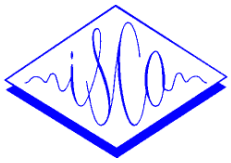

6th International Conference on Spoken Language Processing (ICSLP 2000)
Beijing, China
October 16-20, 2000

EFFICIENT TRAINING METHODS FOR MAXIMUM ENTROPY LANGUAGE MODELING

Jun Wu and Sanjeev Khudanpur *

Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218-2686, USA
{junwu,sanjeev}@mail.clsp.jhu.edu

ABSTRACT

Maximum entropy language modeling techniques combine different sources of statistical dependence, such as syntactic relationships, topic cohesiveness and collocation frequency, in a unified and effective language model. These techniques however are also computationally very intensive, particularly during model estimation, compared to the more prevalent alternative of interpolating several simple models, each capturing one type of dependency. In this paper we present ways which significantly reduce this complexity by reorganizing the required computations. We show that in case of a model with N-gram constraints, each iteration of the parameter estimation algorithm requires the same amount of computation as estimating a comparable back-off N-gram model. In general, the computational cost of each iteration in model estimation is linear in the number of distinct "histories" seen in the training corpus, times a model-class dependent factor. The reorganization focuses mainly on reducing this multiplicative factor from the size of the vocabulary to the average number of words seen following a history. A 15-fold speed-up has been observed by using this method in estimating a language model that incorporates syntactic head-word constraints, nonterminal-label constraints and topic-unigram constraints with N-grams for the Switchboard corpus. This model achieves a perplexity reduction of 13% and a word error rate reduction of 1.5% absolute compared to a trigram model.

1. INTRODUCTION

Several kinds of non-local dependencies, e.g., the syntactic heads in the parse of a sentence [1] and its topic [8, 5], have been shown to improve the performance of statistical language models. It is natural to build a language model that uses all these sources of information. Maximum entropy (ME) is a technique for combining different sources of dependencies in one unified framework [7]. An ME model incorporating syntactic head-word constraints, non-terminal label constraints, and topic constraints with conventional N-gram constraints has the form:

$$P(w|w_{i-1}, w_{-2}, h_{-1}, h_{-2}, nt_{-1}, nt_{-2}, topic) \quad (1)$$
$$= \frac{1}{Z} \cdot e^{\lambda(w)} \cdot e^{\lambda(w_{-1},w)} \cdot e^{\lambda(w_{-2},w_{-1},w)} \cdot e^{\lambda(h_{-1},w)}$$
$$\cdot e^{\lambda(h_{-2},h_{-1},w)} \cdot e^{\lambda(nt_{-1},w)} \cdot e^{\lambda(nt_{-2},nt_{-1},w)} \cdot e^{\lambda(topic,w)}$$

where

w is the current word to be predicted,
w_{-1}, w_{-2} are two previous words,
h_{-1}, h_{-2} are two preceding exposed head-words of the syntactic partial parse.
nt_{-1}, nt_{-2} are two preceding exposed non-terminal labels of the partial parse, and
$Z = Z(w_{-2}, w_{-1}, h_{-2}, h_{-1}, nt_{-2}, nt_{-1}, topic)$ is the normalization factor.

$\lambda(w)$ is independent of histories and is thus called a marginal constraint. $\lambda(w_{-1}, w)$,...,$\lambda(topic, w)$ are history dependent and are called conditional constraints. Details of syntactic language modeling and topic sensitive language modeling are available in [1, 5, 7].

A fundamental difficulty in implementing such a complicated language model is the heavy computational cost of the training procedure. In state-of-the-art training algorithms [3], the computational complexity for each iteration in the training is $O(|X| \cdot |\bar{C}|)$, where $|X|$ is the number of seen histories and $|\bar{C}|$ is the average number of words with any conditional constraint for each history. Although this algorithm is much better than a straight forward implementation with the complexity of $O(|X| \cdot |V|)$, where V is the vocabulary, it is still not practical for complicated language models and/or for large tasks. For instance, even in a small task such as Switchboard [4] with only 2.1M words in the training data, $|X| \sim 10^6$ and $|\bar{C}| \sim 10^3$ for the model (1). These numbers go up by one to two orders of magnitude for larger tasks such as Broadcast News (BN).

