
Machine Learning, 60, 229–250, 2005

Combining Statistical Language Models via the Latent Maximum Entropy Principle

SHAOJUN WANG swang@cs.ualberta.ca
DALE SCHUURMANS dale@cs.ualberta.ca
Department of Computing Science, University of Alberta, Canada

FUCHUN PENG fuchun@cs.umass.edu
Department of Computer Science, University of Massachusetts at Amherst, USA

YUNXIN ZHAO zhaoy@missouri.edu
Department of Computer Engineering and Computer Science, University of Missouri at Columbia, USA

Editors: Dan Roth and Pascale Fung

Abstract. We present a unified probabilistic framework for statistical language modeling which can simultaneously incorporate various aspects of natural language, such as local word interaction, syntactic structure and semantic document information. Our approach is based on a recent statistical inference principle we have proposed—the latent maximum entropy principle—which allows relationships over hidden features to be effectively captured in a unified model. Our work extends previous research on maximum entropy methods for language modeling, which only allow observed features to be modeled. The ability to conveniently incorporate hidden variables allows us to extend the expressiveness of language models while alleviating the necessity of pre-processing the data to obtain explicitly observed features. We describe efficient algorithms for marginalization, inference and normalization in our extended models. We then use these techniques to combine two standard forms of language models: local lexical models (Markov N-gram models) and global document-level semantic models (probabilistic latent semantic analysis). Our experimental results on the Wall Street Journal corpus show that we obtain a 18.5% reduction in perplexity compared to the baseline tri-gram model with Good-Turing smoothing.

Keywords: language modeling, N-gram models, latent semantic analysis, maximum entropy, latent variables

1. Introduction

In the past decade, a vast amount of information has become available on the World Wide Web, much of it in the form of natural language text. As the amount and significance of this data source grows, it will become increasingly important to find ways to effectively exploit it to help us improve our understanding of the world. Thus problems of information extraction and knowledge discovery from massive textual data sets are destined to remain important. In our view, statistical language models that also model hidden features provide a promising approach to robustly extracting information from natural language text.

Statistical language modeling is concerned with determining the probability of naturally occurring word sequences in human natural language. Traditionally, the dominant motivation for language modeling has come from the field of speech recognition (Jelinek, 1998), however statistical language models have recently become more widely used in

many other application areas, such as information retrieval (Lafferty & Zhai, 2001), machine translation (Brown et al., 1992), optical character recognition, spelling correction, document classification (Peng, Schuurmans,& Wang, 2004), and bio-informatics (Durbin et al., 1998).

There are various kinds of language models that can be used to capture different aspects of natural language regularity. The simplest and most successful language models are the Markov chain (N-gram) source models, first explored by Shannon in his seminal paper (Shannon, 1948). Such N-gram models effectively capture local lexical regularities in text. Subsequently, a wide variety of smoothing methods have been developed to address the problem of estimating rare events for these models (Chen & Goodman, 1999). The resulting smoothed N-gram language models have become a key component of state of the art speech recognizers, by helping to resolve acoustic ambiguities by placing higher probability on more likely word strings.

While Markov chains are efficient at encoding local word interactions, natural language clearly has a richer structure than can be conveniently captured by an N-gram model. For example, attempting to increase the order of an N-gram to capture longer range dependencies in natural language immediately runs into the curse of dimensionality (Bengio et al., 2003; Jelinek, 1998; Rosenfeld, 2000).

Many recent approaches have been proposed to capture and exploit different aspects of natural language regularity with the goal of outperforming the simple N-gram model. For example, the structural language model (Chelba & Jelinek, 2000; Roark, 2001) effectively exploits relevant syntactic regularities to improve the perplexity score of N-gram models, and the semantic language model (Bellegarda, 2000; Bengio et al., 2003; Hofmann, 2001) exploits document-level semantic regularities to acheive similar improvements. Although each of these language models outperforms simple N-grams, they each only capture 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 thus also been investigated. The most commonly used method is simple linear interpolation (Chelba & Jelinek, 2000; Jelinek & Mercer, 1980; Roark, 2001; Rosenfeld, 1996), 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 (Rosenfeld, 1996). Another approach is based on Jaynes' *maximum entropy* (ME) principle (Jaynes, 1983). 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 (Bellegarda, 2000; Hofmann, 2001) and syntactic information (Chelba & Jelinek, 2000) which 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 (Berger, Della Pietra, & Della Pietra, 1996; Khudanpur & Wu, 2000; Rosenfeld, 1996). 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 within a maximum entropy method without resorting to explicit preliminary parsing or semantic analysis?

Recently we proposed a *latent maximum entropy* (LME) principle (Wang, Schuurmans, & Zhao, 2003) which extends Jaynes' maximum entropy principle to incorporate latent variables. It is different from both Jaynes' ME principle and maximum likelihood estimation, but often yields better estimates in the presence of hidden variables. Preliminary experiments in (Wang, Schuurmans, & Zhao, 2003) showed that estimation based on the latent maximum entropy principle generally yielded improved results over the maximum likelihood principle when estimating latent variable models on small observed data samples.

In this paper, we show how our new estimation principle can be used for statistical language modeling by training mixtures of exponential families with rich expressive power. Below, we first 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 for combining Markov N-gram models with probabilistic latent semantic analysis (Hofmann, 2001). We apply this model to Wall Street Journal data to obtain experimental results which support the utility of our approach. Finally, we discuss the advantages and limitations of our approach.

2. The latent maximum entropy (LME) principle

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 $f_1, \ldots, f_N$ 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

$$\sum_{x} p(x) f_i(x) \;=\; \sum_{y} \tilde{p}(y) \sum_{z} p(z \mid y)\; f_i(y, z); \quad i = 1, \ldots, N$$

Here $\tilde{p}(y)$ is the empirical distribution of the set of observed components of the training data, and $p(z \mid y)$ encodes the hidden dependency structure into the statistical model. Intuitively, the constraints specify that we require the expectations of $f_i(X)$ in the joint model to match their empirical expectations on the incomplete data Y, taking into account the structure of the implied dependence of the unobserved component Z on Y. Note that the conditional distribution $p(z \mid y)$ implicitly encodes the latent structure and is a nonlinear mapping of $p(x)$. That is, $p(z \mid y) = p(y, z)/\sum_{z' \in \mathcal{Z}} p(y, z') = p(x)/\sum_{z \in \mathcal{Z}} p(x')$ where $x = (y, z)$ and $x' = (y, z')$ by definition. Unfortunately, $p(z \mid y)$ is a nonlinear function of $p(x)$, which raises the main technical challenges we face below.

