DISTANT BIGRAM LANGUAGE MODELLING USING MAXIMUM ENTROPY

M. Simons, H. Ney, S.C. Martin

Lehrstuhl für Informatik VI, RWTH Aachen – University of Technology,
D-52056 Aachen, Germany

ABSTRACT

In this paper, we apply the maximum entropy approach to so-called distant bigram language modelling. In addition to the usual unigram and bigram dependencies, we use distant bigram dependencies, where the immediate predecessor word of the word position under consideration is skipped. The contributions of this paper are:
(1) We analyze the computational complexity of the resulting training algorithm, i.e. the generalized iterative scaling (GIS) algorithm, and study the details of its implementation. (2) We describe a method for handling unseen events in the maximum entropy approach; this is achieved by discounting the frequencies of observed events. (3) We study the effect of this discounting operation on the convergence of the GIS algorithm. (4) We give experimental perplexity results for a corpus from the WSJ task. By using the maximum entropy approach and the distant bigram dependencies, we are able to reduce the perplexity from 205.4 for our best conventional bigram model to 169.5.

1. INTRODUCTION

The advantage of using the maximum entropy principle in language modelling is that the principle provides a well defined method for incorporating different types of dependencies into a language model. There have been a number of papers on the application of the maximum entropy principle to natural language processing [1, 4, 6, 7, 10]. In this paper, we use the maximum entropy approach in the context of distant bigram dependencies and study a couple of questions that arise in the practical use of the maximum entropy method:

- How to implement the GIS training algorithm in a suitable way ?
- How to handle unseen events ?
- How is the convergence of the GIS training algorithm affected if the convergence requirements are not exactly satisfied ?
- How much improvement can we obtain over a conventional bigram model ?

2. MAXIMUM ENTROPY APPROACH

2.1. The Log-Linear Model

The starting point for the maximum entropy approach is to consider certain types of dependencies, so-called features, e.g. specific word bigrams or auto-cooccurrences (cache effect) as in conventional language modelling. For a given word history h and a word w under consideration, we define a feature function for each feature i:

$$f_i(h,w) \in \{0,1\} \quad .$$

For each of these features, the assumption is that we know its frequency. Then the maximum entropy principle [1, 4] tells us that the most general distribution that satisfies these constraints as expressed by the corresponding frequencies has the following functional form [2, pp. 83-87]:

$$p_\Lambda(w|h) = \frac{\exp\left[\sum_i \lambda_i f_i(h,w)\right]}{\sum_{w'} \exp\left[\sum_i \lambda_i f_i(h,w')\right]} \quad ,$$

where for each feature i we have a parameter λ_i and where we define: $\Lambda = \{\lambda_i\}$.

The important result of the maximum entropy principle is that the resulting model has a log–linear or exponential functional form. In the statistical terminology, the underlying sampling approach is referred to as a multinomial one [2, pp. 62-64]. An important difference, however, is that we are considering *conditional* probabilities.

2.2. The Maximum Likelihood Criterion

We consider the log-likelihood function $G(\Lambda)$ for a training corpus of running words $w_1, ..., w_n, ..., w_N$:

$$G(\Lambda) \quad := \quad \sum_{n=1}^{N} \log\, p_\Lambda(w_n|h_n) = \sum_{h\,w} N(h,w) \log\, p_\Lambda(w|h)$$

with the usual count definitions $N(h,w)$. To find the optimal set of parameters λ_i for maximum likelihood, or what is equivalent, minimum perplexity, we take the partial derivatives with respect to each of the parameters λ_i and set them to zero:

$$\frac{\partial G}{\partial \lambda_i} \quad = \quad \sum_{h\,w} N(h,w) \frac{\partial}{\partial \lambda_i} \log\, p_\Lambda(w|h) \;=\; 0 \quad .$$

After some elementary manipulations, we obtain:

$$\frac{\partial G}{\partial \lambda_i} = -\, Q_i(\Lambda) \;+\; N_i = 0$$

with the Λ dependent auxiliary function $Q_i(\Lambda)$:

$$Q_i(\Lambda) := \sum_{h\,w} N(h)\, p_\Lambda(w|h)\, f_i(h,w)$$

and with the Λ independent feature counts N_i:

$$N_i := \sum_{hw} N(h,w)\, f_i(h,w) \quad .$$

3. GIS ALGORITHM

3.1. Principle

The GIS (=generalized iterative scaling) algorithm is a well known method in statistics for finding the numerical solution of the maximum likelihood equations [3, 4]. It is based on the condition that the number of active features for each pair (h,w) with $N(h) > 0$ is constant, i.e.

