraints. In describing these methods, we sometimes use the joint maximum entropy formulation for simplicity. All of these techniques apply equally well to conditional ME models; the analogous conditional ME equations can be derived by replacing x with (x, y) and $q(x)$ with $\tilde{p}(x)q(y|x)$.

A. Constraint Exclusion

One simple smoothing technique is to simply exclude some of the constraints in a model. When we remove constraints from a model within the maximum entropy framework, we will generally produce a model with higher entropy than the original, *i.e.*, a model that is smoother or more uniform. For example, in a maximum entropy n-gram model we might leave out constraints for n-grams that occur fewer than a certain number of times in the training data. This heuristic will tend to exclude constraints for those n-grams for which we do not have reliable estimates of their true frequency.

The technique of leaving out constraints has been widely used in maximum entropy modeling. For maximum entropy n-gram models, omitting constraints for n-grams with low counts is analogous to using *count cutoffs* for conventional n-gram models [6], [5]. Feature induction [2], [3], where only a subset of a set of candidate features are included in a model based on some selection criteria, can also be considered to be a form of constraint exclusion. Constraint exclusion can be used in conjunction with all of the ME smoothing techniques to be described subsequently.

B. Good-Turing Discounting

Good-Turing discounting has been proposed by Lau [6] and Rosenfeld [5] and can be viewed as the maximum entropy analog to Katz smoothing for conventional n-gram models. They observe that the marginals of the model $q(y|x)$ should not be constrained to be exactly those of the empirical distribution $\tilde{p}(y|x)$,[3] but instead target values should be discounted as in conventional n-gram smoothing. Instead of constraints as given by (9), they propose the following constraints

$$\sum_{w_{i-(n-1)}^{i}:\text{suffix}(w_{i-(n-1)}^{i})=\theta} \tilde{p}(w_{i-(n-1)}^{i-1})q(w_i|w_{i-(n-1)}^{i-1}) = \tilde{p}_{\text{GT}}(\theta)$$

where $\tilde{p}_{\text{GT}}(\theta)$ is the Good-Turing estimate of the frequency of θ.

The Good-Turing estimate [20] is a theoretically motivated method for estimating the average discount for an event based on its count in the training data. For an event that occurs r times in N samples, in contrast with the maximum likelihood estimate $\frac{r}{N}$, the Good-Turing estimate of the event's true frequency is $\frac{r^*}{N}$ where

$$r^* = \frac{n_{r+1}}{n_r}(r+1)$$

and where n_r is the number of members of the population with exactly r counts. Katz [17] suggests applying this estimate to each joint m-gram distribution, $m = 1, \ldots, n$,

[2] The equivalence is not exact as exponential models cannot express probabilities equal to zero or one. In addition, for the equivalence to hold the unigram model used in (5) must assign the same probability to all words not occurring in the training data. This is generally the case with existing smoothing algorithms.

[3] More precisely, when we say the marginals of the model $q(y|x)$, we mean the marginals of the associated joint model $q(x, y) \stackrel{\text{def}}{=} \tilde{p}(x)q(y|x)$, and similarly for $\tilde{p}(y|x)$.

separately. Furthermore, as n_r can be very low or zero for large r, Katz proposes a method where n-grams with large counts are not discounted and discounts for low counts are adjusted to compensate. Lau and Rosenfeld use the Katz variation of Good-Turing discounting.

However, when constraining marginals of a model to Good-Turing discounted marginals of the training data, the constraints may no longer be consistent and a maximum entropy model may not exist. For example, in a trigram model consider features that constrain the frequencies of the n-grams TIC TAC TOE and TAC TOE and assume that the word TAC only follows the word TIC in the training data. Then, we will have the constraints

$$\tilde{p}(\text{TIC TAC})q(\text{TOE}|\text{TIC TAC}) = \tilde{p}_{\text{GT}}(\text{TIC TAC TOE})$$

and

$$\sum_{w_{i-2}} \tilde{p}(w_{i-2}\ \text{TAC})q(\text{TOE}|w_{i-2}\ \text{TAC}) = \tilde{p}(\text{TIC TAC})q(\text{TOE}|\text{TIC TAC}) = \tilde{p}_{\text{GT}}(\text{TAC TOE})$$

In general, we will have $\tilde{p}_{\text{GT}}(\text{TIC TAC TOE}) \neq \tilde{p}_{\text{GT}}(\text{TAC TOE})$ since discounts for n-grams of different length are calculated independently; consequently, these constraints will be inconsistent. In practice, there are no dire consequences to having inconsistent constraints. While training algorithms such as iterative scaling may not converge, a reasonable procedure is to stop training once performance on some held-out set stops improving. However, inconsistency is symptomatic of constraints that will lead to poor parameter estimates.

Lau [6] compares the performance of Good-Turing discounting for smoothing ME n-gram models with deleted interpolation [14], a variation of Jelinek-Mercer smoothing, for conventional n-gram models. For a 5 million word training set of Wall Street Journal text, deleted interpolation yielded a perplexity of 225 on an 870,000 word test set. The maximum entropy model yielded a slightly superior perplexity of 221, where constraints for all n-grams that occurred only once in the training data were excluded from the ME model. However, later results by Chen and Goodman [8] strongly indicate that other smoothing methods for conventional n-gram models, such as modified Kneser-Ney smoothing, would outperform deleted interpolation by a much larger margin.

C. Fuzzy Maximum Entropy

In the fuzzy maximum entropy framework developed by Della Pietra and Della Pietra [1], instead of requiring that constraints are satisfied exactly, a penalty is associated with inexact constraint satisfaction. Finding the maximum entropy model is equivalent to finding the model $q(x)$ satisfying the given constraints that minimizes the Kullback-Leibler distance $D(q \parallel p_{\text{unif}})$ from the uniform model $p_{\text{unif}}(x)$. In fuzzy maximum entropy, the objective function is taken to be

$$D(q \parallel p_{\text{unif}}) + U(q) \tag{10}$$

where $U(\cdot)$ is a penalty function minimized when constraints are satisfied exactly. Among others, Della Pietra and Della Pietra propose a penalty function of the form

$$U(q) = \sum_{i=1}^{F} \frac{1}{2\rho_i^2} \left[\sum_x q(x) f_i(x) - \sum_x \tilde{p}(x) f_i(x) \right]^2 \quad .$$

This penalty function can be interpreted as the logarithm of a Gaussian distribution with diagonal covariance centered around the target constraint values. The variance ρ_i^2 associated with feature $f_i(x)$ can be estimated from the empirical distribution of $f_i(x)$ in the training data, and a variant of generalized iterative scaling has been developed to find the optimal model under this objective function [6].

Just as maximum entropy models have an alternate or *dual* specification, *i.e.*, a maximum entropy model is also the maximum likelihood model among the set of models given by (3), fuzzy maximum entropy models also have a dual formulation. Using the maximum likelihood perspective of ME models, we can equivalently view fuzzy maximum entropy as imposing a Gaussian prior centered around $\vec{0}$ on the parameters Λ, where we perform maximum *a posteriori* instead of maximum likelihood estimation.[4] From this perspective, we see that the prior nudges the λ parameters toward zero, thereby making the model more uniform.