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 Y, and LME differs from ME (Wang, Schuurmans, & Zhao, 2003).

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 (Chen & Rosenfeld, 2000; Csiszar, 1996).

Regularized LME (RLME) principle

$$\max_{p,a} \quad H(p) - U(a) \;=\; -\sum_{x} p(x) \log p(x) - U(a) \tag{1}$$

subject to

$$\sum_{x} p(x) f_i(x) = \sum_{y} \tilde{p}(y) \sum_{z} p(z \mid y)\; f_i(y, z) + a_i; \quad i = 1, ..., N \tag{2}$$

Here $a = (a_1, ..., a_N)$, a_i is the error for each constraint, and $U : \Re^N \to \Re$ is a convex function (Chen & Rosenfeld, 2000; Csiszar, 1996) which has its minimum at 0. The function U penalizes errors in satisfying the constraints, and can be used to penalize deviations in the more reliably observed constraints to a greater degree than deviations in less reliably observed constraints.

3. A training algorithm

Assume for the time being that we have already selected the features $f_1, ..., f_N$ that we wish to capture in the training data (we return to this issue in Section 4 below). 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 $p(z \mid y)$ we have nonlinear constraints (2) on p 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, $p_\lambda(x) = \Phi_\lambda^{-1} \exp(\sum_i \lambda_i f_i(x))$, where Φ_λ is a constant that ensures $\sum_{x \in \mathcal{X}} p_\lambda(x) = 1$. This assumption makes it possible to formulate a practical iterative algorithm for finding feasible solutions (below) to approximately satisfying the RLME principle. Our algorithmic strategy then is to generate many feasible candidates (by restarting the iterative procedure at different initial points), evaluate their regularized 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 they are intimately related to finding locally *maximum a posteriori* (MAP) solutions: Given a penalty function U over errors a, an associated *prior* U^* on λ can be obtained by setting U^* to the convex (Fenchel) conjugate (Borwein & Lewis, 2000) of U. Vice versa, given the convex conjugate cost function U^*, the corresponding penalty function U can be derived by using the property of *Fenchel biconjugation* (Borwein & Lewis, 2000); that is, the conjugate of the conjugate of a convex function is the original convex function, $U = U^{**}$.

To illustrate, consider a quadratic penalty $U(a) = \sum_{i=1}^N \frac{1}{2}\sigma_i^2 a_i^2$. Here the convex conjugate $U^*(\lambda) = \sum_{i=1}^N \frac{\lambda_i^2}{2\sigma_i^2}$ can be determined by setting $a_i = \frac{\lambda_i}{\sigma_i^2}$; which specifies a Gaussian prior on λ. A different example can be obtained by considering the Laplacian prior on λ, $U^*(\lambda) = \|\lambda\|_1 = \sum_{i=1}^N \mid \lambda_i \mid$, which leads to the penalty function

$$U(a) = \begin{cases} 0 & \|a\|_\infty = \max_{i=1}^N \mid a_i \mid \leq 1 \\ \infty & \text{otherwise} \end{cases}$$

that forces hard inequality constraints.

Note that in each case, given a prior U^*, the standard MAP estimate maximizes the penalized log-likelihood $R(\lambda) = \sum_y \tilde{p}(y) \log p_\lambda(y) - U^*(\lambda)$. Our key result is that locally maximizing $R(\lambda)$ is equivalent to satisfying the feasibility constraints (2) of the RLME principle.

Theorem 1. *Under the log-linear assumption, locally maximizing the posterior probability of log-linear models on incomplete data is equivalent to satisfying the feasibility constraints of the RLME principle. That is, the only distinction between MAP and RLME in log-linear models is that, among local maxima (feasible solutions), RLME selects the model with the maximum regularized entropy, whereas MAP selects the model with the maximum posterior probability (Wang et al., 2004).*

That is, to find feasible solutions it suffices to find models that maximize the penalized log-likelihood (posterior) on observed data using standard iterative approaches. It is important to emphasize, however, that EM will only find alternative feasible solutions, while the RLME and MAP principles will differ markedly in the feasible solutions they prefer. We illustrate this distinction below.

To find feasible solutions, we use an iterative procedure, R-EM-IS, which employs an EM algorithm (Dempster, Laird, & Rubin, 1977) as an outer loop, but uses a nested GIS/IIS algorithm (Berger, Della Pietra, & Della Pietra, 1996; Della Pietra, Della Pietra, & Lafferty, 1997) to perform the internal M step. To derive this algorithm, first decompose the penalized log-likelihood function $R(\lambda)$ into

$$R(\lambda) = \sum_{y \in \tilde{\mathcal{Y}}} \tilde{p}(y) \log p_\lambda(y) - U^*(\lambda) \;\; = \;\; Q(\lambda, \lambda') \;+\; H(\lambda, \lambda')$$

where $Q(\lambda, \lambda') = \sum_{y \in \tilde{\mathcal{Y}}} \tilde{p}(y) \sum_{z \in \mathcal{Z}} p_{\lambda'}(z \mid y) \log p_\lambda(x) - U^*(\lambda)$, $H(\lambda, \lambda') = -\sum_{y \in \tilde{\mathcal{Y}}} \tilde{p}(y) \sum_{z \in \mathcal{Z}} p_{\lambda'}(z \mid y) \log p_\lambda(z \mid y)$. This is a standard decomposition used for deriving EM. For log-linear models, in particular, we have

$$Q\big(\lambda, \lambda^{(j)}\big) = -\log(\Phi_\lambda) + \sum_{i=1}^{N} \lambda_i \left(\sum_{y \in \tilde{\mathcal{Y}}} \tilde{p}(y) \sum_{z \in \mathcal{Z}} f_i(x)\, p_{\lambda^{(j)}}(z \mid y) \right) - U^*(\lambda) \qquad (3)$$

Interestingly, it turns out that maximizing $Q(\lambda, \lambda^{(j)})$ as a function of λ for fixed $\lambda^{(j)}$ (the M step) is equivalent to solving another constrained optimization problem corresponding to a maximum entropy principle; but a much simpler one than before (Wang et al., 2004).