$$\sum_i f_i(h,w) = F = const(h,w) \quad .$$

Later we will show how we satisfy this condition and how we combine it with the handling of unseen events, i.e. features not seen in training. The GIS algorithm results in the following iterative update formula:

$$\begin{aligned} \lambda_i' &:= \lambda_i + \Delta\lambda_i \quad \text{with} \\ \Delta\lambda_i &= \frac{1}{F} \log \frac{N_i}{Q_i(\Lambda)} \quad . \end{aligned}$$

3.2. Implementation

For implementation reasons, we rewrite $Q_i(\Lambda)$ as follows:

$$Q_i(\Lambda) = \sum_{n=1}^{N} \sum_{w} p_\Lambda(w|h_n)\, f_i(h_n,w) \quad .$$

This equation allows us to pass over the corpus positions n (each with history h_n) rather than over the set of seen histories h (each with count $N(h)$). The advantage of this modification is that we do not need special data structures, which would make the implementation effort more costly, and that we are more flexible with respect to any desired type of features i.

In addition, for efficiency reasons, we define the set $I(h,w)$ of features that is activated by a given pair (h,w):

$$I(h,w) := \{i : f_i(h,w) = 1\} \quad .$$

By definition, we must have: $|I(h,w)| = F$. To compute the conditional probability $p_\Lambda(w|h_n)$, we need to calculate the denominator which is denoted by $Z_\Lambda(h_n)$:

$$\begin{aligned} Z_\Lambda(h_n) &:= \sum_w \exp\Big[\sum_i \lambda_i f_i(h_n,w)\Big] \\ &= \sum_w \exp\Big[\sum_{i \in I(h_n,w)} \lambda_i f_i(h_n,w)\Big] \\ &= \sum_w \exp\Big[\sum_{i \in I(h_n,w)} \lambda_i\Big] \quad . \end{aligned}$$

Thus we have limited the computational effort to the active features $i \in I(h_n,w)$ for a pair (h_n,w) under consideration. The same concept can be used when computing the auxiliary function $Q_i(\Lambda)$. Putting all these concepts together, we obtain an implementation of the GIS algorithm as shown in Table 1. To measure the resulting complexity per iteration, we count the number of λ_i additions in the exponential function. It is easy to see that this number is

$$2 \cdot N \cdot W \cdot F \quad ,$$

where W is the vocabulary size. This is the complexity for a general language model. In the case of a pure bigram language model, there are only W possible histories h, and in each iteration, we can have a loop over the histories h rather than the positions n. Thus we would obtain a complexity of $2 \cdot W^2 \cdot F$, which is a big improvement for a database if $N \gg W$.

Table 1. Implementation of the GIS algorithm.

compute (discounted) feature counts N_i
choose initial values for $\Lambda = \{\lambda_i\}$
for each iteration do
initialize: $G(\Lambda) = 0$
initialize: $Q_i(\Lambda) = 0$ for each feature i
for each position $n = 1 \ldots N$ do
calculate denominator $Z_\Lambda(h_n)$
accumulate for perplexity: $G(\Lambda) = G(\Lambda) + \log\ p_\Lambda(w_n\|h_n)$
for each word w of vocabulary do
for each feature $i \in I(h_n,w)$ do
accumulate $Q_i(\Lambda)$: $Q_i(\Lambda) := Q_i(\Lambda) + p_\Lambda(w\|h_n)$
update λ_i for each feature i

4. THE FEATURES

4.1. Definition of Features

For each possible pair (h,w), where the history h is defined by a sequence of predecessor words $v_1 \ldots v_m$, i.e. $h := v_1 \ldots v_m = v_1^m$, we introduce the following features, no matter whether they occurred in the training data or not:

Unigram features (U):

$$f_b^U(v_1^m, w) = \begin{cases} 1 & \text{if } w = b \\ 0 & \text{otherwise} \end{cases}$$

Bigram features (B):

$$f_{ab}^B(v_1^m, w) = \begin{cases} 1 & \text{if } w = b \text{ and } v_m = a \\ 0 & \text{otherwise} \end{cases}$$

Distant bigram features (D):

$$f_{ab}^D(v_1^m, w) = \begin{cases} 1 & \text{if } w = b \text{ and } v_{m-1} = a \\ 0 & \text{otherwise} \end{cases}$$

Using these three types of features, there are exactly 3 features active for each pair (h,w), namely 1 unigram feature, 1 bigram feature and 1 distant bigram feature. As a result, we have the following functional form for our model for a word w with the two predecessor words (u,v):

$$p(w|u,v) = \frac{\exp[\lambda_D(u,w) + \lambda_B(v,w) + \lambda_U(w)]}{\sum_{w'} \exp[\lambda_D(u,w') + \lambda_B(v,w') + \lambda_U(w')]} \quad ,$$

where the parameters $\lambda_D(u,w), \lambda_B(v,w), \lambda_U(w)$ are to be trained by the GIS algorithm.

