Latent Maximum Entropy Approach for Semantic N-gram Language Modeling

Shaojun Wang Dale Schuurmans Fuchun Peng
School of Computer Science, University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
{sjwang, dale, f3peng}@cs.uwaterloo.ca

Abstract

In this paper, we describe a unified probabilistic framework for statistical language modeling—the latent maximum entropy principle—which can effectively incorporate various aspects of natural language, such as local word interaction, syntactic structure and semantic document information. Unlike previous work on maximum entropy methods for language modeling, which only allow explicit features to be modeled, our framework also allows relationships over hidden features to be captured, resulting in a more expressive language model. We describe efficient algorithms for marginalization, inference and normalization in our extended models. We then present promising experimental results for our approach on the Wall Street Journal corpus.

1 Introduction

Statistical language modeling is concerned with determining the probability of naturally occurring word sequences in a language. Traditionally, the dominant motivation for language modeling has come from speech recognition, however statistical language models have recently become more widely used in many other application areas, such as information retrieval, machine translation, optical character recognition, spelling correction, document classification, information extraction, and bio-informatics.

The goal of language modeling is to predict the probability of natural word sequences, or more simply, to put high probability on word sequences that actually occur (and low probability on word sequences that never occur). Given a word sequence $w_1 w_2 ... w_N$ to be used as a test corpus, the quality of a language model can be measured by the empirical perplexity and entropy scores on this corpus [1]

$$\begin{aligned} Perplexity &= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{Pr(w_i|w_1...w_{i-1})}} \\ Entropy &= \log_2 Perplexity \end{aligned}$$

The goal is to obtain small values of these measures.

There are various kinds of language models that can be used to capture different aspects of regularities of natural language. The simplest and most successful language models are (1) the Markov chain (n-gram) source models [16], which are efficient at encoding local lexical regularities; (2) the structural language model [5], which effectively exploits relevant syntactic regularities; and (3) the semantic language model [2, 12], which can exploit document-level semantic regularities. However each of these language models only aims at some specific linguistic phenomena. None of them can simultaneously take into account the lexical information inherent in Markov chain models, the hierarchical syntactic tree structure in stochastic branching processes, and the semantic content in bag-of-words categorical mixture log-linear models—all in a unified probabilistic framework.

Several techniques for combining language models have been investigated. The most commonly used method is simple linear interpolation [5, 11, 15], where each individual model is trained separately and then combined by a weighted linear combination, where the weights are trained using held out data. Even though this technique is simple and easy to implement, it does not generally yield effective combinations because the linear additive form is too blunt to capture subtleties in each of the component models [15]. Another approach is based on Jaynes' maximum entropy (ME) principle [13]. This approach has several advantages over other methods for statistical modeling, such as introducing less data fragmentation (as in decision tree learning), requiring fewer independence assumptions (as in naive Bayes models), and exploiting a principled technique for automatic feature weighting. The major weakness with maximum entropy methods, however, are that they can only model distributions over explicitly observed features, whereas in natural language we encounter hidden semantic [2, 12] and syntactic information [5] that we do not observe directly.

One way to encode constraints over hidden features in a maximum entropy model is to first pre-process the training corpus to obtain explicit values for all of the hidden features—such as recovering syntactic structure by running a parser, or recovering semantic content by using a latent semantic indexer—and then incorporating statistics over explicitly measured features as additional constraints in the model [3, 14, 15]. However, doing so explicitly is not always possible, and even if attempted, sparse data problems almost always immediately arise in such complex models. Consequently, the perplexity improvements or word error rate reductions obtained are often minimal. In this paper we address the question: is it possible to exploit the hidden hierarchical structure of natural language in a maximum entropy method without resorting to explicit preliminary parsing or semantic analysis?

Recently we proposed a latent maximum entropy (LME) principle [17, 18] which extends Jaynes' maximum entropy principle to incorporate latent variables. In this paper, we show how our new principle can be used for statistical language modeling by training mixtures of exponential families with rich expressive power. We summarize the LME principle, its problem formulation, solution and certain convergence properties. Then we discuss how to use LME for language modeling. By properly using factorization methods and exploiting the sparseness of tri-gram features, we can demonstrate efficient algorithms for feature expectation, inference and normalization. Finally, we apply this model to the Wall Street Journal data to obtain experimental results which support the utility of our approach.