Lemma 1. *Maximizing $Q(\lambda, \lambda^{(j)})$ as a function of λ for fixed $\lambda^{(j)}$ is equivalent to solving*

$$\max_{p,a} \quad H(p) - U(a) \;=\; -\sum_{x} p(x) \log p(x) - U(a) \qquad (4)$$

subject to

$$\sum_{x} p(x) f_i(x) \;=\; \sum_{y} \tilde{p}(y) \sum_{z} p_{\lambda^{(j)}}(z \mid y)\; f_i(y, z) + a_i; \quad i = 1, ..., N \qquad (5)$$

It is critical to realize that the new constrained optimization problem in Lemma 1 is much easier than maximizing (1) subject to (2) for log-linear models, because the right hand side of the constraints (5) no longer depends on λ but rather on the fixed constants from the previous iteration $\lambda^{(j)}$. This means that maximizing (4) subject to (5) with respect to λ is now a convex optimization problem with linear constraints. The generalized iterative scaling algorithm (GIS) (Darroch and Ratchliff, 1972) or improved iterative scaling algorithm (IIS) (Della Pietra, Della Pietra, and Lafferty, 1997) can be used to maximize $Q(\lambda, \lambda^{(j)})$ easily.

From these observations, we can recover feasible log-linear models by using an algorithm that combines EM with nested iterative scaling to calculate the M step. Assuming the Gaussian prior, the explicit iterative procedure we obtain will be as follows.

R-EM-IS algorithm

E step: Compute $\sum_y \tilde{p}(y) \sum_z p_{\lambda^{(j)}}(z \mid y) f_i(y, z)$ for $i = 1, ..., N$.
M step: Perform K parallel updates of the parameter values λ_i for $i = 1, ..., N$ by iterative scaling (GIS or IIS) as follows

$$\lambda_i^{(j+s/K)} = \lambda_i^{(j+(s-1)/K)} + \gamma_i^{(j+s/K)}; \quad s = 1, ..., K \tag{6}$$

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

$$\sum_x p_{\lambda^{(j+(s-1)/K)}}(x) f_i(x) e^{\gamma_i^{(j+s/K)} f(x)} + \frac{\lambda_i^{(j+(s-1)/K)} + \gamma_i^{(j+s/K)}}{\sigma_i^2} = \sum_y \tilde{p}(y) \sum_z p_{\lambda^{(j)}}(z \mid y) f_i(y, z) \tag{7}$$

where $f(x) = \sum_{i=1}^N f_i(x)$. The value of $\gamma_i^{(j+s/K)}$ can be obtained by bisection line search or solving the nonlinear Eq. (7) by Newton-Raphson iteration.

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 penalized log-likelihood and a feasible solution of the RLME problem; otherwise, we iterate until the constraints are met.

Provided that the E and M steps can both be computed, R-EM-IS can be shown to converge to a local maximum in penalized likelihood for log-linear models, and hence is guaranteed to yield feasible solutions to the RLME principle.

Theorem 2. *The R-EM-IS algorithm monotonically increases the penalized log-likelihood function $R(\lambda)$, and all limit points of any R-EM-IS sequence $\{\lambda^{(j+s/K)}, j \geq 0\}, s = 1, ..., K$, belong to the set*

$$\Gamma = \left\{ \lambda \in \Re^N : \frac{\partial R(\lambda)}{\partial \lambda} = 0 \right\} \tag{8}$$

Therefore, R-EM-IS asymptotically yields feasible solutions to the RLME principle for log-linear models (Wang et al., 2004).

Thus, R-EM-IS provides an effective means to find feasible solutions to the RLME principle. (We note that Lauritzen, 1995 has suggested a similar algorithm for the unregularized case, but did not supply a convergence proof. More recently, Riezler (1999) has also proposed an algorithm equivalent to setting $K = 1$ in EM-IS for the unregularized case.

However, we have found $K > 1$ to be more effective in many situations, and regularization also usually yields a substantial benefit.)

We can now exploit the R-EM-IS algorithm to develop a practical approximation to the RLME principle.

R-ME-EM-IS algorithm:
Initialization: Randomly choose initial guesses for λ.
R-EM-IS: Run R-EM-IS to convergence, to obtain feasible λ^*.
Entropy calculation: Calculate the regularized entropy of p_{λ^*}.
Model selection: Repeat the above steps several times to produce a set of distinct feasible candidates. Choose the feasible candidate that achieves the highest regularized entropy.

This leads to a new estimation technique that we will compare to standard MAP estimation below. One apparent complication, first, is that we need to calculate the entropies of the candidate models produced by R-EM-IS. However, it turns out that we do not need to calculate entropies explicitly because one can recover the entropy of *feasible* log-linear models simply as a byproduct of running R-EM-IS to convergence.

Corollary 1. *If* λ^* *is feasible, then* $Q(\lambda^*, \lambda^*) = -H(p_{\lambda^*}) + U(a^*)$ *and* $R(\lambda^*) = -H(p_{\lambda^*}) + U(a^*) + H(\lambda^*, \lambda^*)$.

Therefore, at a feasible solution λ^*, we have already calculated the regularized entropy, $-Q(\lambda^*, \lambda^*)$, in the M step of R-EM-IS.

When there is no hidden variable, $H(\lambda^*, \lambda^*) = 0$, we have a unique solution $R(\lambda^*) = -H(p_{\lambda^*}) + U(a^*)$, the standard global optimal regularized maximum entropy model is equivalent to the global optimal MAP solution. When there are hidden variables, we have multiple solutions which make things different. To draw a clear distinction between RLME and MAP, assume that the term $H(\lambda^*, \lambda^*)$ is constant across different feasible solutions. Then MAP, which maximizes $R(\lambda^*)$, will choose the model that has lowest regularized entropy, whereas RLME, which maximizes $H(p_{\lambda^*}) - U(a^*)$, will chose a model that has least regularized likelihood. (Of course, $H(\lambda^*, \lambda^*)$ will not be constant in practice and the comparison between RLME and MAP is not so straightforward, but this example does highlight their difference.) The fact that RLME and MAP are different raises the question of which method is the most effective when inferring a model from sample data. To address this question we turn to a comparison.

4. RLME for language modeling

The regularized latent maximum entropy principle can be used to model natural language in a principled way by combining different exponential models to obtain rich expressive power. In this section, we describe how to use the RLME principle to combine the trigram Markov model with probabilistic latent semantic analysis (PLSA) to obtain a better language model.

Currently almost all maximum entropy language models use the conditional form first proposed by Brown et al. for statistical machine translation (Brown et al., 1992). The main

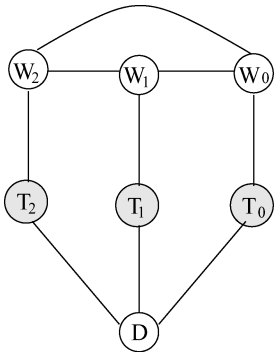

Figure 1. A graphical representation of the semantic tri-gram model, where the curve that connects the three word nodes together denotes the tri-gram feature. In this graphical representation, many arcs share the same parameters.