4.2. Handling of Unseen Features

For the features i that were not seen in the training data, the counts N_i are zero. To gain count mass for them, we discount the counts of the seen features in the spirit of the Turing-Good formula. We use the method of absolute discounting [8]:

$$N_i = \sum_{hw} N(h,w)\, f_i(h,w) \; - d_i \quad ,$$

where we use only three independent discounting parameters d_i, namely one for each of the three feature sets (e.g. $d_U = 0.29, d_B = 0.70, d_D = 0.74$).

Similarly, for each of the three feature types, we pool the parameter λ_i of unseen events. Thus there are only three independent parameters λ_i for unseen features: $\lambda_{\overline{U}}$, $\lambda_{\overline{B}}$, $\lambda_{\overline{D}}$. As an example, we consider the bigram features. For all unseen bigrams (ab), we introduce a single general parameter:

$$N(a,b) = 0: \quad \lambda^B_{ab} = \lambda_{\overline{B}} \quad .$$

To compute the parameter $\lambda_{\overline{B}}$ in the GIS algorithm, we need the corresponding count and the corresponding auxiliary function $Q_{\overline{B}}(\Lambda)$. The feature count of this complementary feature is computed from the count mass that is gained by the discounting operation. Denoting the set of bigram features by the symbol I_B (including the *unseen* features), we compute the auxiliary function $Q_{\overline{B}}(\Lambda)$ from the normalization constraint:

$$\begin{aligned}
\sum_{i \in I_B} Q_i(\Lambda) &= \sum_{i \in I_B} \sum_n \sum_w p_\Lambda(w|h_n)\, f_i(h_n,w) \\
&= \sum_n \sum_w p_\Lambda(w|h_n) \sum_{i \in I_B} f_i(h_n,w) \\
&= \sum_n \sum_w p_\Lambda(w|h_n) \\
&= \sum_n 1 = N \quad .
\end{aligned}$$

It should be noted that the above normalization constraint is a direct consequence of the specific way of defining the features. The advantage of this approach is that we are able to directly use the discounting parameter obtained by leaving-one-out for each feature type.

5. EXPERIMENTAL RESULTS

5.1. Database and Task

For the experiments, a 4.5-million word text from the Wall Street Journal task was used (exact size: 4,472,827 words). The vocabulary consisted of approximately 20,000 words (`vocab20o.nvp`). All other words in the text were replaced by the label `<UNK>` for the unknown word. The test set perplexity was calculated on a separate text of 325,000 words. In the perplexity calculations, the unknown word was included. The corpora used are the same as in [9] and [10]. Table 2 shows the number of different feature constraints used for each kind of feature. A single iteration for the full language model required about 190 CPU hours on the workstation used (SGI with an R4600 processor).

Table 2. Number of distinct features observed in the training corpus (4.5 million words).

Type	Features
Unigram	19,725
Bigram	875,497
Distant Bigram	1,218,320
Trigram	2,370,914

Table 3. Perplexities over the iterations for the unigram/bigram features without (a) and with (b) distant bigram features.

	Iteration	Training	Test
a	1	439.1	470.2
	2	182.6	260.0
	3	138.2	225.3
	4	125.2	217.5
	5	120.4	215.2
	6	118.6	214.2
	7	117.4	213.5
	8	117.0	212.8
	9	116.8	212.3
	10	116.7	211.8
b	0	116.7	211.8
	1	86.2	186.5
	2	72.7	176.3
	3	65.8	171.9
	4	62.0	170.2
	5	59.8	169.5

5.2. Convergence of the GIS algorithm

We tested how the perplexity goes down with the iterations of the GIS algorithm. Table 3 shows the perplexity as a function of the iterations for both the training and the test set. The parameters of the conventional bigram model were used to initialize the distant bigram model. We can see that the models being trained have almost reached the optimum after $5-10$ iterations. We make the following remarks:

1. The optimization criterion applies to the *training* set; therefore there is no guarantee for convergence on the *test* set.
2. Even when considering the training set, we must take into account that the assumptions of the theoretic framework do not apply any more because we use discounted feature counts. As a result, the perplexity may even go up again on the *training* set, which however does *not* happen in Table 3.
3. The important result is that the discounting of the feature counts does not hurt the practical use of the GIS algorithm.

5.3. Perplexity Results

Table 4 shows a comparison of the test set perplexities for various types of language models each of which is based on the same set of the three feature types considered so far. As an alternative to the maximum entropy approach, a discounting approach was used also in the experiment (absolute discounting with interpolation, see [8, 9]). Typically there is a significant improvement if the conventional unigram distribution is replaced by what we refer to as singleton unigram distribution [5]. Therefore, this variant

Table 4. Effect of the distant bigram features on the test set perplexity: a) discounting model (SU = singleton unigram distribution), b) maximum entropy model.