More precisely, we can equate finding the regular maximum entropy model with finding parameters Λ that maximize the log-likelihood $L_X(\Lambda)$ of the training data X

$$L_X(\Lambda) = \sum_x \tilde{p}(x) \log q_\Lambda(x) \quad .$$

With the Gaussian prior, which we take to have diagonal covariance, our objective function $L'_X(\Lambda)$ becomes

$$\begin{aligned} L'_X(\Lambda) &= L_X(\Lambda) + \sum_{i=1}^{F} \log \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{\lambda_i^2}{2\sigma_i^2}) \\ &= L_X(\Lambda) - \sum_{i=1}^{F} \frac{\lambda_i^2}{2\sigma_i^2} + \text{const}(\Lambda) \quad , \end{aligned} \tag{11}$$

where the $\sigma_i^2 = \frac{1}{\rho_i^2}$ are the variances of the Gaussian.

Since the logarithm of the Gaussian prior is concave, the objective function is still concave in Λ and it is still straightforward to find the optimal model. For instance, we can make a simple modification to improved iterative scaling to find the MAP model [27]. The original update of each λ_i in this algorithm is to take

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

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

$$\sum_x \tilde{p}(x) f_i(x) = \sum_x q_{\Lambda^{(t)}}(x) f_i(x) \exp(\delta_i^{(t)} f^{\#}(x)) \tag{12}$$

[4] In an earlier incarnation of this work [26], the authors were unaware of the equivalence of fuzzy maximum entropy and the Gaussian prior, and mistakenly attributed the Gaussian prior to John Lafferty.

and where $f^{\#}(x) = \sum_i f_i(x)$. With the Gaussian prior, (12) is replaced with

$$\sum_x \tilde{p}(x) f_i(x) = \sum_x q_{\Lambda^{(t)}}(x) f_i(x) \exp(\delta_i^{(t)} f^{\#}(x)) + \frac{\lambda_i^{(t)} + \delta_i^{(t)}}{\sigma_i^2} \quad (13)$$

As the right-hand side of this equation is strictly monotonic in $\delta_i^{(t)}$, it is relatively easy to find its solution using a search algorithm. We derive this modified update rule in the appendix. This modification adds very little computational cost to the improved iterative scaling algorithm.

Lau [6] constructed a fuzzy maximum entropy n-gram model excluding all n-grams with only one count using the data sets described in Section IV-B, yielding a perplexity of 230. This is slightly worse than the perplexities achieved by the deleted interpolation and Good-Turing discounted ME models.

D. Fat Constraints

Other methods for relaxing constraints include work by Newman [28] and Khudanpur [29]. In these algorithms, instead of selecting the maximum entropy model over models $q(x)$ that satisfy a set of constraints exactly, they only require that the given marginals of $q(x)$ fall in some range around the target values. Newman suggests a constraint of the form

$$\sum_{i=1}^{F} W_i \left[\sum_x q(x) f_i(x) - \sum_x \tilde{p}(x) f_i(x) \right]^2 \leq \sigma^2$$

with feature weights W_i for the task of estimating power spectra. Khudanpur suggests constraints of the form

$$\alpha_i \leq \sum_x q(x) f_i(x) \leq \beta_i, \quad i = 1, \ldots, F \quad .$$

Both of these approaches can be viewed as instances of the fuzzy maximum entropy framework. Instead of a smooth function, the penalty $U(q)$ is taken to be zero if q satisfies the relaxed constraints and infinite otherwise. These types of methods have yet to be applied to language modeling.

V. Fuzzy Maximum Entropy and Conventional N-Gram Smoothing

In Section II, we listed four factors that were found by Chen and Goodman to significantly affect n-gram smoothing performance. It is informative to assess fuzzy maximum entropy smoothing according to these four criteria.

First, they point out that the modified lower-order distributions of Kneser-Ney smoothing is the primary reason for its superiority among conventional n-gram smoothing algorithms. Recall that these distributions are chosen to satisfy marginal constraints as given in (6). However, this set of constraints is *identical* to the corresponding maximum entropy constraints for $(n-1)$-grams, as given by (9). Thus, smoothed maximum entropy n-gram models potentially have similar modified lower-order distributions as in Kneser-Ney smoothing.

However, fuzzy maximum entropy models do not satisfy the ME constraints exactly. Instead, the constraints that are satisfied have the form

$$\sum_x \tilde{p}(x) f_i(x) - \frac{\lambda_i}{\sigma_i^2} = \sum_x q_\Lambda(x) f_i(x) \quad . \quad (14)$$

That is, the empirical expectations $\sum_x \tilde{p}(x) f_i(x)$ are now "discounted" by the amount λ_i / σ_i^2. (In ME n-gram models, most λ_i are positive.) Qualitatively, this is even more desirable than meeting the targets exactly, as empirical frequencies tend to be higher than true frequencies for events with nonzero counts. Analogous behavior is produced with Kneser-Ney smoothing when applied recursively to lower-order distributions. In this case, target counts are discounted through absolute discounting. A derivation of (14) is given in the appendix.

Second, Chen and Goodman point out that absolute discounting is superior to the other types of discounting considered, and that using a different discount for one-counts and two-counts and a flat discount thereafter as in modified Kneser-Ney smoothing performs even better. With fuzzy maximum entropy, the discount for an n-gram θ is linear in λ_θ as can be seen from (14). As the probability assigned to θ by q_Λ grows exponentially in λ_θ, λ_θ grows logarithmically as a function of the target probability or count. In other words, roughly speaking fuzzy maximum entropy smoothing translates to *logarithmic* discounting. This is a qualitatively appealing model of the ideal average discount displayed in Figure 1 and is more elegant than using multiple flat discounts.

We can contrast the Gaussian prior on Λ parameters of fuzzy maximum entropy with previous work in n-gram smoothing where priors have been applied directly in probability space. MacKay and Peto [30] use a Dirichlet prior and Nádas [31] uses a Beta prior, both resulting in linear discounting which has been shown to perform suboptimally. Applying a prior to parameters proportional to log-probabilities instead of to probabilities produces a "softer" prior, which in this case leads to behavior closer to the empirical ideal.

Third, Chen and Goodman report that interpolated models outperform backed-off models on n-grams with low counts, as lower-order models provide valuable information for estimating these probabilities. Happily, a fuzzy maximum entropy model behaves like an interpolated model as n-gram probability estimates depend on lower-order information. This follows trivially from the observation that the probability q_Λ assigns to an n-gram θ depends on the parameter values $\lambda_{\theta'}$ for all n-grams θ' that are suffixes of θ. However, fuzzy maximum entropy models use the information from lower-order models in a meaningful way. Viewing fuzzy ME as imposing a Gaussian prior centered around $\vec{0}$, we see that fuzzy ME smoothing tends to adjust λ_θ towards zero for any n-gram θ; when λ_θ is zero, the corresponding feature has no effect on the model, and the lower-order n-gram probability estimate is used. In other

words, the prior adjusts n-gram probabilities towards the lower-order probability estimate, as is desirable.