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.

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, and D is a document identifier. Thus, $y = (W_2, W_1, W_0, D)$ is the observed data and $z = (T_2, T_1, T_0)$ is unobserved. 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 1.

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.

$$\sum_x p(x)\delta(W_2 = w_i, W_1 = w_j, W_0 = w_k) = \sum_d \tilde{p}(d)\tilde{p}(w_i w_j w_k \mid d)$$

$$\sum_x p(x) \sum_{\ell=0}^{1} \delta(W_{\ell+1} = w_i, W_\ell = w_j) = \sum_d \tilde{p}(d) \sum_{\ell=0}^{1} \tilde{p}(W_{\ell+1} = w_i, W_\ell = w_j \mid d)$$

$$\sum_x p(x) \sum_{\ell=0}^{2} \delta(W_\ell = w_i) = \sum_d \tilde{p}(d) \sum_{\ell=0}^{2} \tilde{p}(W_\ell = w_i \mid d) \tag{9}$$

Here, for example, $\tilde{p}(w_i w_j w_k \mid d)$ denotes the empirical distribution of the tri-grams actually seen in the training corpus, and $\delta(\cdot)$ is an indicator that returns 1 if the event is

active and 0 otherwise. Note the δ functions specify the features that the learned model $p(x)$ should respect, which above are the tri-gram, bi-gram and uni-gram constraints 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

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

$$\sum_x p(x) \sum_{\ell=0}^{2} \delta(T_\ell = t, W_\ell = w_i) = \sum_d \tilde{p}(d) \sum_{\ell=0}^{2} \tilde{p}(W_\ell = w_i \mid d) \times p(T_\ell = t \mid W_\ell = w_i, D = d) \tag{10}$$

Again, these δ functions specify the features the learned model should respect. The first equality imposes the $\mid D \mid \times \mid T \mid$ constraints between the document node and the topic nodes, and the second equality imposes the $\mid W \mid \times \mid T \mid$ constraints between the topic nodes and words.

We can now learn a probability model that simultaneously takes all of these information sources into account, by employing the RLME 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 1 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.

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 (Khudanpur & Wu, 2000), 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 obtain

$$\begin{aligned}
\Phi &= \sum_{w_2, w_1, w_0, t_2, t_1, t_0, d} (e^{\lambda_{w_2}} e^{\lambda_{w_1}} e^{\lambda_{w_0}} e^{\lambda_{w_2 w_1}} e^{\lambda_{w_1 w_0}} e^{\lambda_{w_2 w_1 w_0}} e^{\lambda_{w_2 t_2}} e^{\lambda_{w_1 t_1}} e^{\lambda_{w_0 t_0}} e^{\lambda_{t_2 d}} e^{\lambda_{t_1 d}} e^{\lambda_{t_0 d}}) \\
&= \sum_{w_0} e^{\lambda_{w_0}} \sum_{t_0} e^{\lambda_{w_0 t_0}} \sum_{w_1} e^{\lambda_{w_1}} e^{\lambda_{w_1 w_o}} \sum_{t_1} e^{\lambda_{w_1 t_1}} \\
&\quad \sum_{w_2} e^{\lambda_{w_2}} e^{\lambda_{w_1 w_2}} e^{\lambda_{w_2 w_1 w_0}} \sum_{t_2} e^{\lambda_{w_2 t_2}} \left(\sum_d e^{\lambda_{t_2 d}} e^{\lambda_{t_1 d}} e^{\lambda_{t_0 d}} \right)
\end{aligned} \tag{11}$$

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

$$\begin{aligned}
&\sum_x p(x)\delta(W_2 = w_i, W_1 = w_j, W_0 = w_k) \\
&\quad = \Phi^{-1} e^{\lambda_{w_i}} e^{\lambda_{w_j}} e^{\lambda_{w_k}} e^{\lambda_{w_i w_j}} e^{\lambda_{w_j w_k}} e^{\lambda_{w_i w_j w_k}} \\
&\quad \times \sum_{t_0} e^{\lambda_{w_k t_0}} \sum_{t_1} e^{\lambda_{w_j t_1}} \sum_{t_2} e^{\lambda_{w_i t_2}} \\
&\quad \times \left(\sum_d e^{\lambda_{t_2 d}} e^{\lambda_{t_1 d}} e^{\lambda_{t_0 d}} \right)
\end{aligned} \tag{12}$$

Semantic smoothing

To make use of semantic similarity and subtle variation between words, we can introduce an additional node C between each topic node and word node. The first set of feature constraints in (10) can be augmented to incorporate this new cluster variable C as follows:

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

The effect of these cluster nodes critically depends on the range of their variation. For example, if all the words are grouped into a single class, then the model will be maximally smoothed. On the other hand, if there are as many classes as words in the vocabulary, there will be no smoothing effect at all. There is a trade-off between smoothing to reduce the effective number of parameters in the model, and non-smoothing to permit a more detailed model.

As a further extension which takes account of the semantic similarity and sub-topic variation within each document and among documents, we can introduce additional node S between the topic nodes and the document node. Again, the second set of feature constraints as in (10) can be written analogously as follows:

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

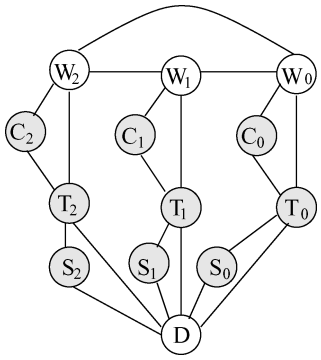

Figure 2. A graphical representation of the extended PLSA tri-gram model, which includes word cluster variables, C_2, C_1, C_0, and topic cluster variables, S_2, S_1, S_0. The cliques of $W-C-T$ and $T-S-D$ correspond to the features in Section 4.3. In this graphical representation, many arcs share the same parameters.

Again the effect of node S critically depends on the range of its variation. If all the documents are grouped in a single cluster, then the model is the same as (10) and is over-smoothed, and in the context of diverse discourse this could not capture the specific topics. On the other hand, if there are as many clusters as documents in the corpus, there will be no smoothing effect at all. Again, we encounter a trade-off between smoothing to reduce parameters, versus non-smoothing to permit variation.

Note that the benefit of the maximum entropy combination method is that the cluster nodes behave like latent variables in a mixture model for "soft clustering", instead of the "hard clusters" created by methods like K-means used in Bellegarda (2000).

Figure 2 shows an extended semantic smoothed version of the model that incorporates additional word cluster variables within each topic, and additional topic cluster variables with each document.