Distant Bigram Features	no	yes
Discounting Model	214.7	198.8
Discounting Model with SU	205.4	193.4
Maximum Entropy Model	211.8	169.5

Table 5. Test set perplexities (PP) for the linear interpolation of the discounting model (with trigram/bigram/unigram features; SU = singleton unigram distribution) and the maximum entropy model (with unigram/bigram/distant bigram features)

Model	PP
Discounting Model (with SU)	152.9
Maximum Entropy Model	169.5
Lin. Interpolation of Both Models	144.0

is included in Table 4. For the discounting approach, the distant bigram features were incorporated by linearly interpolating the corresponding relative frequencies with the baseline discounting model. From Table 4, we see that the maximum entropy approach does not make much difference as long as *no* distant bigram features are used. However, when these features are added, there is a significant difference in the perplexities for the discounting model and the maximum entropy model: the maximum entropy model reduces the perplexity down to 169.5, which is an improvement by 12.4 % over the best discounting model with a perplexity of 193.4

In addition, we carried out another experiment in which we combined the maximum entropy model with a discounting model. Unlike the previous experiment, here the discounting model included trigram features. Table 5 shows the perplexities for each of the two models and for a linear interpolation of them. As can be seen, the perplexity of the trigram discounting model could be decreased from 152.9 to 144.0, which is an improvement by 6.2 %.

5.4. Comparison: Log-linear vs. Additive Model

In [8], the point of view was emphasized that there are many functional forms for expressing the statistical dependencies in language modelling. In this view, the log–linear form is simply one out of many possible forms. Therefore, it is instructive to compare the functional form of the log–linear model considered so far with that of an additive model, which is more commonly used.

Using the interpolation parameters μ_D and μ_B, we have the equation of the additive model:

$$\begin{aligned}\tilde{p}(w|u,v) &= \mu_D\, p_D(w|u) + \mu_B\, p_B(w|v) \\ &\quad + [1-\mu_D-\mu_B]\, p_U(w)\end{aligned}$$

with the probability distributions $p_D(w|u), p_B(w|v)$ and $p_U(w)$ for the distant bigram, the bigram and the unigram features, respectively. This additive model has the same number of free parameters as the log–linear model. For a fair comparison with the log–linear model, all these three distributions must be trained *jointly* using the EM algorithm. Although the use of the singleton unigram distribution is a first-step into this direction, a systematic comparison has not been done yet.

6. CONCLUSION

In this paper, we have studied some aspects of the maximum entropy method in the context of language modelling. We have presented a method for handling unseen features and discounting the feature counts. The effect of this method on the convergence of the GIS algorithm has been studied experimentally. In addition to the usual unigram and bigram features, we have included the distant bigram features. The resulting maximum entropy language model could significantly reduce the perplexity of the conventional discounting approach.

REFERENCES

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

[2] Y. M. M. Bishop, S. E. Fienberg, P. W. Holland: *Discrete Multivariate Analysis.* MIT press, Cambridge, MA, 1975.

[3] J. N. Darroch, D. Ratcliff: Generalized Iterative Scaling for Log–Linear Models. *Annals of Mathematical Statistics*, Vol. 43, pp. 1470–1480, 1972.

[4] S. Della Pietra, V. Della Pietra, J. Lafferty: Inducing Features of Random Fields. Technical Report CMU-CS-95-144, Carnegie Mellon University, Pittsburgh, PA, 1995.

[5] R. Kneser, H. Ney: Improved Backing-Off for m–gram Language Modeling. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Detroit, MI, Vol. I, pp. 49-52, May 1995.

[6] J. D. Lafferty, B. Suhm: Cluster Expansion and Iterative Scaling for Maximum Entropy Language Models. In K. Hanson, R. Silver (eds.): *Maximum Entropy and Bayesian Methods*, Kluwer Academic Publishers, 1995.

[7] R. Lau, R. Rosenfeld, S. Roukos: Trigger-Based Language Models: A Maximum Entropy Approach. Proc. IEEE Inter. Conf. on Acoustics, Speech and Signal Processing, Minneapolis, MN, Vol. II, pp. 45-48, April 1993.

[8] H. Ney, U. Essen, R. Kneser: On Structuring Probabilistic Dependences in Stochastic Language Modelling. *Computer Speech and Language*, Vol. 8, pp. 1-38, 1994.

[9] H. Ney, F. Wessel, S. Martin: Statistical Language Modeling Using Leaving-One-Out. In S. Young, G. Bloothooft (eds.): *Corpus-Based Methods in Speech and Language*, pp. 174-207, Kluwer Academic Publishers, in press, 1996/97.

[10] R. Rosenfeld: *Adaptive Statistical Language Modeling: A Maximum Entropy Approach.* Ph.D. Thesis, Technical Document CMU–CS–94–138, Carnegie Mellon University, Pittsburgh, PA, 1994.