2 Latent Maximum Entropy (LME)

To express a joint probability model, let $X \in \mathcal{X}$ denote the complete data, $Y \in \mathcal{Y}$ be the observed incomplete data and $Z \in \mathcal{Z}$ be the missing data. That is, $X = (Y, Z)$. For example, Y might be observed natural language in the form of text, and X might be the text along with its missing syntactic and semantic information Z. The goal of maximum entropy is to find a probability model that matches certain constraints in the observed data while otherwise maximizing entropy. When the data has both missing and observed components we extend the maximum entropy principle to the latent maximum entropy principle as follows.

Latent maximum entropy principle Given features $f_1, ..., f_N$ specifying the properties we would like to match in the data, select a joint model p_* from the set of possible probability distributions that maximizes the entropy

$$\max_p \quad H(p) \;=\; -\sum_x p(x) \log p(x)$$

subject to

$$\sum_x p(x) f_i(x) \;=\; \sum_y \tilde{p}(y) \sum_z p(z|Y=y)\ f_i(y,z); \qquad i = 1..N$$

Here $\tilde{p}(y)$ is the empirical distribution of the set of observed components of the training data, and $p(z|Y = y)$ encodes the hidden dependency structure into the statistical model.

The LME principle is strictly more general than the ME principle, and only becomes equivalent to ME in the special case when the features only depend on the observable data Y. However, if the features depend on unobserved components of the data Z then ME only models the observed part of the data, and LME differs from ME [18].

Below we will apply the LME principle to the problem of combining language models. However, we first consider a small improvement that will prove useful. In many statistical modeling situations, the constraints used in the maximum entropy principle are subject to errors due to the empirical data, especially in a very sparse domain. One way to gain robustness to these errors is to relax the constraints but add a penalty to the entropy of the joint model [6, 7].

Regularized LME principle

$$\max_p \quad H(p) - U(a) \;=\; -\sum_x p(x) \log p(x) - U(a) \quad (1)$$

subject to (for $i = 1...N$)

$$\sum_x p(x) f_i(x) = \sum_y \tilde{p}(y) \sum_z p(z|Y=y)\ f_i(y,z) + a_i \quad (2)$$

Here $a = (a_1, ..., a_N)$ and a_i is the error for each constraint, and $U : \Re^N \to R$ is a smoothing convex function [6, 7] which has minimum at 0. The regularization term U penalizes deviations in more reliably observed constraints to a greater degree than deviations in less reliably observed constraints.

3 A Training Algorithm

We are now left with the problem of solving the constrained optimization problem posed in (1) and (2). Note that due to the nonlinear mapping introduced by $p(z|Y=y)$ we have nonlinear constraints (2) on the objective and the feasible set is no longer convex. So even though the objective function (1) is concave, no unique optimal solution can be expected. In fact, minima and saddle points may exist.

To make progress, we first restrict $p(x)$ to be an exponential model, $p_\lambda(x) = \Phi_\lambda^{-1} \exp\left(\sum_i \lambda_i f_i(x)\right)$, where Φ_λ is a constant that ensures $\sum_x p_\lambda(x) = 1$. This assumption makes it possible to formulate an iterative algorithm for finding feasible solutions (below). Our algorithmic strategy then is to generate many feasible candidates (by restarting the iterative procedure at different initial points), evaluate their entropy and select the best model. The hardest part of this process is generating feasible solutions.

The key observation to finding feasible solutions is to note that the stationary points of the penalized log-likelihood of the observed data, $R(\lambda, \sigma) = \sum_y \tilde{p}(y) \log p_\lambda(y) + U^*(\lambda)$, are among the feasible set of the relaxed constraints; where $U^*(\lambda)$ is the convex conjugate of U.[1] That is, to find feasible solutions it suffices to find models that maximize the penalized log-likelihood on observed data using standard iterative approaches. We use an iterative procedure, EM-IS, which employs an EM algorithm [10] as an outer loop, but uses a nested GIS/IIS algorithm [3, 8, 9] to perform the internal M step. Assuming the Gaussian prior, we obtain

EM-IS algorithm

E step:

Compute $\sum_y \tilde{p}(y) \sum_z p_{\lambda^{(j)}}(z|Y=y) f_i(y,z)$, $i = 1..N$

M step:

Perform K parallel updates of the parameter values $\lambda_i, i = 1...N$ by iterative scaling (GIS or IIS) as follows

$$\lambda_i^{(j+s/K)} = \lambda_i^{(j+(s-1)/K)} + \gamma_i^{(j+s/K)}; \; s = 1...K \qquad (3)$$

[1]Note that for a quadratic penalty $U(a) = \sum_{i=1}^N \frac{1}{2}\sigma_i^2 a_i^2$ with $a_i = \frac{\lambda_i}{\sigma_i^2}$ we obtain $U^*(\lambda) = \sum_{i=1}^N \frac{\lambda_i^2}{2\sigma_i^2}$, the Gaussian prior.