Finally, Chen and Goodman note that additional tunable parameters can improve current smoothing methods. For fuzzy maximum entropy smoothing, the natural free parameters are the variances σ_i. In the basic version of fuzzy maximum entropy that we implemented, we had n free parameters σ_m, $m = 1, \ldots, n$, where all m-grams of the same length were constrained to have the same variance σ_m^2.

As fuzzy maximum entropy smoothing satisfies all of the desiderata listed by Chen and Goodman, it may perform competitively in n-gram smoothing. The experiments in Section VI show that this is indeed the case.

VI. Experiments

To compare the performance of maximum entropy and conventional n-gram smoothing techniques, we ran experiments over many training set sizes using several different text corpora for both bigram and trigram models.

A. Methodology

Of the conventional n-gram smoothing techniques, we implemented Katz smoothing [17], which is perhaps the most popular algorithm in practice, and modified Kneser-Ney smoothing [8], which has been shown to outperform all other widely-used techniques. In addition, we implemented the variation of Jelinek-Mercer smoothing given by (4) where instead of a single λ parameter a different λ_m is used for each level of the n-gram model. This method does not perform particularly well, but is used as a baseline algorithm for expository purposes. We refer to these three implementations with the mnemonics `katz`, `kneser-ney-mod`, and `baseline`, respectively.

We also implemented several maximum entropy smoothing techniques. For each technique, all λ_θ parameters are initialized to zero and improved iterative scaling is applied to train the model. Iterative scaling is terminated when the perplexity of a held-out set no longer decreases appreciably. Cluster expansion [32] is employed to reduce computation. In the implementation `ME-no-smooth`, no smoothing is performed. (Since training is terminated when performance on a held-out set no longer improves, no probabilities will converge to zero as in the case where training is continued to convergence.) The algorithm `ME-disc-katz` is an implementation of Good-Turing discounting as described in Section IV-B. The algorithm `ME-fuzzy` is an implementation of fuzzy maximum entropy smoothing as described in Section IV-C. As mentioned earlier, this method has n free parameters σ_m, one for each level of the n-gram model.

In conjunction with each of the maximum entropy smoothing techniques, we also considered using constraint exclusion or count cutoffs as discussed in Section IV-A. We tried several configurations, and found that excluding constraints only for trigrams that occur less than twice in the training data in trigram models (and for analogous bigrams in bigram models) generally gave the best performance. We call the corresponding implementations `ME-no-smooth-cutoff`, `ME-disc-katz-cutoff`, and `ME-fuzzy-cutoff`. As discussed elsewhere [8], using count cutoffs for conventional n-gram smoothing methods generally leads to worse performance for the methods that perform well.

We used data from four sources: the Brown corpus, which contains text from a number of miscellaneous sources [33]; Wall Street Journal (WSJ) newspaper text [34]; the Broadcast News (BN) corpus, which contains transcriptions of television and radio news shows [35]; and the Switchboard (SWB) corpus, which contains transcriptions of telephone conversations [36]. In each experiment, we selected a training set of a given length from one source, and two held-out sets from the same source. The first held-out set was used to optimize the parameters of each smoothing algorithm, *e.g.*, the σ_m parameters of `ME-fuzzy` or the discounts D_* of modified Kneser-Ney smoothing. Parameters were selected to minimize the perplexity of the held-out set; Powell's search algorithm [37] was used to perform this search. This held-out set was also used to decide when to terminate iterative scaling for the ME models. The second held-out set was used to evaluate the final perplexity of each smoothing algorithm.

For each data source, we ran experiments using training sets from 100 sentences (about 2,000 words) to around 100,000 sentences (about 2 million words). Held-out sets were 2,500 sentences. While training sets for language models may reach hundreds of millions of words in practice, we were unable to consider larger training sets than we did due to computational limitations. Training maximum entropy n-gram models requires a great deal more computation than training conventional n-gram models. In addition, when considering multiple parameter settings in the Powell search (as for the σ_m parameters in `ME-fuzzy`), the iterative scaling algorithm must be applied separately for each parameter setting. To train a single model using method `ME-fuzzy` for a 2 million word training set required around six hours of computation on a 400 MHz Pentium II computer. Substantially larger training sets are feasible if parameter optimization is not used. In contrast, constructing the conventional n-gram models on a 2 million word training set required only a few minutes. However, maximum entropy n-gram models can be applied just as efficiently as conventional n-gram models once built.

Our data sets are identical to those used by Chen and Goodman [8] and consequently our results are directly comparable to the analogous results presented by Chen and Goodman. More details of our methodology can be found in that work.

B. Results

In Figure 2, we display the cross-entropy of the baseline Jelinek-Mercer smoothing algorithm over a range of training set sizes on several corpora. In the graphs to follow, we display the performance of each algorithm as the difference of its cross-entropy on the test set from the cross-entropy of the baseline method (using the same training set) to facilitate visualization. Each point in the graphs

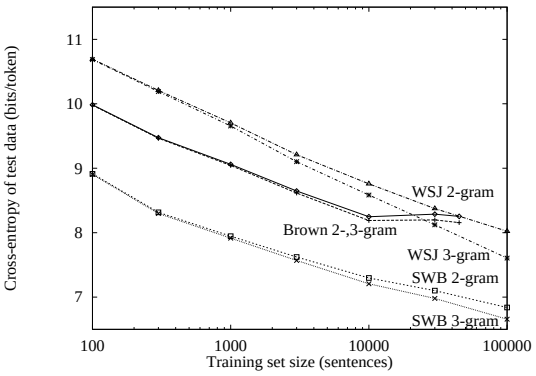


Fig. 2. Cross-entropy of baseline smoothing algorithm on test set over multiple training set sizes on Brown, Switchboard, and Wall Street Journal corpora

presented here represents a single experiment; for an analysis of the standard error of these observations refer to Chen and Goodman [8]. To give a rough idea of the statistical error involved, in Figures 4 and 5 the difference between `kneser-ney-mod` and `ME-fuzzy` may not be significant, while the difference between these two algorithms and all of the others almost certainly is for almost every data point.