Computation in testing

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

$$\begin{aligned} p(w_1 \ldots w_{|d|}, d) &= \prod_{\ell=1}^{|d|} p_{D_\ell}(w_\ell, d_\ell \mid w_1 \ldots w_{\ell-1}) \\ &= \prod_{\ell=1}^{|d|} \sum_{T_2,T_1,T_0} p_{D_\ell}(w_\ell, d_\ell, T_2, T_1, T_0 \mid w_1 \ldots w_{\ell-1}) \\ &= \prod_{\ell=1}^{|d|} \sum_{T_2,T_1,T_0} p_{D_\ell}(w_\ell, d_\ell, T_2, T_1, T_0 \mid w_{\ell-2}, w_{\ell-1}) \end{aligned}$$

where $\mid d \mid$ denotes the length of the test document d. Since the representation for a document of the test data is not contained in the original training corpus, we use similar "fold-in"

heuristic approach as used in Hofmann (2001): the Lagrange multipliers corresponding to the document-topic arcs are re-estimated by the same formulation as (10) while holding other parameters fixed, where the empirical distribution is given by the current updated document history. To be more precise, let $p_{D_0}(x)$ to be the exponential distribution for Figure 1 with the estimated RLME or MAP Lagrange multipliers as weights, except that the multipliers corresponding to the document-topic arcs are fixed to be zero. Then the new model p_{D_ℓ} will be the result of the following optimization

$$\min_{p,a} \quad KL(p, p_{D_{\ell-1}}) + U(a)$$

subject to

$$\sum_x p(x) \sum_{\ell=0}^{2} \delta(T_\ell = t, D = d_\ell) = \sum_{\ell=0}^{2} \tilde{p}(d_\ell) \sum_{W_\ell} \tilde{p}(W_\ell \mid d_\ell) p(T_\ell = t \mid W_\ell, d_\ell) + a_{t,d}$$

where $KL(p, q)$ is the Kullback-Leiber divergence.

Since our model provides the probability of complete data $p_{D_\ell}(W_2, W_1, W_0, D_\ell, T_2, T_1, T_0)$, the conditional probability $p_{D_\ell}(W_0, D_\ell, T_2, T_1, T_0 \mid W_2, W_1)$ can be easily obtained by marginalization (and division).

5. Experimental evaluation

Experimental data sets and performance measure

The corpus used to train our model was taken from the WSJ portion of the NAB corpus and was composed of about 150,000 documents spanning the years 1987 to 1989, comprising approximately 42 millions words. The vocabulary was constructed by taking the 20,000 most frequent words of the training data. We split another separate set of data consisting of 325,000 words taken from the year 1989 into half, one half used as development data by random selection, another half for testing.

The statistics of the datasets are shown in Table 1.

To evaluate a language model we use the standard definitions of perplexity and entropy on held-out test data. That is, given a test corpus $s = \{d_1, \ldots, d_L\}$ and a language model

Table 1. Data sets statistics.

	No. of articles	No. of sentences	No. of words
Train	150,981	1,611,571	41,780,924
Dev	378	6904	157,312
Test	379	6638	153,801

defining $p(s)$ we calculate test perplexity and test entropy as

$$\begin{aligned} \textit{Perplexity} &= \sqrt[|s|]{\frac{1}{p(s)}} = \sqrt[|s|]{\prod_{i=1}^{L} \frac{1}{p(w_1 \ldots w_{|d_i|}, d_i)}} \\ &= \sqrt[|s|]{\prod_{i=1}^{L} \prod_{l=1}^{|d_i|} \frac{1}{p(w_l, d_i \mid w_i \ldots w_{l-1})}} \\ \textit{Entropy} &= \log_2 \textit{Perplexity} \end{aligned}$$

where $\mid s \mid = \sum_{i=1}^{L} \mid d_i \mid$ is the length of test corpus. The goal is to obtain small values of these measures. That is, 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).

Experimental design

To serve as a baseline standard of performance, we use a conventional tri-gram model with Good-Turing back-off smoothing. Implementing this approach with the CMU language modeling toolkit (Clarkson & Rosenfeld, 1997), we obtained a test perplexity score of 103.

We then implemented and compared a few simple versions of our RLME approach using the models described above. Note that in all of the experiments below, to implement the inner R-EM-IS procedure we set the number of EM iterations to 5 and the number of internal IIS loop iterations to 20. We also had to choose a variance for the Gaussian prior to serve as a regularizer, but simply chose the default value of $\sigma_i = 1$ for every feature $i = 1, ..., N$ in each case, and therefore performed no optimization of the regularization parameters.

To control for the effects of maximizing regularized entropy (RLME) verus maximizing a posteriori probability (MAP), we first omitted the outer R-ME-EM-IS procedure and instead just initialized the parameters to zero and executed a single run of R-EM-IS, then perturb the parameters randomly and run a single R-EM-IS to find a single locally MAP model (or equivalently a single feasible model for the RLME principle). Then, using these results as a control, we re-ran the procedures with the outer R-ME-EM-IS procedure re-introduced, to find higher regularized entropy (RLME) solutions and higher penalized likelihood (MAP) solutions. Specifically, we used 20 random starting points for λ, ran R-EM-IS from each, and then selected the highest regularized entropy solution as the RLME estimate, and the highest penalized maximum likelihood solution as the MAP estimate. Among all feasible solutions, the one which has the lowest perplexity score on the development corpus is denoted as *best feasible solution*.

We first considered a simple model which only considered the features (and the constraints) from the simple tri-gram model. In this simple case, there are no hidden variables, and our estimation principles (MAP, RLME, a single run of R-EM-IS) all reduce to the same standard (regularized) ME principle for completely observed data. Here we observed a perplexity score of 112 on development data, which is slightly worse than the 109 perplexity

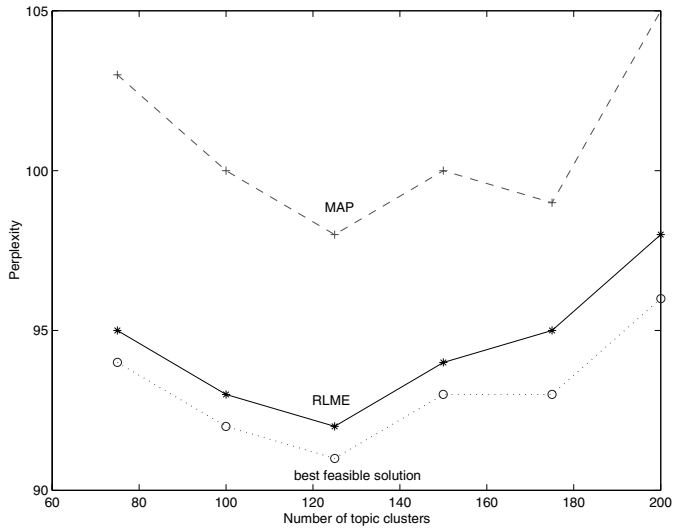

Figure 3. Perplexity versus number of topics for the semantic tri-gram language model.