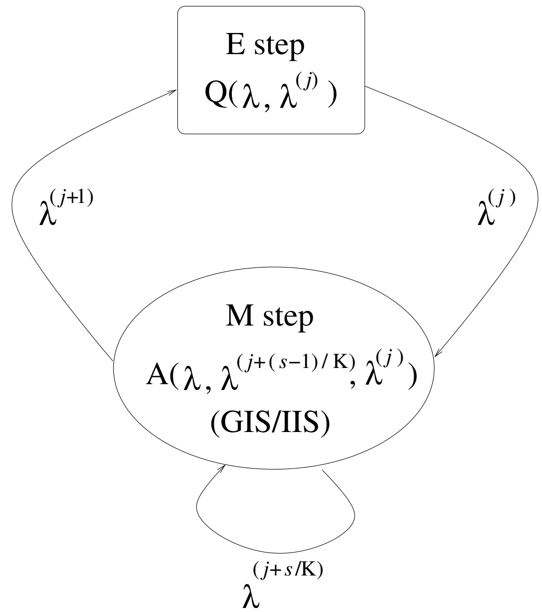

Figure 1: LME, an EM procedure embedding an iterative scaling loop, where $A(\lambda^{(j+s/K)}, \lambda^{(j+(s-1)/K)}, \lambda^{(j)})$ is the auxiliary function in IIS, s denotes the index of one cycle of full parallel update of $\lambda_i, i = 1, \cdots, N$ and K denotes the number of cycles of full parallel updates.

where $\delta_i^{(j+s/K)}$ satisfies

$$\sum_x p_{\lambda^{(j+(s-1)/K)}}(x) f_i(x) e^{\gamma_i^{(j+s/K)} f(x)} + \frac{\lambda_i^{(j+(s-1)/K)} + \gamma_i^{(j+s/K)}}{\sigma_i^2}$$

$$= \sum_y \tilde{p}(y) \sum_z p_{\lambda^{(j)}}(x|Y=y) f_i(y,z) \qquad (4)$$

where $f(x) = \sum_{i=1}^N f_i(x)$. The value of $\delta_i^{(j+s/K)}$ can be obtained by bisection line search or solving the nonlinear equation (4) by Newton-Raphson iteration.

The embedded EM-IS training algorithm is illustrated in Figure 1. A natural interpretation of this iterative procedure is that, if the right hand side of (2) is constant, then the optimal solution $p_\lambda(x)$ is a log-linear model with parameters provided by GIS/IIS. Once we obtain p_λ we can calculate the value of the right hand side of (2). If this value matches the value previously assigned, then by the optimality condition we have reached a stationary point of the log-likelihood and a feasible solution of the LME problem; otherwise, we iterate until the constraints are met.

Theorem 1 *The EM-IS algorithm monotonically increases the likelihood function $L(\lambda)$, and all limit points of any EM-IS sequence $\{\lambda^{(j+s/K)}, j \geq 0\}$, $s =$*

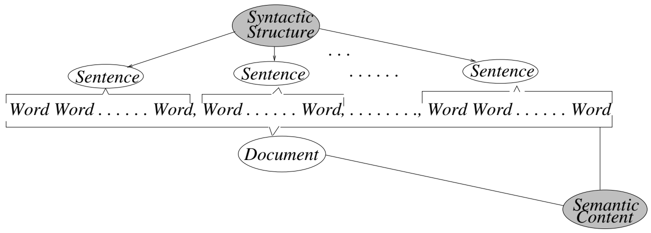

Figure 2: Natural language, observed incomplete data are words, sentences, documents, missing data are syntactic structure at sentence level, semantic content at document level, where dark nodes denote missing information.