We present a new and efficient hierarchical training method for ME model estimation. We have applied this method to train a trigram model and the composite language model of (1) for Switchboard and have achieved 30 fold and 15 fold speed-ups respectively over previously used algorithms.

We will introduce the background of the ME method and the standard training algorithm in Section 2. In Section 3, we describe the idea of our hierarchical training method by the example of a trigram model and a model with two kinds of overlapping bigram constraints. Section 4 shows the nominal speed-up of our method for different tasks and the actual training time for Switchboard. The speech recognition performance of the composite ME model is also reported in this section.

2. BACKGROUND

2.1. ME Principle

Let X be a history and Y be the set of following words. The basic problem of statistical language modeling is to

*This work was partially supported by the U.S. National Science Foundation under a STIMULATE grant: No IRI-9618874.

estimate a probability measure $p(X,Y)$ from observations $\langle x_1, y_1\rangle$,..., $\langle x_L, y_L\rangle$, where L is the number of training data. For example, in a trigram model, Y is the vocabulary and X is the set of seen bigrams. In the maximum entropy framework, a set of binary features, G, is defined as

$$G = \{g_i : (X \times Y) \rightarrow \{0,1\}\}.$$

Each binary feature $g_i(x,y)$, $i = 1,...,k$, partitions the domain X, Y into two sets: those $\langle x,y\rangle$ for which $g_i(x,y)$ is active and those for which it is not. A vector $\mathbf{a} = a_1,...,a_k$ is chosen to be the target expectations for these features based on the observed samples $\langle x_1, y_1\rangle$,..., $\langle x_L, y_L\rangle$. Now, the feature set G and their target expectations define a class $\mathcal{P}$ of all probability distributions whose feature expectations match the target expectations,

$$\mathcal{P} = \{p : p[g_i(x,y)] = a_i \ \forall \ i = 1,...,k\},$$

where $p[g_i(x,y)]$ is the expectation of $g_i(x,y)$ with respect to the distribution $p(x,y)$, as

$$p[g_i(x,y)] = \sum_{\langle x,y\rangle} p(x,y) \cdot g_i(x,y). \qquad (2)$$

Given such a class $\mathcal{P}$ of probability distributions, we want to find the distribution $p^*(x,y)$ in $\mathcal{P}$ that maximizes the entropy $H(p)$ where

$$H(p) = -\sum_{\langle x,y\rangle} p(x,y) \cdot log\ p(x,y).$$

The solution $p^*(x,y)$ has an exponential form [2]. So we consider the alternate class $\mathcal{R}$ of exponential models $m(y|x)$ defined over our features G as

$$\mathcal{R} = \{m : m(y|x) = \frac{r(y,x)}{Z(x)}\},$$

where

$$r(y,x) = \prod_{i=1}^{k} \alpha_i^{g_i(x,y)}, \qquad Z(x) = \sum_y r(y,x). \qquad (3)$$

and the α's are nonnegative real numbers. Sometimes, we use $\lambda_i = g_i \cdot ln(\alpha_i)$ instead of g_i in (3).

The intersection of $\mathcal{P}$ and $\mathcal{R}$ contains the maximum entropy distribution $p^*(x,y)$ and is unique [2]. Therefore, we only need to find a single exponential model $m^*(y|x)$ in $\mathcal{R}$ that satisfies a set of linear constraints $\mathcal{P}$.