In Figure 3, we compare the performance of the various maximum entropy smoothing algorithms over multiple training set sizes using Wall Street Journal data. The left graph is for bigram models and the right graph is for trigram models. We see that `ME-no-smooth` is outperformed by the other algorithms by a large margin, demonstrating the necessity of smoothing for maximum entropy models. Using count cutoffs as in `ME-no-smooth-cutoff` leads to substantially better performance, though still poor. Of the remaining algorithms, fuzzy maximum entropy smoothing performed best, with `ME-disc-katz` beating `ME-disc-katz-cutoff` for bigram models and the reverse happening over most data sets for trigram models. Though not shown here, we see similar behavior in experiments on the other three corpora. The algorithm `ME-fuzzy-cutoff` is consistently outperformed by `ME-fuzzy` and has been omitted from all graphs.[5]

In Figures 4 and 5, we compare the performance of maximum entropy smoothing algorithms with conventional n-gram smoothing algorithms over several corpora.[6] Of the conventional smoothing methods, we see that Katz smoothing generally outperforms the baseline and that modified Kneser-Ney smoothing is substantially better. Of the maximum entropy methods, we see that `ME-disc-katz` performs comparably to Katz smoothing for bigram models, while `ME-disc-katz-cutoff` performs somewhat worse. For trigram models, `ME-disc-katz-cutoff` is superior to Katz smoothing, with `ME-disc-katz` generally worse. The method `ME-fuzzy` performs about as well as modified Kneser-Ney smoothing, and is slightly better over most data sets. Thus, fuzzy maximum entropy smoothing performs as well as or better than all other widely-used algorithms for smoothing n-gram models.

To investigate how the logarithmic discounting of fuzzy maximum entropy smoothing compares to the multiple absolute discounts of modified Kneser-Ney smoothing, we computed how closely the expected number of certain n-grams in a test set according to each model matched the actual number of those n-grams in the test set. In particular, for all n-grams occurring exactly r times in a 750,000 word training set X for some r, we computed the ratio of the expected number of times these n-grams occurred in a 10,000,000 word test set X' to the actual number of times they occurred:

$$\frac{\sum_{w_{i-(n-1)}^{i}:c_X(w_{i-(n-1)}^{i})=r} c_{X'}(w_{i-(n-1)}^{i-1}) q(w_i|w_{i-(n-1)}^{i-1})}{\sum_{w_{i-(n-1)}^{i}:c_X(w_{i-(n-1)}^{i})=r} c_{X'}(w_{i-(n-1)}^{i})}.$$

These ratios are displayed for $r < 40$ in Figure 6 for bigram and trigram models. Fuzzy maximum entropy achieves ratios closer to the ideal value of one than modified Kneser-Ney smoothing for most r, which is evidence that fuzzy maximum entropy smoothing is superior to multiple flat discounts at predicting correct average discounts.

We also investigated how the number of independent variance parameters used with fuzzy maximum entropy smoothing affects performance. In the original implementation `ME-fuzzy`, a different σ_m is used for each level of the n-gram model.[7] We also considered using a single σ over the whole model (`ME-fuzzy-1`), and using three parameters $\sigma_{m,1}$, $\sigma_{m,2}$, and $\sigma_{m,3+}$ for each level of the n-gram model, to be applied to m-grams with 1, 2, or 3 or more counts in the training data, respectively. This latter parameterization (`ME-fuzzy-3n`) is analogous to the parameterization of modified Kneser-Ney smoothing. The performance of these three variations on the Wall Street Journal corpus is displayed in Figure 7. The variations `ME-fuzzy` and `ME-fuzzy-3n` yield almost identical performance, and the variation `ME-fuzzy-1` performs slightly worse.[8] As having separate variances for each n-gram level leads to improved performance, this is a useful distinction to make. We also investigated many other parameter-tying schemes, but none significantly outperformed this simple technique.

[5] We also ran experiments using *complemented* n-grams [5], where each n-gram feature is nonzero only when no longer n-gram feature is nonzero. This resulted in significantly inferior performance.

[6] The large spikes in the Switchboard graphs are discussed by Chen and Goodman [8]. They are caused by a duplicated segment of text in the training set.

[7] The optimal variances σ_m^2 found by the parameter search were generally in the range $1.5N < \sigma_m^2 < 5N$ for $m > 1$, where N is the size of the training set. The optimal values found for σ_1^2 varied widely; for large data sets, it sometimes approached infinity, corresponding to no smoothing for unigram marginals. The linear dependence of σ_m^2 on N is an artifact of the way we define the fuzzy ME objective function in (11). If we replace $L_X(\Lambda)$, the log-likelihood per event, with $N \times L_X(\Lambda)$, the total log-likelihood of the training set, the corresponding optimal σ_m^2 would be largely independent of training set size and take on moderate values (for $m > 1$). While this alternate formulation is perhaps more intuitive when using a prior, we chose the given formulation for notational expedience.

[8] A variation with more parameters may not outperform a variation with fewer parameters due to a mismatch between the evaluation set and the held-out set used to optimize parameters or due to search errors in parameter optimization.

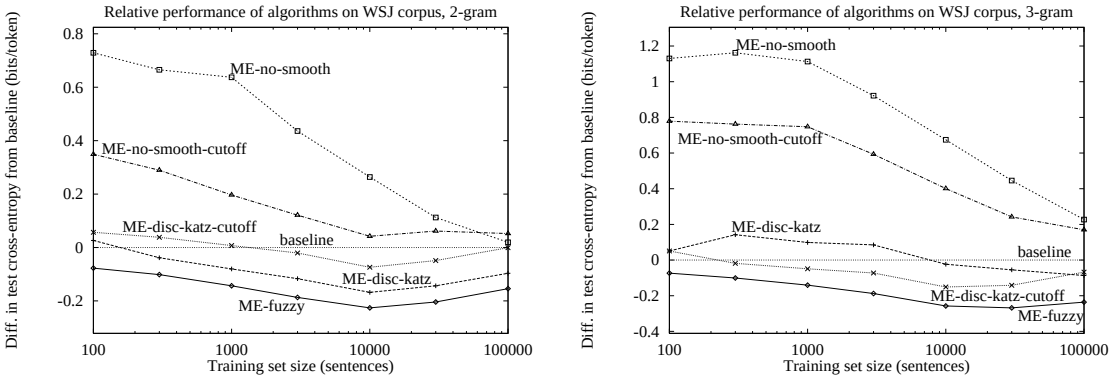


Fig. 3. Performance relative to baseline of various maximum entropy smoothing algorithms over multiple training set sizes on the Wall Street Journal corpus, bigram and trigram models

