EFFICIENT SAMPLING AND FEATURE SELECTION IN WHOLE SENTENCE MAXIMUM ENTROPY LANGUAGE MODELS

Stanley F. Chen and Ronald Rosenfeld

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
{sfc,roni}@cs.cmu.edu

ABSTRACT

Conditional Maximum Entropy models have been successfully applied to estimating language model probabilities of the form $p(w|h)$, but are often too demanding computationally. Furthermore, the conditional framework does not lend itself to expressing global sentential phenomena. We have recently introduced a non-conditional Maximum Entropy language model which directly models the probability of an entire sentence or utterance. The model treats each utterance as a "bag of features," where features are arbitrary computable properties of the sentence. Using the model is computationally straightforward since it does not require normalization. Training the model requires efficient sampling of sentences from an exponential distribution.

In this paper, we further develop the model and demonstrate its feasibility and power. We compare the efficiency of several sampling techniques, implement smoothing to accommodate rare features, and suggest an efficient algorithm for improving convergence rate. We then present a novel procedure for feature selection, which exploits discrepancies between the existing model and the training corpus. We demonstrate our ideas by constructing and analyzing competitive models in the Switchboard domain.

1. INTRODUCTION

1.1. Conditional language models

Conventional statistical language models estimate the probability of a sentence s by using the chain rule to decompose it into a product of conditional probabilities:

$$\Pr(s) \stackrel{\text{def}}{=} \Pr(w_1 \ldots w_n) \stackrel{\text{def}}{=} \prod_{i=1}^{n} \Pr(w_i|h_i) \qquad (1)$$

where $h_i \stackrel{\text{def}}{=} w_1 \cdots w_{i-1}$ is the *history* when predicting word w_i. The vast majority of work in statistical language modeling to date is thus devoted to estimating terms of the form $\Pr(w|h)$. While this practice is understandable from a historical perspective (N-gram modeling cannot be done on whole sentences), it is not always desirable. Global features of sentences, such as their length or grammaticality, are impossible or awkward to encode in a conditional framework. Also, external influences on the sentence (*e.g.*, the effect of preceding utterances, or dialog level variables) are equally hard to encode, and factoring them into the prediction of every word in the current sentence causes small but systematic biases in probability estimation to be compounded.

We are grateful to Larry Wasserman for many discussions and much advice on Markov Chain Monte Carlo sampling.

1.2. Conditional Maximum Entropy Models

In the last few years, Maximum Entropy (ME, [9]) models have been successfully used to estimate conditional language probabilities of the form $P(w|h)$ [6, 11, 1, 15] (as well as to model prepositional phrase attachment [14] and induce features of word spelling [5]).

In using Maximum Entropy to model $P(w|h)$, one major obstacle is the heavy computational requirements of training and using the model. These requirements are particularly severe because of the need to renormalize the model for each new value of h.

1.3. Whole Sentence Maximum Entropy Models

We have recently introduced a new Maximum Entropy language model which directly models the probability of an entire sentence or utterance [16]. The new model is conceptually simpler, as well as more naturally suited to modeling whole-sentence phenomena, than the conditional ME models proposed earlier. By avoiding the chain rule, the model treats each sentence or utterance as a "bag of features", where features are arbitrary computable properties of the sentence. Furthermore, the single, universal normalizing constant cannot be computed exactly,[1] but this does not interfere with training (done via sampling) or with use. Using the model is computationally straightforward. The feasibility of training the model depends crucially on efficient sampling of sentences from an exponential distribution.

In this paper, we further develop the model and demonstrate its usefulness in real domains. Section 2 reviews the whole-sentence Maximum Entropy model. Section 3 presents several sampling strategies, compares their relative efficiencies, and discusses step size selection and smoothing. In Section 4, we introduce a new procedure for feature selection, and illustrate its use by constructing models in the Switchboard domain and measuring their impact.

An expanded and updated version of this paper can be found at http://www.cs.cmu.edu/~sfc/wsme-icassp99.ps.