obtained by training a tri-gram model with Good-Turing smoothing. Nevertheless, this result shows that the parameter smoothing effect achieved by our (untuned) regularization principle is nearly as effective as using Good-Turing smoothing for N-gram models.[1]

Next, we introduced the PLSA features to our model, setting the number of possible hidden topics to be $| T | = 125$. First, running R-EM-IS from a single starting point resulted in a model that achieved perplexity score 95 on the development corpus; comprising a 12.8% reduction in perplexity from the baseline tri-gram model. Then, using 20 runs from different starting points to yield the RLME, MAP and best feasible solutions estimates, we obtained models with perplexities 92, 95 and 91 respectively; comprising respective 15.6, 10.6 and 16.5% reductions in perplexity from the baseline tri-gram model. Figure 3 shows how the perplexities of these estimated models change as the number of topics is increased and decreased, with the best perplexities achieved in each case when the number equals 125. Therefore, in the remaining experiments we fix the number of hidden topics to be 125.

Finally, we considered a few augmentations of the basic combined tri-gram, PLSA model illustrated in figure 1, which is shown in figure 2.

First, when we add just the *word* cluster nodes to our model, we find that the result is sensitive to the number of classes. For the single feasible model which is given by perturbing the parameters of a single run of R-EM-IS from uniform initializing, then running a single R-EM-IS, we find that when the class number is chosen to be 10, the perplexity achieved is now 93, which equals to the best result achieved by any technique above. However, if the class number is set to 50, then the perplexity increases to 95, probably due to the huge increase in parameters. For the MAP and RLME estimators we find that we obtain similar results. Figure 4 shows how the perplexity changes as the number of word clusters changes for each of the best feasible solution, MAP and RLME estimates. The best perplexity achieved is 87 (by RLME) for a number of word clusters set to 30.

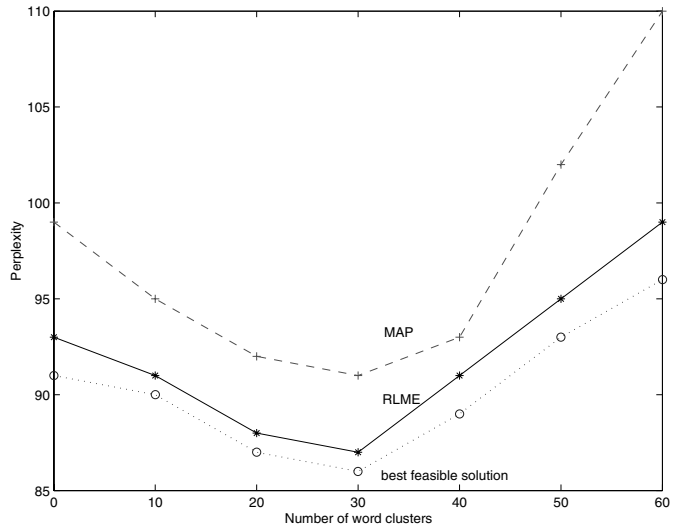

Figure 4. Perplexity versus number of word clusters for the extended PLSA tri-gram language model.

Next, when we add just the *topic* cluster nodes to the model, but omit the word cluster nodes, we also find that the result depends on the number of topic clusters. For the single feasible model which is given by perturbing the parameters of a single run of R-EM-IS from uniform initializing, then running a single R-EM-IS, we find that when the cluster number is chosen to be 10, the perplexity achieved is 92, which is the same as the previous perplexity of 92 (with no cluster variables). However, if the cluster number is set to 30, the perplexity increases to 93. For the MAP and RLME estimators, again, we obtain similar results. Figure 5 shows how the perplexity changes as the number of topic clusters changes for each of the best feasible solution, MAP and RLME estimates. The best perplexity achieved is 90 (by RLME) when the number of topic clusters is set to 20.

Finally, when we add *both* the word cluster nodes and topic cluster nodes simultaneously to our model, we find that the result is again sensitive to the number of classes, but notably improved. When the number of word classes is 10 and the number of topic clusters is 10, the perplexity achieved by the single run feasible model is now 91, which is the best performance it achieves, and comprises about an 16.5% improvement over the baseline tri-gram model. Similarly, the MAP and RLME estimators show improved results. Figures 6 and 7 shows how the perplexity changes as the number of topic clusters changes for each of the single feasible, MAP and RLME estimates. The best perplexity achieved is now 85 by RLME when the number of word and topic clusters is set to 10 and 20 respectively. This result is a substantial improvement over the baseline tri-gram model, comprising a 22% reduction in perplexity. The best perplexity achieved is now 92 by MAP when both numbers of word and topic clusters is set to 0. This result is also a substantial improvement over the baseline tri-gram model, comprising a 15.6% reduction in perplexity. The best perplexity achieved is now 84 by best feasible solution when the number of word and topic

Table 2. Perplexity and reduction results on development corpus for the semantic tri-gram RLME, MAP and best feasible solution models under various smoothing schemes.

Language model	RLME	MAP	Best feasible
Baseline	108 (0.0%)		
Semantic tri-gram	92 (14.8%)	98 (9.2%)	91 (15.7%)
Semantic tri-gram + Word smoothing	87 (19.4%)	91 (15.7%)	86 (20.3%)
Semantic tri-gram + Topic smoothing	90 (16.7%)	95 (12.0%)	89 (17.5%)
Semantic tri-gram + Word & Topic smoothing	85 (21.3%)	94 (13.0%)	84 (22.2%)

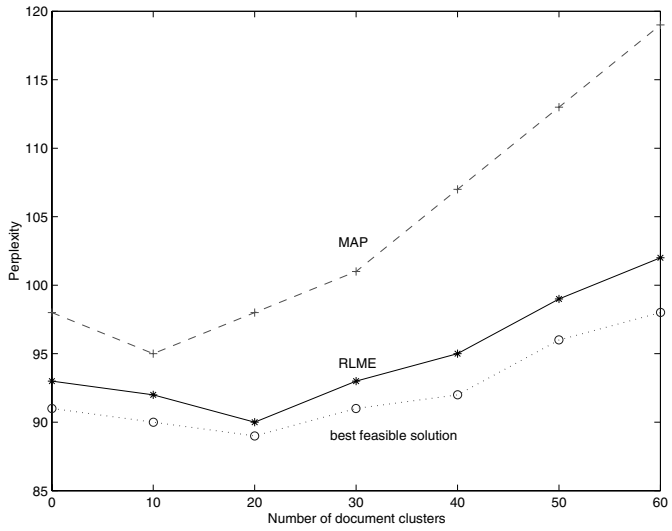

Figure 5. Perplexity versus number of topic clusters for the extended PLSA tri-gram language model.

clusters is set to 10 and 20 respectively. Table 2 summarizes the best results obtained in each case by RLME, MAP and best feasible solution models.