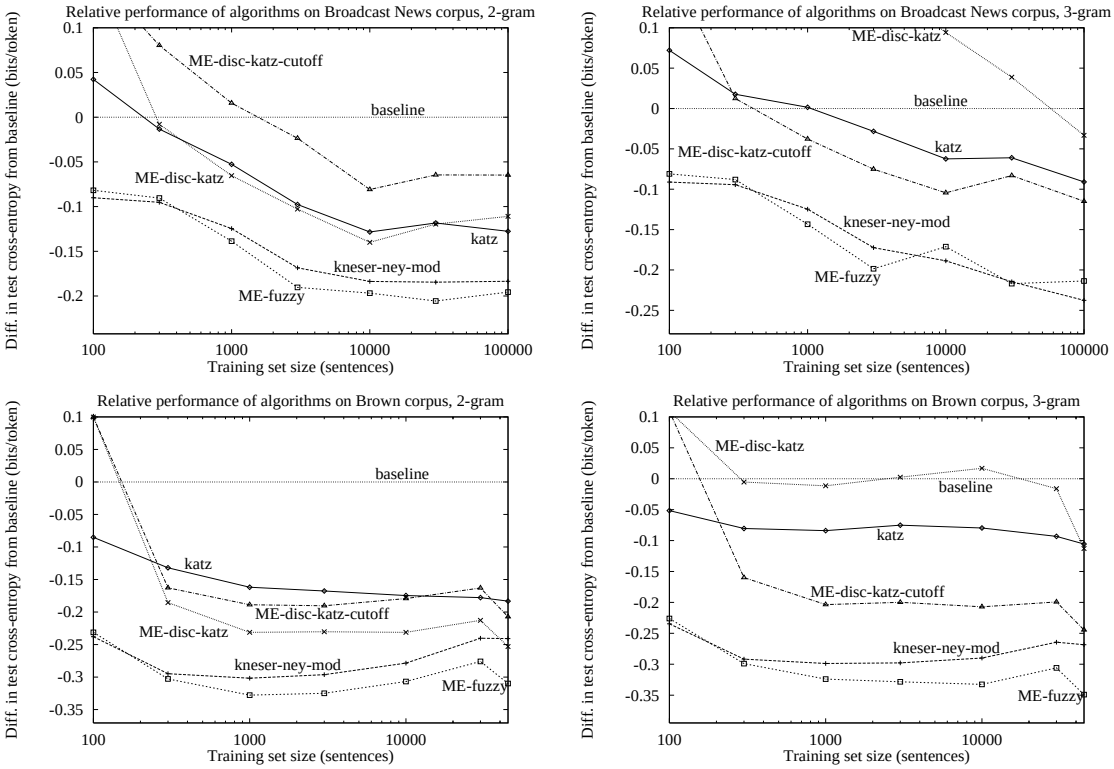


Fig. 4. Performance relative to baseline of various smoothing algorithms over multiple training set sizes on the Broadcast News and Brown corpora, bigram and trigram models

	training set size (sentences)							
	100		3000		100000		1000000	
	WER	entr.	WER	entr.	WER	entr.	WER	entr.
ME-disc-katz	52.9%	11.84	46.8%	10.29	41.0%	9.06	37.3%	8.19
katz	52.4%	11.83	46.6%	10.12	41.1%	8.89	37.2%	8.12
kneser-ney-mod	52.8%	11.69	46.5%	9.96	40.2%	8.68	36.4%	7.94
ME-fuzzy	52.9%	11.70	46.2%	9.94	39.9%	8.68	36.5%	7.96

TABLE I

SPEECH RECOGNITION WORD ERROR RATES AND TEST SET CROSS-ENTROPIES OF VARIOUS SMOOTHING ALGORITHMS OVER SEVERAL TRAINING SET SIZES ON BROADCAST NEWS DATA

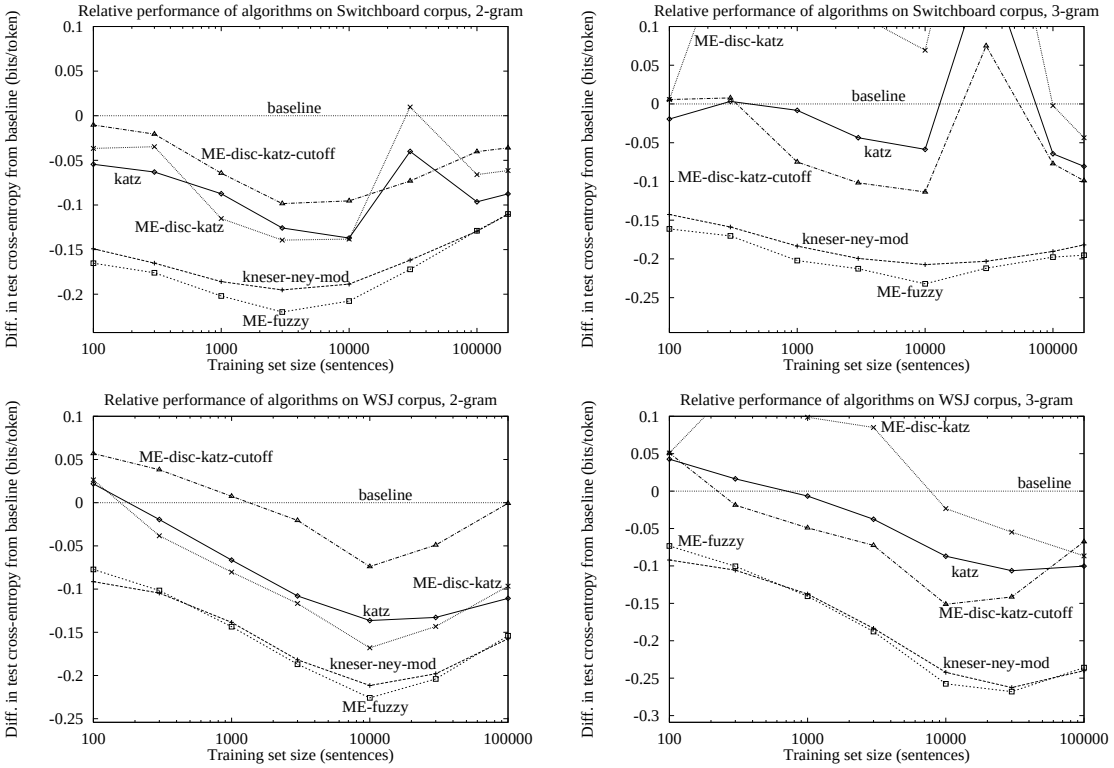


Fig. 5. Performance relative to baseline of various smoothing algorithms over multiple training set sizes on the Switchboard and Wall Street Journal corpora, bigram and trigram models

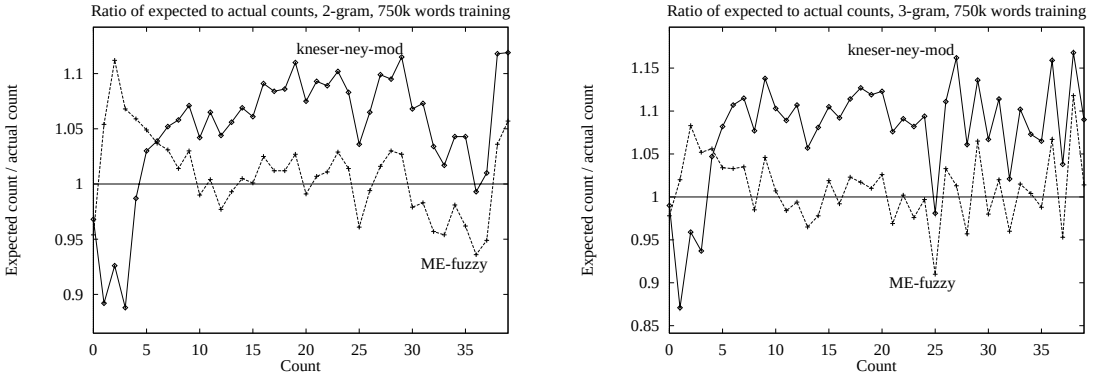


Fig. 6. Ratio of expected number to actual number in test set of n-grams with a given count in training data, 750,000 word Wall Street Journal training set, bigram and trigram models