1...K, belong to the set

$$\Gamma = \left\{ \lambda \in \Re^N : \frac{\partial R(\lambda)}{\partial \lambda} = 0 \right\} \quad (5)$$

Therefore, EM-IS asymptotically yields feasible solutions to the LME principle for log-linear models [18].

4 LME for Language Modeling

Natural language is a complex, hierarchically organized code used to represent messages. Simple low level patterns are combined in a well-defined manner to form more complex patterns at successively higher levels. The function of the hierarchy is to constrain the ways in which the individual low level patterns can be combined, and build redundancy into the source code to make it robust to errors made by speakers. As a result, relatively few primitive patterns can be combined in a multilevel hierarchy to form a rich, robust information-bearing code. To illustrate, note that a sequence of words can be viewed as a concatenation of low level n-grams (Markov chain), while the generation of such sequences is also governed by a hidden hierarchical grammar (syntax) and by certain semantic components, such as hidden document topics (Figure 2). A good language model should therefore be able to consider all of these information sources simultaneously.

The latent maximum entropy principle can be used to combine different language modeling components in a principled way. In this section, we describe how to use the LME principle to combine the tri-gram Markov model with probabilistic latent semantic analysis (PLSA) [12] to obtain a richer language model.

Currently almost all maximum entropy language models use the conditional form first proposed by Brown et al. [4] for statistical machine translation. The main reason for using the conditional model is to avoid enumerating all possible histories to perform inference. Here we use the joint probability model, but point out that once the set of features are selected, the problem of calculating the needed feature expectations and normalization terms becomes tractable by using proper factorization methods and exploiting the sparseness of tri-grams.

4.1 Combining N-gram and PLSA Models

Define the complete data as $x = (W_2, W_1, W_0, D, T_2, T_1, T_0)$, where W_0, W_1, W_2 are the current and two previous words, T_2, T_1, T_0 are the hidden 'topic' values associated with these words, D is a document identifier, and $y = (W_2, W_1, W_0, D)$ is the observed data. Typically the number of documents, words in the vocabulary, and latent class variables are on the order of 100,000, 10,000 and 100, respectively. A graphical representation of a semantic node interacting with a tri-gram is illustrated in Figure 3.

For the tri-gram portion of the model, all features are explicitly observed in the training data, and the corresponding constraints can be modeled directly as follows.

$$\begin{aligned}
\sum_x p(x)\delta(W_2 = w_i, W_1 = w_j, W_0 = w_k) &= \sum_d \tilde{p}(d)\tilde{p}(w_i w_j w_k | d) \\
\sum_x p(x) \sum_{\ell=0}^{1} \delta(W_{\ell+1} = w_i, W_\ell = w_j) &= \sum_d \tilde{p}(d)\tilde{p}(w_i w_j | d) \\
\sum_x p(x) \sum_{\ell=0}^{2} \delta(W_\ell = w_i) &= \sum_d \tilde{p}(d)\tilde{p}(w_i | d) \quad (6)
\end{aligned}$$

These specify the tri-gram, bi-gram and uni-gram constraints the model should respect, respectively.

For the semantic (PLSA) portion of the model, the constraints involve the hidden topic variables T and can be encoded by the more complex constraints

$$\sum_x p(x) \sum_{\ell=0}^{2} \delta(T_\ell = t, D = d) = \tilde{p}(d) \sum_{\ell=0}^{2} \tilde{p}(W_\ell | d) p(t | W_\ell, d) \quad (7)$$

$$\begin{aligned}
&\sum_x p(x) \sum_{\ell=0}^{2} \delta(T_\ell = t, W_\ell = w_i) \\
&\quad = \sum_d \tilde{p}(d)\tilde{p}(w_i | d) \sum_{\ell=0}^{2} p(t | W_\ell = w_i, d) \quad (8)
\end{aligned}$$

where $\delta(.)$ is 1 if the event is active and zero otherwise. The first equality (7) imposes the constraints between the document node and the topic node, and the second

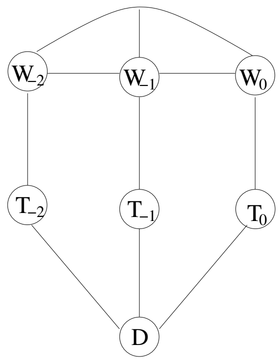

Figure 3: A graphical representation of the semantic tri-gram model, where the curve that connects the three word nodes together denotes the tri-gram feature. In this graphical representation, many arcs share the same parameters.

equality (8) imposes the constraints between the topic node and words.