Once we use the development corpus to tune the optimal numbers of topic, word clusters, and topic clusters for the best feasible solution, MAP and RLME estimates, we calculate the perplexity on test corpus. The perplexity score by the baseline tri-gram model is 103. We obtain 84, 90 and 82 perplexity scores respectively by the tuned RLME, MAP and best feasible solution. The corresponding perplexity reductions are 18.5, 12.6; and 20.4%. Table 3 summarizes the results on test corpus by these models.

Table 3. Perplexity results on test corpus by best RLME, MAP and best feasible solution models.

Language model	Test perplexity	Reduction (%)
Baseline	103	
RLME	84	18.5
MAP	90	12.6
Best feasible	82	20.4

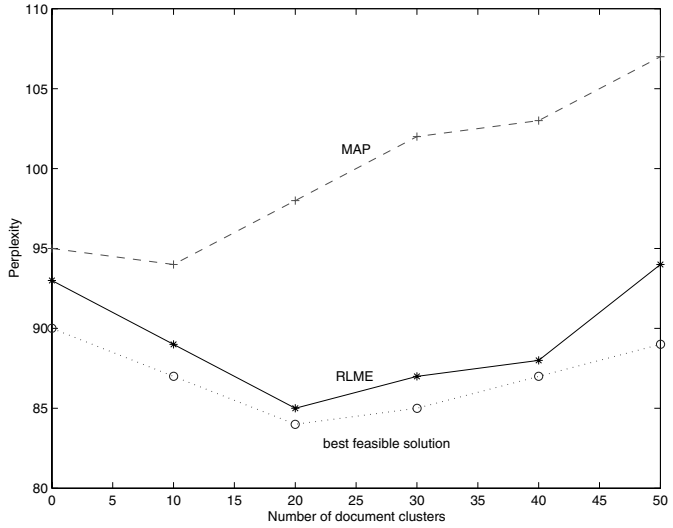

Figure 6. Perplexity versus number of word clusters for the extended PLSA tri-gram model when fixing the number of topic clusters to 10.

6. Discussion

The main observation to draw from this investigation is that extending language models to incorporate extra hidden features is conceptually very easy in the LME framework. All one has to do is identify the features they wish to model and add the respective constraints to the feasibility problem solved by R-EM-IS. Provided the E and M steps remain feasible, this presents no practical barrier to extending the model.

Clearly, however, standard concerns such as over-fitting avoidance remain inescapable issues when estimating model parameters from training data. Nevertheless, the above Figures 4– 7 seem to clearly indicate that RLME estimation consistently gives better results than MAP estimation for this type of large scale parameter estimation from sparse data. RLME appears to cope more robustly when simultaneously estimating a large number of parameters in the presence of hidden variables and sparse training data.

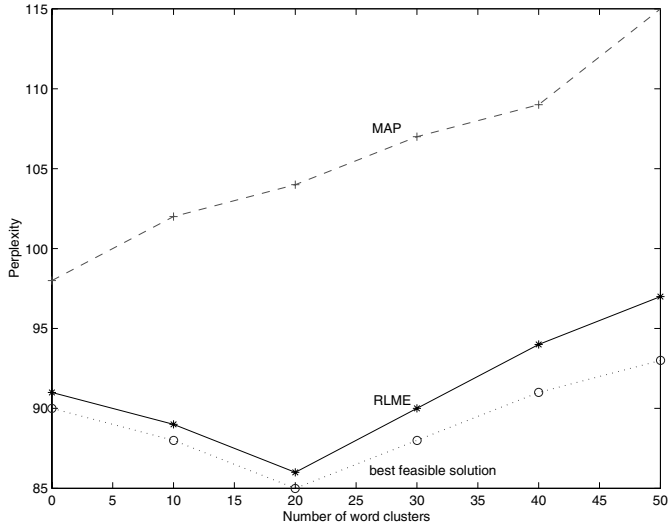

Figure 7. Perplexity versus number of word clusters for the extended PLSA tri-gram model when fixing the number of topic clusters to 20.

7. Extensions

We should note that there remain several avenues to improving the quality of the language models we are able to estimate from data. In language modeling research, there exists a body of ad hoc tricks which are known to make significant improvements. For example, (Bellegarda, 2000) proposes an ad hoc combination of tri-gram and latent semantic analysis (LSA) models, by using the perplexity formula

$$p(w_\ell \mid w_1 \dots w_{\ell-1}) = \frac{p(w_\ell \mid w_{\ell-2} w_{\ell-1}) p_{LSA}(d_\ell \mid w_\ell)}{\sum_{w_i} p(w_i \mid w_{\ell-2} w_{\ell-1}) p_{LSA}(d_\ell \mid w_i)} \tag{13}$$

$$= \frac{p(w_\ell \mid w_{\ell-2} w_{\ell-1}) \frac{p_{LSA}(w_\ell \mid d_\ell)}{p_{LSA}(w_\ell)}}{\sum_{w_i} p(w_i \mid w_{\ell-2} w_{\ell-1}) \frac{p_{LSA}(w_i \mid d_\ell)}{p_{LSA}(w_i)}} \tag{14}$$

Here $d_\ell = (w_1 \dots w_{\ell-1})$ is the current document d's history; $p_{LSA}(d_\ell \mid w_\ell)$ is the probability of the current document history given the current word w_ℓ; $p_{LSA}(w_\ell \mid d_\ell)$ is the probability of the current word w_ℓ history given the current document d_ℓ; and $p_{LSA}(w_\ell)$ is the probability of current word w_ℓ.[2] Each of these components is obtained by the latent semantic analysis. We calculated the perplexity of Bellagarda's model using the same training data and develop data considered above. The perplexity obtained by Bellegarda's model in this case is 99, which is an 9% reduction in perplexity compared to the baseline tri-gram model.