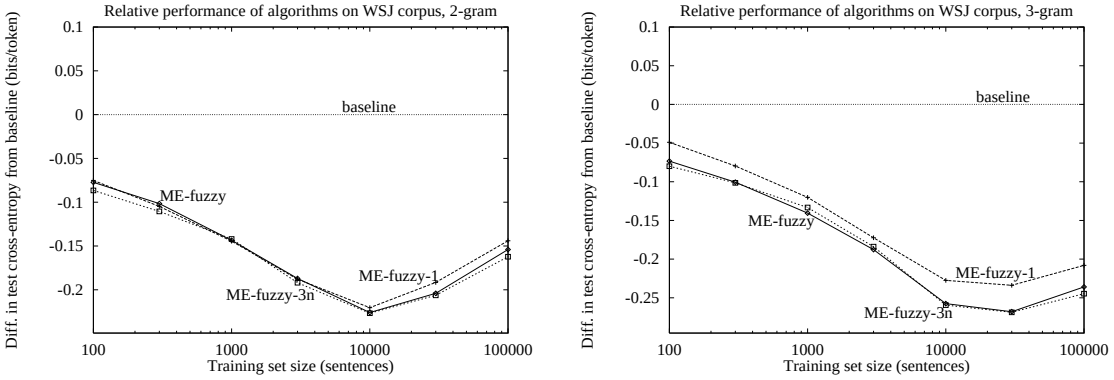


Fig. 7. Performance relative to baseline of different parameterizations of fuzzy maximum entropy smoothing over multiple training set sizes on the Wall Street Journal corpus, bigram and trigram models

Finally, we examined how smoothing affects the performance of n-gram models in a speech recognition task. We constructed trigram language models for each of four smoothing algorithms for four different training set sizes on Broadcast News data. Due to the computational demands of maximum entropy training, we were unable to consider training sets larger than 1,000,000 sentences, though substantially more relevant training material is available. We calculated word error rates by rescoring lattices produced by the Sphinx-III speech recognition system for the TREC-7 spoken document retrieval task [38]. The word error rates achieved on a 33,000-word test set are displayed in Table I, as well as the cross-entropy of each model on the correct transcript of the test set.[9]

As expected, `kneser-ney-mod` and `ME-fuzzy` consistently yielded the lowest cross-entropies. However, the method `katz` achieved the lowest word error rate for the smallest training set, with `kneser-ney-mod` and `ME-fuzzy` performing better for the three larger training sets. The methods `ME-fuzzy` and `kneser-ney-mod` yielded very similar performance in terms of cross-entropy and word error rate over all of the training sets, and outperformed the other two algorithms tested by as much as 1% absolute in word error rate on the larger training sets. Thus, fuzzy maximum entropy smoothing seems to yield competitive performance on a speech recognition task.

VII. Discussion

It has been argued that maximum entropy models do not require smoothing because they are already as uniform or smooth as possible given the constraints. However, maximum entropy models can be viewed as maximum likelihood exponential models, and have similar properties as other maximum likelihood methods. For example, as can be seen in Figure 3, when data is plentiful, smoothing has a smaller effect, and when data is sparse, smoothing is essential.

In many tasks including language modeling, it has been found that superior performance can be achieved by constructing very large models (so parameters are sparsely estimated) and then smoothing them. Thus, for maximum entropy models to be competitive with other techniques in these domains, we need effective maximum entropy smoothing algorithms.

In this work, we showed that the fuzzy maximum entropy method can be used to smooth maximum entropy n-gram models to achieve performance equal to or superior to that of all other techniques for smoothing n-gram models, a field which has an extensive body of associated research. This is the first clear demonstration that a maximum entropy smoothing method can be as effective as smoothing techniques for other types of models, and makes it possible to construct maximum entropy models in sparse data situations without loss of performance. Furthermore, it adds virtually no computational cost to the maximum entropy training procedure. However, because of the large underlying computational cost of maximum entropy algorithms, building maximum entropy models for very large data sets is still a challenging problem.

While fuzzy maximum entropy smoothing can be expressed very simply, we show that it possesses all of the desirable qualities of n-gram smoothing noted by Chen and Goodman from empirical analysis. In addition, it achieves its excellent performance using fewer parameters than the comparably performing modified Kneser-Ney smoothing.

Fuzzy maximum entropy smoothing and variations of Kneser-Ney smoothing consistently outperform other smoothing techniques. The distinction between these algorithms and the others is their use of modified lower-order distributions as described in Sections II and V. These distributions are chosen to satisfy certain marginal constraints derived from the training data. Thus, the use of marginal constraints may be a powerful technique for designing novel smoothing algorithms, whether for language modeling or for other domains. Enforcing marginal constraints would mark a significant departure from traditional techniques used in smoothing.

In addition, fuzzy maximum entropy smoothing can be viewed as imposing a qualitatively different prior than has been used previously in n-gram smoothing. As touched on in Section V, linear discounting can be motivated through a Dirichlet or Beta prior on probabilities [30], [31], but it has been shown to perform poorly. While absolute discounting yields better performance, it is unclear how to elegantly express this technique through a prior distribution. In contrast, the Gaussian prior of fuzzy maximum entropy is applied to λ_θ parameters which are linear in log-probability, and leads to logarithmic discounting. This simple prior yields discounting that is qualitatively and quantitatively similar to the empirical ideal, and suggests that logarithmic discounting may be employed profitably in a conventional n-gram smoothing technique.

Not only can fuzzy maximum entropy smoothing be applied to maximum entropy modeling, but it can also be applied in the more general *minimum divergence* paradigm [40], [41]. Maximizing entropy is equivalent to finding the model with the smallest Kullback-Leibler divergence from the uniform distribution. In minimum divergence modeling, one selects the model satisfying the given constraints closest to some default distribution $q_0(x)$. The model $q_0(x)$ can be used to express prior knowledge about the domain. Minimum divergence models have the form

$$q_{\text{MD}}(x) = \frac{1}{Z_\Lambda}\, q_0(x)\, \exp\Big(\sum_{i=1}^{F} \lambda_i f_i(x)\Big) \quad .$$

The analysis in Section IV-C applies to these models without modification, except that $p_{\text{unif}}(x)$ is replaced by $q_0(x)$ in (10).