We can now learn a probability model that simultaneously takes the two information sources into account, by employing the LME principle to find the log-linear model $p_\lambda(x)$ that maximizes entropy subject to satisfying all of the constraints. This model will encapsulate the n-gram and semantic models as special cases. Figure 3 gives a graphical representation of the structure resulting from satisfying all of the imposed constraints. Note that many of the components share the same parameters; namely, (T_2, D), (T_1, D), and (T_0, D) are identical; (T_2, W_2), (T_1, W_1), and (T_0, W_0) are identical; (W_2, W_1) and (W_1, W_0) are identical; and (W_2), (W_1) and (W_0) are identical.

4.2 Efficient Feature Expectation and Inference

The computational bottleneck is calculating the feature expectations and normalization constants needed to perform inference. Note that the full joint distribution is in the form of a product over exponential functions of features. The key idea for efficient calculation is to "push" the sums in as far as possible when summing (marginalizing) out irrelevant terms. Since calculating feature expectations has the same computational cost as normalization [14], we only show how to do normalization efficiently here. The normalization factor can be calculated efficiently by sum-product algorithm, that is, summing over all the links at each time slice and passing through the trellis nodes with the product of the weight to the ongoing nodes we

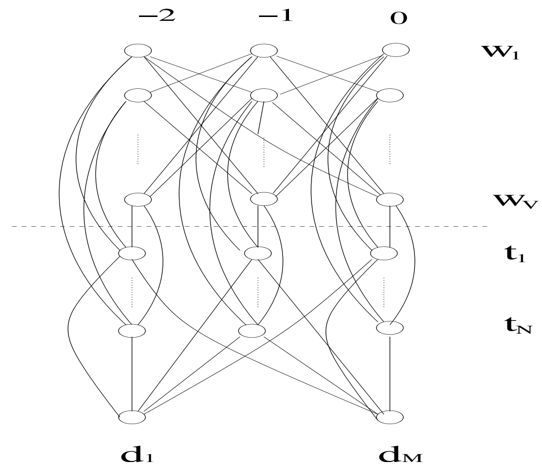

Figure 4: Trellis to calculate the normalization factor and feature expectation. The weight of each link is an exponentiated version of the corresponding Lagrange multiplier.

obtain

$$
\begin{aligned}
\Phi &= \sum_{w_2,w_1,w_0,t_2,t_1,t_0,d} \big(e^{\lambda_{w_2}} e^{\lambda_{w_1}} e^{\lambda_{w_0}} e^{\lambda_{w_2 w_1}} \\
&\qquad e^{\lambda_{w_1 w_0}} e^{\lambda_{w_2 w_1 w_0}} e^{\lambda_{w_2 t_2}} \\
&\qquad e^{\lambda_{w_1 t_1}} e^{\lambda_{w_0 t_0}} e^{\lambda_{t_2 d}} e^{\lambda_{t_1 d}} e^{\lambda_{t_0 d}} \big) \\
&= \sum_{w_0} e^{\lambda_{w_0}} \sum_{t_0} e^{\lambda_{w_0 t_0}} \\
&\quad \sum_{w_1} e^{\lambda_{w_1}} e^{\lambda_{w_1 w_o}} \sum_{t_1} e^{\lambda_{w_1 t_1}} \\
&\quad \sum_{w_2} e^{\lambda_{w_2}} e^{\lambda_{w_1 w_2}} e^{\lambda_{w_2 w_1 w_0}} \\
&\quad \sum_{t_2} e^{\lambda_{w_2 t_2}} \Big(\sum_{d} e^{\lambda_{t_2 d}} e^{\lambda_{t_1 d}} e^{\lambda_{t_0 d}} \Big) \qquad (9)
\end{aligned}
$$

Simultaneously to obtaining the normalization constant, we can also calculate all of the feature expectations. For example, the expectation of a given tri-gram feature $w_i w_j w_k$ can be calculated as

$$
\begin{aligned}
&\sum_x p(x) \delta(W_2 = w_i, W_1 = w_j, W_0 = w_k) \\
&= \Phi^{-1} e^{\lambda_{w_i}} e^{\lambda_{w_j}} e^{\lambda_{w_k}} e^{\lambda_{w_i w_j}} e^{\lambda_{w_j w_k}} e^{\lambda_{w_i w_j w_k}} \\
&\quad \sum_{t_0} e^{\lambda_{w_k t_0}} \sum_{t_1} e^{\lambda_{w_j t_1}} \sum_{t_2} e^{\lambda_{w_i t_2}} \\
&\quad \Big(\sum_d e^{\lambda_{t_2 d}} e^{\lambda_{t_1 d}} e^{\lambda_{t_0 d}} \Big) \qquad (10)
\end{aligned}
$$