The generalized iterative scaling (GIS) algorithm [2] and its improved version [3] called IIS provide an iterative procedure to estimate the α's in an ME model as

$$\alpha_i^{(n+1)}(x,y) = \alpha_i^{(n)}(x,y) \cdot \frac{a_i}{m[g_i(x,y)]} \qquad (4)$$

where $m[g_i(x,y)]$ is the expectation of $g_i(x,y)$. Starting with arbitrary α's, usually initialized as unity, and using these algorithms, we can obtain the model p^*.

2.2. Training an ME model

We need to estimate the expectations of all features based on:

$$m[g_i(x,y)] = \sum_{\langle x',y'\rangle : g_i(x',y')=1} P(x') \cdot m(y'|x') \cdot g_i(x,y). \qquad (5)$$

We also need to calculate the normalization factor Z for each history according to (3). It can be proved that the estimation of all feature expectations $m[g_i(x,y)]$ takes the same amount of time as the calculation of all normalization factors Z. Therefore, we only show how to reduce the computation for the denominator Z. The simplification for computing the feature expectation is similar.

In conventional training algorithms, features in G are partitioned into two subsets: the marginal-feature subset G_m and the conditional-feature subset G_c. The vocabulary Y is split into subsets Y_m and $Y - Y_m$ where

$$Y_m = \{y : g_i(y) = 1 \text{ for some } g_i \in G_m\}$$

contains all y's that have some marginal constraints activated. Y is also split into subsets Y_x and $Y - Y_x$ as

$$Y_x = \{y : g_i(x,y) = 1 \text{ for some } g_i \in G_c\},$$

i.e., Y_x contains all y's that have a conditional constraint in a given context x. The calculation of $Z(x)$ is simplified as follows.

$$Z(x) = |Y - Y_m| + \sum_{y \in Y_m} r(y) + \sum_{y \in Y_x} (r(y,x) - r(y)) \qquad (6)$$

where $r(y)$ is the product of marginal constraints. The total computation for all $Z(x)$'s is $O(|X| \cdot \bar{C} + |Y|)$, where $\bar{C}$ is the average size of Y_x — the first term in (6) is a constant, the second term can be precalculated for all x's in $O(|Y|)$ time, and the third term requires $O(|X| \cdot \bar{C})$ time for all x. Typically $\bar{C}$ is two orders of magnitude smaller than the vocabulary size $|Y|$, resulting in a considerable computational speed-up relative to $O(|X| \cdot |Y|)$.

This implementation, however, is still impractical for a large corpus, especially when constraints other than N-grams are considered, because the number of distinct histories increases by several orders of magnitude and $\bar{C}$ becomes extremely large. Obviously, the most computationally expensive part in (6) is in the third term. We reduce computation by simplifying the calculation of this term.

3. HIERARCHICAL TRAINING METHOD

In a typical ME model, some features, such as the unigram feature $\lambda_i(w)$, the bigram feature $\lambda_j(w_{-1}, w)$ and the trigram feature $\lambda_k(w_{-2}, w_{-1}, w)$ are nested, while other features, such as the syntactic feature $\lambda(h_{-1}, w)$ and the bigram feature $\lambda(w_{-1}, w)$, are not. First we show how to train a model with only hierarchically nested features using the example of a trigram model. Then we show how to train a model with non-nested features using the example of a model with two kinds of overlapping bigram constraints. This work extends the core ideas outlined in [6] to take decisive advantage of the specific structure of N-gram models.

3.1. Training a Model with Only Nested Features

In computing the denominators in a standard trigram ME model

$$P(w|w_{-2},w_{-1}) = \frac{\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} \cdot \alpha_k^{g_k(w_{-2},w_{-1},w)}}{Z(w_{-2},w_{-1})}, \quad (7)$$

the third term in (6) is

$$\sum_{w \in Y_x} (\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} \cdot \alpha_k^{g_k(w_{-2},w_{-1},w)} - \alpha_i^{g_i(w)}). \quad (8)$$