Maximum entropy modeling has advantages over competing approaches in terms of elegance, generality, and performance, and the fuzzy maximum entropy method is a

[9] Because of the enormous amount of speech data in the TREC-7 task, Sphinx-III was configured to run with only a single decoding pass and with narrow search beams. With several passes, wider beams, and a larger language model, Sphinx-III achieved a word-error rate of 23.8% on a similar Broadcast News task [39].

powerful tool for smoothing general ME models. Whether fuzzy maximum entropy smoothing proves superior to other algorithms in domains other than n-gram modeling is still an open empirical question. In n-gram models, no features partially overlap each other (*i.e.*, any two features are either never active on the same event, or one feature is active for a superset of the events that the other is active), and this is not the case in general. However, promising results have been achieved with fuzzy maximum entropy smoothing in text classification [42], as well as in topic adaptation for language modeling [43]. In addition, how parameters should be tied in other domains has yet to be explored.[10] Nonetheless, our results and analysis justify the choice of fuzzy maximum entropy smoothing for use in n-gram modeling, and strongly suggest its use in other situations as well.

Appendix

I. Derivation of Modified Constraints and Modified Iterative Scaling for Fuzzy Maximum Entropy Smoothing

In this section, we derive the modified constraints given in (14) and the modified update for improved iterative scaling given in (13) for fuzzy maximum entropy smoothing. We use the conditional ME formulation. For further details about improved iterative scaling such as proof of convergence, refer to [3].

To derive the modified constraints, we take the partial derivatives of the objective function given in (11) with respect to the parameters λ_i and set them to zero.

$$\begin{aligned}
L'_X(\Lambda) &= \sum_{x,y} \tilde{p}(x,y) \log q_\Lambda(y|x) - \sum_{i=1}^{F} \frac{\lambda_i^2}{2\sigma_i^2} + \text{const}(\Lambda) \\
&= \sum_{x,y} \tilde{p}(x,y) \sum_i \lambda_i f_i(x,y) - \\
&\quad \sum_{x,y} \tilde{p}(x,y) \log \sum_{y'} \exp(\sum_i \lambda_i f_i(x,y')) - \\
&\quad \sum_{i=1}^{F} \frac{\lambda_i^2}{2\sigma_i^2} + \text{const}(\Lambda) \\
\frac{\partial L'_X(\Lambda)}{\partial \lambda_i} &= \sum_{x,y} \tilde{p}(x,y) f_i(x,y) - \\
&\quad \sum_{x,y} \tilde{p}(x,y) \sum_{y'} \frac{\exp(\sum_i \lambda_i f_i(x,y'))}{Z_\Lambda(x)} f_i(x,y') - \\
&\quad \frac{\lambda_i}{\sigma_i^2} \\
&= \sum_{x,y} \tilde{p}(x,y) f_i(x,y) - \\
&\quad \sum_{x,y} \tilde{p}(x,y) \sum_{y'} q_\Lambda(y'|x) f_i(x,y') - \frac{\lambda_i}{\sigma_i^2} \\
&= \sum_{x,y} \tilde{p}(x,y) f_i(x,y) - \\
&\quad \sum_{x} \tilde{p}(x) \sum_{y'} q_\Lambda(y'|x) f_i(x,y') \sum_{y} \tilde{p}(y|x) - \frac{\lambda_i}{\sigma_i^2} \\
&= \sum_{x,y} \tilde{p}(x,y) f_i(x,y) - \\
&\quad \sum_{x,y} \tilde{p}(x) q_\Lambda(y|x) f_i(x,y) - \frac{\lambda_i}{\sigma_i^2}
\end{aligned}$$

Equation (14) follows simply from the last line.

The derivation of the modified improved iterative scaling update is identical to the original derivation except for the presence of extra terms for the prior. In each iteration, we try to find $\Delta = \{\delta_i\}$ that maximizes the increase in the objective function:

$$\begin{aligned}
L'_X(\Lambda+\Delta) - L'_X(\Lambda) &= \sum_{x,y} \tilde{p}(x,y) \sum_i \delta_i f_i(x,y) - \\
&\quad \sum_x \tilde{p}(x) \log \sum_y q_\Lambda(y|x) \exp(\sum_i \delta_i f_i(x,y)) - \\
&\quad \frac{1}{2\sigma_i^2} \sum_i (2\lambda_i \delta_i + \delta_i^2) \quad .
\end{aligned}$$

As it is not clear how to maximize this function directly, we find an auxiliary function $A(\Delta)$ that we *can* maximize that bounds this function from below.

Using the inequality $\log x \leq x - 1$, we get

$$\begin{aligned}
L'_X(\Lambda+\Delta) - L'_X(\Lambda) &\geq \sum_{x,y} \tilde{p}(x,y) \sum_i \delta_i f_i(x,y) + \\
&\quad 1 - \sum_x \tilde{p}(x) \sum_y q_\Lambda(y|x) \exp(\sum_i \delta_i f_i(x,y)) - \\
&\quad \frac{1}{2\sigma_i^2} \sum_i (2\lambda_i \delta_i + \delta_i^2) = B(\Delta) \quad .
\end{aligned}$$

Substituting in $f^\#(x,y) = \sum_i f_i(x,y)$ and applying Jensen's inequality, we arrive at

$$\begin{aligned}
B(\Delta) &\geq \sum_{x,y} \tilde{p}(x,y) \sum_i \delta_i f_i(x,y) + 1 - \\
&\quad \sum_x \tilde{p}(x) \sum_y q_\Lambda(y|x) \sum_i \frac{f_i(x,y)}{f^\#(x,y)} \exp(\delta_i f^\#(x,y)) - \\
&\quad \frac{1}{2\sigma_i^2} \sum_i (2\lambda_i \delta_i + \delta_i^2) = A(\Delta) \quad .
\end{aligned}$$

Taking the partial derivative of $A(\Delta)$ with respect to δ_i, we get

$$\begin{aligned}
\frac{\partial A(\Delta)}{\partial \delta_i} &= \sum_{x,y} \tilde{p}(x,y) f_i(x,y) - \\
&\quad \sum_x \tilde{p}(x) \sum_y q_\Lambda(y|x) f_i(x,y) \exp(\delta_i f^\#(x,y)) - \frac{\lambda_i + \delta_i}{\sigma_i^2} \quad .
\end{aligned}$$

Equation (13) follows by setting these derivatives to zero.

[10] With n-gram models, we found that a single variance σ for the whole model worked quite well, though using separate σ_m for each level of the n-gram model worked slightly better. However, this partitioning is not applicable in general.

Acknowledgements

The authors would like to thank John Lafferty and Joshua Goodman for many helpful discussions and the anonymous reviewers for their excellent comments. We would also like to thank Scott Axelrod for pointing out to us the equivalence of fuzzy maximum entropy and the Gaussian prior.

References

[1] S. Della Pietra and V. Della Pietra, "Statistical modeling by maximum entropy," unpublished report, 1993.

[2] A. Berger, S. Della Pietra, and V. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[3] 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, Apr. 1997.

[4] A. Ratnaparkhi, *Maximum Entropy Models for Natural Language Ambiguity Resolution*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, 1998.

[5] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Computer Speech and Language*, vol. 10, pp. 187–228, 1996.

[6] R. Lau, "Adaptive statistical language modelling," M.S. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1994.

[7] R. Rosenfeld, *Adaptive Statistical Language Modeling: a Maximum Entropy Approach*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, Apr. 1994.

[8] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Tech. Rep. TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, 1998.