Figure 4 illustrates a normalization procedure that incorporates LSA with a 3-gram model. The nodes above the bar are the word vocabulary

$\{W_1, W_2, ...W_V\}$ and the nodes below the bar are the the possible topic set $\{t_1, ...t_N\}$. The documents $\{d_1, ...d_M\}$ are connected to topic nodes and are represented horizontally below the bar. Links represent active features, and the horizontal axis above the bar is the time axis.

4.3 Computation in Testing

To evaluate the perplexity of our semantic tri-gram model on the observable portion of the test data, note that

$$\begin{aligned} & p(w_L ... w_1) \\ = & \prod_{\ell=1}^{L} p(w_\ell | w_L ... w_{\ell+1}) \\ = & \prod_{\ell=1}^{L} \sum_{D, T_2, T_1, T_0} p(w_\ell, D, T_2, T_1, T_0 | w_L ... w_{\ell+1}) \\ = & \prod_{\ell=1}^{L} \sum_{D, T_2, T_1, T_0} p(w_\ell, D, T_2, T_1, T_0 | w_{\ell+2}, w_{\ell+1}) \end{aligned}$$

Since our model provides the probability of complete data $p(W_2, W_1, W_0, D, T_2, T_1, T_0)$, the conditional probability $p(W_0, D, T_2, T_1, T_0 | W_2, W_1)$ can be easily obtained by marginalization (and division).

5 Experimental Results and Discussions

The corpus used to train our model was taken from the WSJ portion of the NAB corpus and was composed of about 87,000 documents spanning the years 1987 to 1989, comprising approximately 38 millions words. The vocabulary was constructed by taking the 20,000 most frequent words of the training data. Another separate set of data consisting of 325,000 words was taken from the year 1989 and used for testing.

We perform EM-IS to train our models where we set the internal IIS loop iterations to be 20, and the outer EM loop iterations to be 5.

We chose $|T| = 125$ as number of possible topics. The baseline tri-gram model with Good-Turing back-off smoothing has perplexity of 105. In our model, we fixed the variance of the Gaussian prior σ_i to be 1. When only the tri-gram constraints are considered, we obtain a perplexity of 107. After the PLSA constraints are added, the perplexity is reduced to 91; comprising a 13.3% reduction in perplexity from the baseline tri-gram model.

In [2], Bellegarda built a language model that combined a tri-gram model and an LSA model using an *ad hoc* approach. The formula he used to calculate the perplexity was

$$\begin{aligned} & p(w_\ell | w_L ... w_{\ell+1}) \\ = & \frac{p(w_\ell | w_{\ell+2} w_{\ell+1}) p_{LSA}(d_\ell | w_\ell)}{\sum_{w_i} p(w_i | w_{\ell+2} w_{\ell+1}) p_{LSA}(d_\ell | w_\ell)} \end{aligned} \tag{11}$$

where $p_{LSA}(d_\ell | w_\ell)$ is the probability of current document history given current word w_ℓ, obtained by the latent semantic analysis. We calculated the perplexity of his model using the same training data and test data considered above. The perplexity obtained by Bellegarda's model is 97, which is only an 8% reduction in perplexity compared to the baseline tri-gram model above. However, if we intentionally emphasize the LSA portion of Bellegarda's model by taking its 7th power, and renormalizing

$$\begin{aligned} & p(w_\ell | w_L ... w_{\ell-1}) \\ = & \frac{p(w_\ell | w_{\ell+2} w_{\ell+1}) (p_{LSA}(d_\ell | w_\ell))^7}{\sum_{w_i} p(w_i | w_{\ell+2} w_{\ell+1}) (p_{LSA}(d_\ell | w_\ell))^7} \end{aligned} \tag{12}$$

we obtain a drastic perplexity reduction. The perplexity achieved in this case is reduced to 82, which is a remarkable reduction (21% compared to the baseline tri-gram model).

We are investigating principles for adopting an analogous technique in our LME approach.