[1] As discussed later, this constant can be estimated through sampling.

2. OVERVIEW OF WHOLE SENTENCE MAXIMUM ENTROPY LANGUAGE MODELING

A whole sentence ME language model has the form:

$$P(s) = \frac{1}{Z} \cdot p_0(s) \cdot \exp(\sum_i \lambda_i f_i(s)) \qquad (2)$$

where the $\{\lambda_i\}$'s are the parameters of the model, Z is a universal normalization constant which depends only on the $\{\lambda_i\}$'s,[2] and the $\{f_i(s)\}$'s are arbitrary computable properties, or *features*, of the sentence s. The distribution $p_0(s)$ is an arbitrary factor that often plays the role of a prior.

The features $\{f_i(s)\}$ are selected by the modeler to capture those aspects of the data they consider appropriate or profitable. These can include conventional n-grams, longer-distance dependencies, global sentence properties, as well as more complex functions based on part-of-speech tagging, parsing, or other types of post-processing.

Next, for each selected feature $f_i(s)$, its expectation under $P(s)$ is constrained to a specific value K_i:

$$E_P f_i = K_i \quad . \qquad (3)$$

These target values are typically set to the expectation of that feature under the empirical distribution $\tilde{p}$ of some training corpus $\{s_1, \ldots, s_N\}$.[3] Then, the constraint becomes:

$$\sum_s P(s) \cdot f_i(s) = E_{\tilde{p}} f_i = \frac{1}{N} \sum_{j=1}^{N} f_i(s_j) \quad . \qquad (4)$$

If the constraints (3) are consistent, there exists a unique solution $\{\lambda_i\}$ within the exponential family (2) which satisfies them. Among all (not necessarily exponential) solutions to equations (3), the exponential solution is the one closest to the prior $p_0(s)$ (in the Kullback-Liebler sense), and is thus called the Minimum Divergence or Minimum Discrimination Information (MDI) solution. If the prior is flat, this becomes simply the Maximum Entropy solution. Furthermore, if the feature target values K_i are the empirical expectations over some training corpus (as in equations (4)), the MDI or ME solution is also the Maximum Likelihood solution of the exponential family. For more information, see [9, 1, 15].

The MDI or ME solution can be found by an iterative procedure such as the Generalized Iterative Scaling (GIS) algorithm [4]. GIS starts with arbitrary λ_i's. At each iteration, the algorithm improves the $\{\lambda_i\}$ values by comparing the expectation of each feature under the current P to the target value, and modifying the associated λ. In particular, we take

$$\lambda_i \leftarrow \lambda_i + F_i \log \frac{E_{\tilde{p}}[f_i]}{E_P[f_i]} \qquad (5)$$

where F_i is the step size (see Section 3.2).

In training a whole-sentence Maximum Entropy model, computing the expectations $E_P[f_i] = \sum_s P(s) \cdot f_i(s)$ requires a summation over all possible sentences s, a clearly infeasible task. Instead, we estimate $E_P[f_i]$ by sampling from the distribution $P(s)$ and using the sample expectation of f_i. Sampling from an exponential distribution is a non-trivial task, and will be discussed in the next section. Efficient sampling is crucial to successful training.

It is equally infeasible to compute the normalization constant $Z = \sum_s p_0(s) \cdot \exp(\sum_i \lambda_i f_i(s))$. Fortunately, this is not necessary for training: sampling can be done without knowing Z, as will be shown in the next section. Using the model as part of a classifier (*e.g.*, a speech recognizer) does not require knowledge of Z either, because the relative ranking of the different classes is not changed by a single, universal constant.[4] Notice that this is not the case for conditional Maximum Entropy models.

For more details, see [16].

3. MODEL TRAINING

3.1. Sampling

In this section, we describe several statistical sampling methods for estimating the values $E_P[f_i]$, and present results evaluating their relative efficacy.

DellaPietra *et al.* [5] build a joint ME model of word spelling. They use **Gibbs sampling** [8] to generate a set of word spellings $\{s_1, \ldots, s_M\}$, and estimate $E_P[f_i] \approx \frac{1}{M} \sum_{j=1}^{M} f_i(s_j)$.