Notice that for most of the word in Y_x, only bigram features are activated. We take advantage of this fact and split the word set Y_x into Y_b and $Y_x - Y_b$ where

$$Y_b = \{w : g_j(w_{-1},w) = 1 \text{ for some bigram feature } g_j\}$$

and also into Y_t and $Y_x - Y_t$ where

$$Y_t = \{w : g_k(w_{-2},w_{-1},w) = 1 \text{ for some trigram feature } g_k\}$$

After some algebraic manipulations, (8) can be rewrite as

$$\begin{aligned} &\sum_{w \in Y_h} (\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} \cdot \alpha_k^{g_k(w_{-2},w_{-1},w)} - \alpha_i^{g_i(w)}) \quad (9) \\ &= \sum_{w \in Y_b} (\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} - \alpha_i^{g_i(w)}) \\ &+ \sum_{w \in Y_t} (\alpha_i^{g_i(w)} \alpha_j^{g_j(w_{-1},w)} \alpha_k^{g_k(w_{-2},w_{-1},w)} - \alpha_i^{g_i(w)} \alpha_j^{g_j(w_{-1},w)}) \end{aligned}$$

The sum over Y_b is common to all histories x sharing a suffix and need to be computed only once. The complexity is $O(B)$ for all histories. In practice, Y_t has few words which have trigram constraints in any history, and so the sum over Y_t can be evaluated in $O(T)$ time for all histories. Therefore our method reduces the overall complexity to $O(U+B+T)$ which is the same as that of training a back-off trigram model.

To train an N-gram model in general, we can hierarchically group histories according to their suffixes. The computation for Z is also split into two parts. The first part is fixed for the histories with the same suffix and need be computed only once. The second part can be recursively computed by applying the scheme in (9) N-1 times. The overall complexity is in the order of the number of all unigrams, bigrams, ... , N-grams, and is bounded by $O(N \cdot L)$ since the number of distinct n-grams of any order $n < N$ is less than the size of the training data.

3.2. Models with Non-Nested Features

Consider the ME model with with both the bigram constraints of the syntactic head-words and the conventional bigram constraints:

$$P(w|h_{-1},w_{-1}) = \frac{\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} \cdot \alpha_k^{g_k(h_{-1},w)}}{Z(h_{-1},w_{-1})}. \quad (10)$$

The third summation in (6) becomes

$$\sum_{w \in Y_x} \alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} \cdot \alpha_k^{g_k(h_{-1},w)} - \alpha_i^{g_i(w)}. \quad (11)$$

We split Y_x into two complementary sets Y_w and $Y_x - Y_w$ where

$$Y_w = \{w : g_j(w_{-1},w) = 1 \text{for some bigram features } g_j\},$$

and also into sets Y_h and $Y_x - Y_h$, where

$$Y_h = \{w : g_k(h_{-1},w) = 1 \text{for some syntactic features } g_k\}$$

Then this summation in (11) can be estimated as

$$\begin{aligned} &\sum_{w \in Y_x} \alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} \cdot \alpha_k^{g_k(h_{-1},w)} - \alpha_i^{g_i(w)} \\ &= \sum_{w \in Y_w \cap Y_h} (\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} \cdot \alpha_k^{g_k(h_{-1},w)} \quad (12) \\ &\quad -\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} - \alpha_i^{g_i(w)} \cdot \alpha_k^{g_k(h_{-1},w)} + \alpha_i^{g_i(w)}) \\ &\quad + \sum_{w \in Y_w} (\alpha_i^{g_i(w)} \cdot \alpha_j^{g_j(w_{-1},w)} - \alpha_i^{g_i(w)}) \\ &\quad + \sum_{w \in Y_h} (\alpha_i^{g_i(w)} \cdot \alpha_k^{g_k(h_{-1},w)} - \alpha_i^{g_i(w)}) \end{aligned}$$

The last two summations in (13) require the time of $O(B)$ and $O(B_h)$ to compute respectively, where B_h is the number of syntactic bigrams. The complexity of calculating the first summation is difficult to estimate in theory, but it is much less than the complexity of using (11) directly in practice, because only a small number of words have both kinds of bigram constraints for each history. The computation necessary for training a model with both bigram and syntactic bigram constraints is reduced by two orders of magnitude in Switchboard, and its actual complexity is still bounded by $O(L)$.