[9] E. Jaynes, "Information theory and statistical mechanics," *Physics Reviews*, vol. 106, pp. 620–630, 1957.

[10] J. Darroch and D. Ratcliff, "Generalized iterative scaling for log-linear models," *The Annals of Mathematical Statistics*, vol. 43, pp. 1470–1480, 1972.

[11] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 179–190, Mar. 1983.

[12] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Computational Linguistics*, vol. 16, no. 2, pp. 79–85, Jun. 1990.

[13] M. Kernighan, K. Church, and W. Gale, "A spelling correction program based on a noisy channel model," in *Proc. of the Thirteenth International Conf. on Computational Linguistics*, pp. 205–210, Aug. 1990.

[14] F. Jelinek and R. L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," in *Proc. of the Workshop on Pattern Recognition in Practice*, pp. 381–397, May 1980.

[15] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer, "An estimate of an upper bound for the entropy of English," *Computational Linguistics*, vol. 18, no. 1, pp. 31–40, Mar. 1992.

[16] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*, Prentice Hall, Englewood Cliffs, N.J., 1990.

[17] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 35, no. 3, pp. 400–401, Mar. 1987.

[18] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proc. of the IEEE International Conf. on Acoustics, Speech and Signal Processing*, vol. I, pp. 181–184, May 1995.

[19] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modeling," *Computer Speech and Language*, vol. 8, pp. 1–38, 1994.

[20] I. Good, "The population frequencies of species and the estimation of population parameters," *Biometrika*, vol. 40, no. 3 and 4, pp. 237–264, 1953.

[21] K. W. Church and W. A. Gale, "A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams," *Computer Speech and Language*, vol. 5, pp. 19–54, 1991.

[22] S. F. Chen, D. Beeferman, and R. Rosenfeld, "Evaluation metrics for language models," in *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, pp. 275–280, 1998.

[23] P. Brown, S. Della Pietra, V. Della Pietra, R. Mercer, A. Nadas, and S. Roukos, "A maximum penalized entropy construction of conditional log-linear language and translation models using learned features and a generalized csiszar algorithm," Internal IBM Report, 1992.

[24] R. Lau, R. Rosenfeld, and S. Roukos, "Adaptive language modeling using the maximum entropy principle," in *Proc. of the ARPA Workshop on Human Language Technology*, pp. 108–113, 1993.

[25] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proc. of the DARPA Speech and Natural Language Workshop*, pp. 357–362, Feb. 1992.

[26] S. F. Chen and R. Rosenfeld, "A Gaussian prior for smoothing maximum entropy models," Tech. Rep. CMU-CS-99-108, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1999.

[27] J. Lafferty, personal communication, 1997.

[28] W. Newman, "Extension to the maximum entropy method," *IEEE Trans. on Information Theory*, vol. 23, no. 1, pp. 89–93, Jan. 1977.

[29] S. Khudanpur, "A method of maximum entropy estimation with relaxed constraints," in *1995 Johns Hopkins University Language Modeling Workshop Proc.*, pp. 1–17, 1995.

[30] D. J. C. MacKay and L. C. Peto, "A hierarchical Dirichlet language model," *Natural Language Engineering*, vol. 1, no. 3, pp. 1–19, 1995.

[31] A. Nádas, "Estimation of probabilities in the language model of the IBM speech recognition system," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 32, no. 4, pp. 859–861, Aug. 1984.

[32] J. Lafferty and B. Suhm, "Cluster expansions and iterative scaling for maximum entropy language models," in *Maximum Entropy and Bayesian Methods*, K. Hanson and R. Silver, Eds., pp. 195–202. Kluwer Academic Publishers, 1995.

[33] H. Kucera and W. Francis, *Computational Analysis of Present-Day American English*, Brown University Press, Providence R.I., 1967.

[34] R. M. Stern, "Specification of the 1995 ARPA Hub 3 evaluation: Unlimited vocabulary NAB news baseline," in *Proc. of the DARPA Speech Recognition Workshop*, pp. 5–7, Feb. 1996.

[35] A. Rudnicky, "Hub 4: Business Broadcast News," in *Proc. of the DARPA Speech Recognition Workshop*, pp. 8–11, Feb. 1996.

[36] J. Godfrey, E. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. of the IEEE International Conf. on Acoustics, Speech and Signal Processing*, vol. I, pp. 517–520, Mar. 1992.

[37] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1988.

[38] M. Siegler, A. Berger, M. Witbrock, and A. Hauptmann, "Experiments in spoken document retrieval at CMU," in *Proc. of TREC-7: The Seventh Text Retrieval Conf.*, pp. 319–325, 1998.

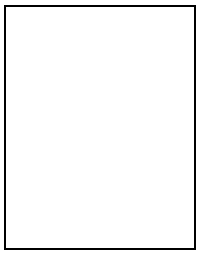
[39] K. Seymore, S. Chen, S. Doh, M. Eskenazi, E. Gouvea, B. Raj, M. Ravishankar, R. Rosenfeld, M. Siegler, R. Stern, and E. Thayer, "The 1997 CMU Sphinx-3 English Broadcast News transcription system," in *Proc. of the DARPA Speech Recognition Workshop*, pp. 55–59, 1998.

[40] S. Kullback, *Information Theory and Statistics*, Wiley, New York, 1959.

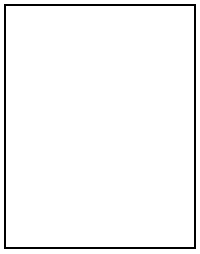
[41] D. Beeferman, A. Berger, and J. Lafferty, "A model of lexical attraction and repulsion," in *Proc. of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 373–380, 1997.

[42] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pp. 61–67, 1999.

[43] A. Berger and R. Miller, "Just-in-time language modeling," in *Proc. of the IEEE International Conf. on Acoustics, Speech and Signal Processing*, vol. II, pp. 705–708, 1998.



Stanley F. Chen Stanley F. Chen is currently a researcher in speech recognition at the IBM T.J. Watson Research Center. He received a B.S. in Mathematics from Caltech in 1989 and an M.S. and Ph.D. from Harvard University in Computer Science in 1995 and 1996, respectively. Before joining IBM, he spent three years as a postdoctoral fellow at Carnegie Mellon University. His research interests include language modeling, maximum entropy modeling, machine translation, and statistical natural language processing.



Ronald Rosenfeld Ronald Rosenfeld (www.cs.cmu.edu/~roni) is an Associate Professor in the School of Computer Science and the Graduate School of Industrial Administration at Carnegie Mellon University, Pittsburgh, Pennsylvania. He received a B.Sc. in Mathematics and Physics from Tel-Aviv University (1985) and a M.Sc. (1991) and Ph.D. (1994) in Computer Science from Carnegie Mellon University. He is a National Science Foundation Graduate Fellow (1986-1990) and a recipient of the Allen Newell Medal for Research Excellence (1992). His research interests include statistical language modeling, human language technologies, speech recognition, and human-machine speech communication.