Gibbs sampling is not efficient for sentence models, as the probability of a great many sentences must be computed to generate each sample. **Metropolis sampling** [12], is more appropriate for this situation. An initial sentence is chosen randomly. For each word position in turn, a new word is proposed to replace the original word in that position, and this change is *accepted* with some probability. After all word positions have been examined, the resulting sentence is added to the sample, and this process is repeated.[5] The distribution used to generate new word candidates for each position affects the sampling efficiency; we chose to use a unigram distribution.

Adapting the Metropolis algorithm to sentences of variable-length requires care. In one solution, we pad each sentence with end-of-sentence tokens </s> up to a fixed length l. A sentence becomes shorter if the last non-</s> token is changed to </s>, longer if the first </s> token is changed to something else.

In applying Metropolis sampling, instead of replacing a single word at a time it is possible to replace larger units. In particular, in **independence sampling** we consider replacing the whole sentence in each iteration. For efficiency, the distribution $Q(s)$ used to generate new sentence candidates must be similar to the distribution $P(s)$ we are attempting to sample from.

In **importance sampling**,[6] a sample $\{s_1, \ldots, s_M\}$ is generated according to some distribution $Q(s)$ (which similarly must be close to $P(s)$ for efficient sampling). Then, each sample s_j is counted $\frac{P(s_j)}{Q(s_j)}$ times, so that we have

$$E_P[f_i] \approx \frac{\sum_{j=1}^{M} \frac{P(s_j)}{Q(s_j)} f_i(s_j)}{\sum_{j=1}^{M} \frac{P(s_j)}{Q(s_j)}} \quad . \qquad (6)$$

[2] In statistical mechanics, Z is known as the *partition function*.

[3] For binary features, this is simply the prevalence of that feature in the corpus.

[4] Nonetheless, at times it may be desirable to approximate Z, perhaps in order to compute perplexity. This can be done to any desired accuracy by generating a large sample from $P(s)$ and observing the frequency of some frequent sentence s_0.

[5] The sampling procedure is still correct if the current sentence is added to the sample after *each* word position is examined; however, this process becomes less well-defined when we consider variable-length sentences.

[6] This was mistakingly dubbed *corrective sampling* in [16].

	sampling algorithm		
	Metropolis	independence	importance
$f_{1,4}$	0.38±0.07	0.438±0.001	0.439±0.001
$f_{5,8}$	0.10±0.02	0.1001±0.0004	0.1001±0.0006
$f_{9,12}$	0.08±0.01	0.0834±0.0006	0.0831±0.0006
$f_{13,16}$	0.073±0.008	0.0672±0.0005	0.0676±0.0007
$f_{17,\infty}$	0.37±0.09	0.311±0.001	0.310±0.002

Table 1: Mean and standard deviation (of mean) of feature expectation estimates for sentence-length features for various sampling algorithms over ten runs

Which sampling method is best depends on the nature of $P(s)$ and $Q(s)$. We evaluated these various methods on the models to be described in Section 4. These models employ a trigram model as the prior $p_0(s)$ and include relatively few features, so that the resulting model is rather similar to a trigram model. As it is possible to generate sentences from a trigram model efficiently, taking $Q(s)$ to be a trigram model for independence and importance sampling is very effective. To measure the effectiveness of different algorithms, for each algorithm we generated ten independent samples of the same length. We estimated the expectations of a set of features on each sample, and calculated the empirical variance in the estimate of these expectations over the ten samples. More efficient sampling algorithms should yield lower variances.

In our experiments, we found that independence sampling and importance sampling both yielded excellent performance, while word-based Metropolis sampling performed substantially worse. As an example, we estimated expectations for sentence-length features of the form

$$f_{l_1,l_2}(s) = \begin{cases} 1 & \text{if } l_1 \leq \text{length}(s) \leq l_2 \\ 0 & \text{otherwise} \end{cases}$$

over ten samples of size 100,000. In Table 1, we display the means and standard deviations of several feature expectations for several sampling algorithms.