3.3. General Cases

The above two schemes can be extended to solve the problem for training any ME model. All features can be assigned to one of M groups, and in each group, features are nested with the highest order no more than N. We call this kind of model an MN-model. We define the compounds of features with the same order as super-features. For instance, conventional bigram features g_j and head word bigram features g_k are combined to generate super bigram features f_{jk} where $f_{jk}(w_{-1},h_{-1},w) = 1$ iff $g_j(w_{-1},w) = 1$ or $g_k(h_{-1},w) = 1$. All super-features are now nested features. We apply both the method in Section 3.1 and that in 3.2 simultaneously in training an MN-model and may achieve a computation complexity of $O(N \cdot 2^M \cdot L)$ in practice.

A simplified training method similar to that in Section 3.2 has been proposed in [6]. However, this method does not take advantage of nested-features, and therefore results in less efficiency compared to ours. The complexity for training an N-gram model is $O(2^N \cdot L)$ instead of $O(N \cdot L)$. This method is of little use in training a complicated model as (1) in which the number of interacting constraints is large.

4. EXPERIMENTAL RESULTS

4.1. Nominal Speed-Up

From the previous section we can see that the training time is in proportion to the total number of terms to be summed up for all Z's. We count this number for both the old method and the new one and estimate the running time of a training method according to this number. The nominal speed-up is defined as:

$$\text{speed-up} = \frac{\text{\# of terms in the old method}}{\text{\# of terms in the new method}}.$$

The speed-up for training a trigram model and a 4-gram model is estimated based on the statistics from three corpora: Switchboard, a 14M-word subset of BN corpus and the entire 130M-word BN corpus (Table 4.1). Column 2 shows the tasks with their size. Column 3 are the numbers of terms in the old method and Column 4 are for the new method respectively. Column 5 shows the nominal speed-up. One can see that: 1. the nominal speed-up is more than 2 orders of magnitude. 2. when the size of the training set increases, the training time of the old method increases almost in square, 3. the training time of the new method however is bounded by the size of the training data and therefore only increases slightly as the train set increases. 4. the speed-up is greater as the size of the task increases, and 5. the speed-up is greater for higher order language models (4-gram vs trigram). Therefore, our hierarchical training method will be very efficient for training large language models. Table 4.1 shows the

Model	Corpus(size)	IIS	New	Speed-up
3-gram	Swbd(2.1M)	1.6e8	9e6	2e2
	BN(14M)	2.7e9	6.6e7	4e2
	BN(130M)	2.4e10	4.3e7	5.6e2
4-gram	Swbd(2.1M)	3.2e9	2.1e6	1.6e3
	BN(14M)	1.0e11	2.1e7	7e3
	BN(130M)	3.4e12	9.4e7	3.6e4

Table 1: Nominal Speed-Up for N-gram Models

nominal speed-up for training the language model (1) with non-nested features for the Switchboard. The computational complexity is also nominal to reduce in 2 orders of magnitude using the new algorithm.

IIS	New	Speed-up
6.8e9	7.5e7	9e1

Table 2: Nominal Speed-up for the Composite Model

4.2. Actual Observed Training Time

We train the trigram model and the composite model (1) for the Switchboard using both the conventional and the new methods. The following table shows the actual training times. The speed-up is about 30 folds training the trigram model, and is about 15 folds for training the composite language model. The actual speed-up is less than the nominal one because the program is more complicated for the new method than that for the old one.