6 Conclusion

We have presented a latent maximum entropy principle for statistical language modeling. Our LME method provides a general statistical framework for incorporating arbitrary aspects of natural language into a parametric model. The parameters can be estimated by combining standard iterative procedures, interactions among various aspects of language can be taken into account automatically and simultaneously, and the general model is reduced to a familiar model when aiming at a specific linguistic phenomenon. We can demonstrate efficient algorithms for feature expectation, normalization and inference.

We believe that our preliminary results on the WSJ corpus are very promising because we have not significantly tuned the parameters. We are investigating techniques for finding the optimal number of clusters to use in smoothing. Also, we are currently only combining an n-gram model with document semantic information, and we are now investigating how to efficiently add syntactic information (such as context free grammatical structure) to this framework and expect to obtain further improvement.

7 Acknowledgments

Research supported by Bell University Labs, MITACS and NSERC.

References

[1] L. Bahl, F. Jelinek, and R. Mercer, (1983); A maximum likelihood approach to continuous speech recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, No. 2, pp. 179–190.

[2] J. Bellegarda, (2000); Exploiting Latent Semantic Information in Statistical Language Modeling, *Proceedings of IEEE*, Vol. 88, No. 8, pp. 1279-1296, August

[3] A. Berger, S. Della Pietra and V. Della Pietra, (1996); A Maximum Entropy Approach to Natural Language Processing, *Computational Linguistics*, Vol. 22, No. 1, pp. 39-71

[4] P. Brown, S. Della Pietra, V. Della Pietra, R. Mercer, A. Nadas, and S. Roukos, (1992); A Maximum Entropy Construction of Conditional Log-linear Language and Translation Models Using Learned Features and a Generalized Csiszar Algorithm, *IBM report*

[5] C. Chelba and F. Jelinek, (2000); Structured Language Modeling, *Computer Speech and Language*, Vol. 14, No. 4, pp. 283-332, October

[6] S. Chen and R. Rosenfeld, (2000); A Survey of Smoothing Techniques for ME Models, *IEEE Trans. on Speech and Audio Processing*, Vol. 8, No. 1, pp. 37–50

[7] I. Csiszar, "Maxent, Mathematics, and Information Theory," *Maximum Entropy and Bayesian Methods*, Edited by K. Hanson and R. Silver, pp. 35-50, Kluwer Academic Publishers, 1996

[8] S. Della Pietra, V. Della Pietra and J. Lafferty, (1997); Inducing Features of Random Fields, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 4, pp. 380-393

[9] J. Darroch and D. Ratchliff, (1972); Generalized Iterative Scaling for Log-Linear Models, *The Annals of Mathematical Statistics*, Vol. 43, No. 5, pp. 1470-1480

[10] A. Dempster, N. Laird and D. Rubin, (1977); Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm, *Journal of Royal Statistical Society, Series B*, Vol. 39, pp 1-38

[11] C. Genestand J. Zidek, (1986); Combining Probability Distributions: A Critique and an Annotated Bibliography, *Statistical Science*, Vol. 1, No. 1, February, pp. 114-135

[12] T. Hofmann, (2001); Unsupervised Learning by Probabilistic Latent Semantic Analysis, *Machine Learning*, Vol. 42, No. 1, pp.177-196

[13] E. Jaynes, (1983); *Papers on Probability, Statistics, and Statistical Physics*, edited by R. Rosenkrantz, D. Reidel Publishing Company

[14] S. Khudanpur and J. Wu, (2000); Maximum Entropy Techniques for Exploiting Syntactic, Semantic and Collocational Dependencies in Language Modeling, *Computer Speech and Language*, Vol. 14, No. 4, pp. 355-372

[15] R. Rosenfeld, (1996); A Maximum Entropy Approach to Adaptive Statistical Language Modeling, *Computer Speech and Language*, Vol. 10, pp. 187-228

[16] C. Shannon, (1948); A Mathematical Theory of Communication, *Bell System Technical Journal*, Vol. 27, pp. 398-403

[17] S. Wang, R. Rosenfeld, Y. Zhao and D. Schuurmans, (2002); The Latent Maximum Entropy Principle, *IEEE International Symposium on Information Theory*

[18] S. Wang, D. Schuurmans and Y. Zhao (2002); The Latent Maximum Entropy Principle, submitted to *IEEE Trans. on Information Theory*