The efficiency of importance and independence sampling depends on the distance between the generating distribution $Q(s)$ and the desired distribution $P(s)$. If $Q(s) = p_0(s)$ (the prior), that distance will grow with each training iteration. Once the distance becomes too large, Metropolis sampling can be used for one iteration, say iteration k, and the resulting sample retained. Subsequent iterations can "recycle" that sample using importance or independence sampling with $Q(s) = P^{[k]}(s)$.

3.2. Step Size

In GIS, the step size for feature update is inversely related to the number of active features. As sentences typically have many features, this may result in very slow convergence. Improved Iterative Scaling (IIS,[5]) uses a larger effective step size than GIS, but requires a great deal more bookkeeping.

However, when feature expectations are near their target value, IIS can be closely approximated with equation (5) where F_i is taken to be a weighted average of the feature sum over all sentences; *i.e.*, if the set of sentences s were finite, we would take

$$F_i = \frac{1}{\sum_s P(s) f_i(s)} \sum_s P(s) f_i(s) \sum_{i'} f_{i'}(s) \quad . \tag{7}$$

In our implementation, we approximated F_i by summing only over the sentences in the sample used to calculate expectations. This technique resulted in convergence in all of our experiments.

3.3. Smoothing

From equation (5) we can see that if $E_{\tilde{p}}[f_i] = 0$ then we will have $\lambda_i \to -\infty$. To *smooth* our model, we take the approach described by Berger and Miller [2]: We introduce a Gaussian prior on λ_i values and search for the maximum *a posterior* model instead of the maximum likelihood model.

4. FEATURE SELECTION

In this section, we discuss feature selection and model construction, using Switchboard as our example domain. Our training data consisted of three million words of Switchboard text. We constructed a trigram model on this data using a variation of Kneser-Ney smoothing [10], and used it as our prior $p_0(s)$. We employed features that constrained the frequency of word n-grams (up to n=4), distance-two word n-grams (up to n=3) [15], and class n-grams (up to n=5) [3]. That is, we considered features of the form

$$f_\alpha(s) = \text{\# of times } n\text{-gram } \alpha \text{ occurs in } s \quad .$$

We partitioned our vocabulary (of 15,000 words) into 100, 300, and 1000 classes using the word classing algorithm of Ney *et al.* [3, 13] on our training data.

To select specific features we devised the following procedure. First, we generated an artificial corpus by sampling from our prior trigram distribution $p_0(s)$. This "trigram corpus" was of the same size as the training corpus. For each n-gram, we compared its count in the "trigram corpus" to that in the training corpus. If these two counts differed significantly (using a χ^2 test), we added the corresponding feature to our model.[7] We tried thresholds on the χ^2 statistic of 500, 200, 100, 30, and 15, resulting in approximately 900, 3,000, 10,000, 20,000 and 52,000 n-gram features, respectively.

In Table 2, we display the n-grams with the highest χ^2 scores. The majority of these n-grams involve a 4-gram or 5-gram that occurs zero times in the training corpus and occurs many times in the trigram corpus. These are clear examples of longer-distance dependencies that are not modeled well with a trigram model. However, the last feature is a class unigram, and indicates that the trigram model overgenerates words from this class. On further examination, the class turned out to contain a large fraction of the rarest words. This indicates that perhaps the smoothing of the trigram model could be improved.

For each feature set, we trained the corresponding model by initializing all λ_i to 0. We used importance sampling to calculate expectations. However, instead of generating an entirely new sample for each iteration, we generated a single corpus from our prior trigram model, and re-weighted this corpus for each iteration using importance sampling.[8] We trained each of our feature sets for 50 iterations of iterative scaling; each complete training run took less than three hours on a 200 Mhz Pentium Pro computer.

We measured the impact of these features by rescoring speech recognition N-best lists ($N \leq 200$) which were generated by the

[7] N-grams with zero counts were considered to have 0.5 counts in this analysis.