However, if one incorporates another ad hoc improvement from speech recognition research, a substantial futher reduction can be obtained. The idea is to strengthen the influence of the LSA portion of the model by raising its contribution to some power (here 7) and renormalizing

$$p(w_\ell \mid w_1 \ldots w_{\ell-1}) = \frac{p(w_\ell \mid w_{\ell-2} w_{\ell-1})(p_{LSA}(d_\ell \mid w_\ell))^7}{\sum_{w_i} p(w_i \mid w_{\ell-2} w_{\ell-1})(p_{LSA}(d_\ell \mid w_i))^7} \tag{15}$$

Doing so for Bellagarda's model results in a suprising and dramatic perplexity value of 86; almost equal to the best of our results using RLME above. Although we have yet to investigate such techniques in our RLME framework, given the potentially dramatic improvements they potentially offer, it remains important future research to understand how such improvements might be incorporated in a principled manner.

8. Conclusion

We have applied our proposed RLME principle for estimating sophisticated mixed chain-table graphical models of natural language, where local word interactions and global semantic document information can be modeled by mixtures of exponential families in a unified framework. The parameters are estimated in the sense of maximum entropy over a set of feasible solutions produced by an EM algorithm with nested iterative scaling. When modeling specific linguistic phenomena, the general framework reproduces standard models. However, our proposed approach allows one to model interactions among multiple aspects of natural language simultaneously and automatically. This yields dramatic improvements over standard maximum a posteriori estimation in our experiments on natural language modeling.

Although we have shown the potential benefits of RLME over standard MAP estimation for large scale parameter estimation problems, the R-ME-EM-IS algorithm we use to find solutions is not a very sophisticated optimization algorithm, and is probably needlessly time consuming as a result. It would be worthwhile to investigate more direct optimization methods which simultaneously seek to achieve feasibility while maximizing the objective.

The experimental results also show that performance is sensitive to the number of word clusters and topic clusters. We are currently investigating techniques to automatically determine optimal word and topic cluster numbers.

In principle, our RLME approach provides a general statistical framework for incorporating arbitrary aspects of natural language into a parametric model. Therefore, we could hope to incorporate syntactic structure encoded by probabilistic context free grammars into the RLME framework. However, in practice, this raises some difficult challenges. In particular, these models appear to make the left hand side of the constraints (2) infeasible to calculate. It also appears to be difficult to apply standard approximation techniques, such as loopy belief propagation and variational approximation (Wainwright & Jordan, 2003) to this problem. One possibility is to resort to computationally expensive Monte Carlo methods, such as those employed by Abney (1997).

Acknowledgments

This work is supported by AICML, MITACS, NSERC, and the Canada Research Chairs program. Fuchun Peng is also supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors. Finally we thank the anonymous reviewers for a much improved version.

Notes

1. It turns out that regularization has a significant effect in this case, because removing it leads to inferior results (perplexity scores over 150 (Chen & Rosenfeld, 2000). Therefore, we stick to our standard regularization parameters in all further experiments.
2. $p_{LSA}(w_\ell)$ is not the standard unigram probability as obtained using the whole training corpus (as claimed in Bellegarda (2000), one line below equation 27, page 1289). Intuitively, it is the weighted unigram averaged over possible semantic information.

References

Abney, S. (1997). Stochastic attribute-value grammars. *Computational Linguistics, 23:4*, 597–618.

Bellegarda, J. (2000). Exploiting latent semantic information in statistical language modeling. *Proceedings of IEEE, 88:8*, 1279–1296.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research, 3*, 1137–1155.

Berger, A., Della Pietra, S., & Della Pietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics, 22:1*, 39–71.

Borwein, J., & Lewis, A. (2000). *Convex analysis and nonlinear optimization: Theory and examples*. Springer.

Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R., Nadas, A., & Roukos, S. (1992). A maximum entropy construction of conditional log-linear language and translation models using learned features and a generalized Csiszar algorithm. IBM Report.

Chelba, C., & Jelinek, F. (2000). Structured language modeling. *Computer Speech and Language, 14:4*, 283–332.

Chen, S., & Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech and Language, 13:4*, 319–358.

Chen, S., & Rosenfeld, R. (2000). A survey of smoothing techniques for ME models. *IEEE Trans. on Speech and Audio Processing, 8:1*, 37–244.

Clarkson, P., & Rosenfeld, R. (1997). Statistical language modeling using the CMU-Cambridge toolkit. *Proceedings of Eurospeech*, 2707–2710.

Csiszar, I. (1996). Maxent, mathematics, and information theory. In K. Hanson and R. Silver (Eds.), *Maximum Entropy and Bayesian Methods* (pp. 35–50). Kluwer Academic Publishers

Darroch, J., & Ratchliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics, 43:5*, 1470–1480.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:4*, 380–393.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of Royal Statistical Society, Series B, 39*, 1–38.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning, 42:1*, 177–196.

Jaynes, E. (1983). Papers on probability, statistics, and statistical physics. In R. Rosenkrantz & D. Reidel, Publishing Company.

Jelinek, F., & Mercer, R. (1980). Interpolated estimation of Markov source parameters from sparse data. In E. Gelsema, & L. Kanal, (Eds.), *Pattern Recognition in Practice*. (pp. 381–397) North Holland.

Jelinek, F. (1998). *Statistical methods for speech recognition*. MIT Press.

Khudanpur, S., & Wu, J. (2000). Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language, 14:4*, 355–372.

Lafferty, J., & Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR).

Lauritzen, S. (1995). The EM-algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis, 1*, 191–201.

Peng, F., Schuurmans, D., & Wang, S. (2004). Augumenting naive Bayes text classifier using statistical language models. *Information Retrieval, 7:3–4*, 317–345.

Riezler, S. (1999). Probabilistic constraint logic programming. Ph.D. Dissertation, University of Stuttgart, Germany.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. *Computational Linguistics, 27:2*, 249–285.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language, 10*, 187–228.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here?. *Proceedings of the IEEE, 88:8*, 1270–1278.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*, 379–423.

Wainwright, M., & Jordan, M. (2003). Graphical models, exponential families, and variational inference. Technical Report 649, Department of Statistics, University of California, Berkeley.

Wang, S., Schuurmans, D., & Zhao, Y. (2003). The latent maximum entropy principle. Manuscript submitted.

Wang, S., Schuurmans, D., Peng, F., & Zhao, Y. (2004). Learning mixture models with the regularized latent maximum entropy principle. *IEEE Transactions on Neural Networks: Special Issue on Information Theoretic Learning*, *154*.

Received October 8, 2003
Revised May 19, 2004
Accepted May 24, 2004