Model	Old Method	New Method	Speed-up
Trigram	2	0.06	30
Composite	150	9.5	15

Table 3: Training Time per Iteration (CPU-Hour)

4.3. The Performance of the Composite ME model

The composite ME model (1) is trained for the Switchboard task. It out-performs models using any subset of its constraints, as expected. Table 4 shows the performance of the language models using different sources of information. Overall, the composite language model (Row 7) achieves a perplexity reduction by 13% and a WER reduction by 1.5% absolute compared to the results of the trigram model (Row 1).

No.	Model Constraints	Perplexity	WER
1	3-gram	79.0	38.5%
2	3-gram + NT	75.1	37.8%
3	3-gram +HW	74.5	37.7%
4	3-gram +Topic	73.5	37.8%
5	3-gram +HW + NT	74.0	37.5%
6	3-gram +Topic + HW	69.9	37.2%
7	3-gram +Topic + HW+NT	67.9	37.0%

Table 4: Performance of ME Language Models with N-gram, Headword (HW), Nonterminal Label (NT) and Topic Dependencies

5. REFERENCES

1. C. Chelba and F. Jelinek, "Exploiting Syntactic Structure for Language Modeling", *Proceedings of COLING-ACL'98*, pp. 225-231, Aug. 1998.
2. I. Csiszár, "Why Least Squares and Maximum Entropy? An Axiomatic Approach to Inference for Linear Inverse Problems", *The Annals of Statistics*, Vol. 19, No 4, Dec. 1991.
3. S. Della Pietra, V. Della Pietra and J. Lafferty, "Inducing Features of Random Fields", *IEEE Trans. on PAMI*, 19(4), pp280-393, April 1997.
4. J. Godfrey, E. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development", *Proc. of ICASSP'92*, pp 517-520, 1992.
5. S. Khudanpur and J. Wu, "A Maximum Entropy Language Model to Integrate N-Grams and Topic Dependencies for Conversational Speech Recognition", *Proceedings of ICASSP'99*, pp. 553-556, 1999
6. J. Lafferty and B. Suhm. "Cluster expansions and iterative scaling for maximum entropy language models", in *Maximum Entropy and Bayesian Methods*, Kluwer Academic Publishers, 1996.
7. J. Wu and S. Khudanpur, "Combining Nonlocal, Syntactic and N-Gram Dependencies in Language Modeling", *Proceedings of Eurospeech'99*, vol 5, pp2179-2182, Sep. 1999.
8. R. Iyer and M. Ostendorf, "Modeling Long Range Dependence in Language," *Proceedings of ICSLP'96*, pp 236-239, Oct. 1996.

EFFICIENT TRAINING METHODS FOR MAXIMUM ENTROPY LANGUAGE MODELING

Jun Wu and Sanjeev Khudanpur [1]
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218-2686, USA
{junwu,sanjeev}@mail.clsp.jhu.edu

Maximum entropy language modeling techniques combine different sources of statistical dependence, such as syntactic relationships, topic cohesiveness and collocation frequency, in a unified and effective language model. These techniques however are also computationally very intensive, particularly during model estimation, compared to the more prevalent alternative of interpolating several simple models, each capturing one type of dependency. In this paper we present ways which significantly reduce this complexity by reorganizing the required computations. We show that in case of a model with N-gram constraints, each iteration of the parameter estimation algorithm requires the same amount of computation as estimating a comparable back-off N-gram model. In general, the computational cost of each iteration in model estimation is linear in the number of distinct "histories" seen in the training corpus, times a model-class dependent factor. The reorganization focuses mainly on reducing this multiplicative factor from the size of the vocabulary to the average number of words seen following a history. A 15-fold speed-up has been observed by using this method in estimating a language model that incorporates syntactic head-word constraints, nonterminal-label constraints and topic-unigram constraints with N-grams for the Switchboard corpus. This model achieves a perplexity reduction of 13% and a word error rate reduction of 1.5% absolute compared to a trigram model.