[8] Admittedly, for rare features this often results in mutually inconsistent constraints.

feature	training corpus count	trigram corpus count	χ^2
TALKING TO YOU KNOW	0	148	43512.50
TALKING TO _ KNOW	0	148	43512.50
TALKING/CHATTING TO YOU KNOW	0	148	43512.50
NICE/HUMONGOUS TALKING/CHATTING TO YOU KNOW	0	60	7080.50
HOW ABOUT YOU KNOW	0	56	6160.50
HOW ABOUT _ KNOW	0	56	6160.50
<s> HAVE _ KNOW	0	42	3444.50
KIND OF A WHILE/SUDDEN	0	42	3444.50
VAGUELY/BLUNTLY	15389	22604	3382.69

Table 2: N-grams with largest discrepancy (according to χ^2 statistic) between training corpus and trigram-generated corpus of same length; n-grams with "_" token are distance-two n-grams; w_1/w_2 notation represents a class whose two most frequent members are w_1 and w_2

χ^2 threshhold	*baseline*	100	30	15
# features	0	3,500	19,000	52,000
WER	36.53	36.49	36.37	36.29
LM only	40.92	40.95	40.68	40.46
avg. rank	27.29	27.26	26.34	26.42
LM only	35.20	35.28	34.59	33.93

Table 3: Top-1 WER and average rank of best hypothesis using varying feature sets.

Janus system [7] on a Switchboard/CallHome test set of 8,300 words. The trigram $p_0(s)$ served as baseline. For each model, we computed both the top-1 word error rate and the average rank of the least errorful hypothesis. These figures were computed first by combining the new language scores with the existing acoustic scores, and again by considering the language scores only. Results are summarized in Table 3. While the specific features we selected here made only a small difference in N-best rescoring, they were nonetheless useful in demonstrating the extreme generality of our model: *Any* computable property of the sentence which is currently not adequately modeled can and should be added into the model.

5. DISCUSSION AND SUMMARY

Unlike conditional ME models, sentence-based ME models are efficient to use (because they do not require renormalization) and can naturally express sentence-level phenomena. In this paper, we described efficient algorithms for constructing sentence ME models, offering solutions to the questions of sampling, step size and smoothing. We also introduced a procedure for feature selection which seeks and exploits discrepancies between an existing model and the training corpus.

Given the framework and algorithms presented here, a language modeler can focus on *which* properties of language to model as opposed to *how* to model them. This framework can conveniently express arbitrary features and combines them in a theoretically elegant manner.

6. REFERENCES

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

[2] A. Berger and R. Miller. Just-in-time language modeling. In *ICASSP-98*, Seattle, Washington, 1998.

[3] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Comput. Linguistics*, 18(4):467–479, Dec. 1992.

[4] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Ann. of Math. Stat.*, 43:1470–1480, 1972.

[5] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3), 1997.

[6] S. Della Pietra, V. Della Pietra, R. Mercer, and S. Roukos. Adaptive language modeling using minimum discriminant estimation. In *Proc. of the Speech and Natural Language DARPA Workshop*, February 1992.

[7] M. Finke, J. Fritsch, P. Geutner, K. Ries, T. Zeppenfeld, and A. Waibel. The JanusRTk Switchboard/Callhome 1997 evaluation system. In *Proc. of LVCSR Hub 5-E Workshop*, Baltimore, May 1997.

[8] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

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

[10] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181–184, 1995.

[11] R. Lau, R. Rosenfeld, and S. Roukos. Trigger-based language models: A maximum entropy approach. In *Proc. of ICASSP-93*, pages II–45 – II–48, April 1993.

[12] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. of Chemical Physics*, 21:1087–1092, 1953.

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

[14] A. Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy approach for prepositional phrase attachment. In *Proc. of the ARPA Workshop on Human Language Technology*, March 1994.

[15] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech, and Language*, 10, 1996.

[16] R. Rosenfeld. A whole sentence maximum entropy language model. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.