RC23224 (W0405-149) May 26, 2004
Computer Science

IBM Research Report

Conditional Maximum Likelihood Estimation for Improving Annotation Performance of N-gram Models Incorporating Stochastic Finite State Grammars

Vaibhava Goel
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Conditional Maximum Likelihood Estimation for Improving Annotation Performance of N-gram Models Incorporating Stochastic Finite State Grammars

Vaibhava Goel

IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

vgoel@us.ibm.com

Abstract

Language models that combine stochastic grammars and N-grams are often used in speech recognition and language understanding systems. One useful aspect of these models is that they can be used to annotate phrases in the text with their constituent grammars; such annotation often plays an important role in subsequent processing of the text. In this paper we present an estimation procedure, under a conditional maximum likelihood objective, that aims at improving the annotation performance of these models over their maximum likelihood estimate. The estimation is carried out using the extended Baum-Welch procedure of Gopalakrishnan et.al. We find that with conditional maximum likelihood estimation the annotation accuracy of the language models can be improved by over 7% relative to their maximum likelihood estimation.

1. Introduction

N-gram language models that include stochastic grammars to capture local linguistic constraints are often used in speech recognition and language understanding systems. Language models combining N-grams and stochastic finite state grammars (FSG) were discussed by Nasr et.al. [4], and those using context free grammars (CFG) have been described by Wang et.al. [5, 6].

To illustrate such models, consider the bigram model shown in panel A of Figure 1. This is an example of the modified Kneser-Ney interpolated bigram [1] that we consider in this paper; it defines the bigram probability of word a_2 conditioned on word a_1 as

$$P(a_2|a_1) = f(a_2|a_1) + b(a_1)u(a_2), \tag{1}$$

where $f(a_2|a_1)$ is a discounted bigram probability, $b(a_1)$ is a history dependent interpolation weight, and $u(a_2)$ is a unigram probability. The bigram of Figure 1 includes a FSG called DATE. With this grammar it captures the constraints on number of days in different months; <s> today is 30th day of march </s> is a valid sentence (i.e. sentence with non-zero probability) with respect to this languge model, whereas <s> today is 30th day of february </s> is not.

In addition to capturing the language constraints that may not be easily handled by N-grams, a significant advantage of using grammars in N-gram models is that such language models may be used to provide an annotation of text with the constituent grammar tags. For example, the sentence

<s> today is 5th day of may </s>

would be annotated by the language model of Figure 1 as

<s> today is DATE 5th day of may DATE </s>.

This annotation is achieved by finding the most likely path through the language model and if a portion of this path is accounted for by a grammar then that portion is marked with the grammar tag; a precise statement of this process is given in Section 2.

In addition to providing text annotation, the use of stochastic grammars in N-grams offers several other advantages. In some cases words or phrases may be easily added to the grammar whereas adding them to the language model would require retraining; for instance in a leap year we could add 29th day of february in the DATE grammar of Figure 1, presumably without having to retrain the language model. Furthermore, since the underlying N-gram model is only trained on words and grammar tags, use of grammars may help alleviate data sparsity problem and may provide robustness in estimation. In many cases the grammars are application independent, e.g. date expressions, confirmation, etc., and once they are carefully designed, they can be used in language models built for several different applications.

In this paper we focus on the annotation aspect of the N-gram + FSG models and present an estimation method that aims at improving their annotation performance over their maximum likelihood estimate. Our method utilizes an HMM formulation [3, 4] of these language models. We first review this formulation in Section 2. We then describe our conditional maximum likelihood (CML) estimation procedure in Section 3. Finally we present our experimental setup and compare CML and ML estimation for their annotation accuracy.

2. HMM Formulation of Models Combining N-grams and FSGs

We first recap a HMM based language model. Let $\mathcal{V}$ denote a set of words. Let $\mathcal{W} = \mathcal{V}^*$ be a countable set of strings made from all possible concatenations of zero or more words from $\mathcal{V}$. Define $H = (\mathcal{S}, \mathcal{F}, \mathcal{W}, \pi, \mathcal{T}, \mathcal{M})$ to be a hidden Markov model with a set of states $\mathcal{S}$ having π as their initial distribution. $\mathcal{F} \subset \mathcal{S}$ is the set of HMM final states. $\mathcal{T}$ is the set of HMM transition probabilities, and $\mathcal{M}$ is a set of probability measures defined over $\mathcal{W}$ with one measure associated with each state in $\mathcal{S}$. H defined in this way is an HMM based language model.

An N-gram model including stochastic FSGs can now be formulated as an HMM based language model, as follows. Let $\mathcal{A}$ be the alphabet of the N-gram; $\mathcal{A}$ contains words, grammar tags, a start-of-sentence symbol <s>, and an end-of-sentence symbol </s>. Each N-gram history, a string of $N-1$ symbols from $\mathcal{A}$, defines a state in the HMM. The transition between state labeled $a_1...a_{N-1}$ and another state labeled $a_2...a_N$ has probability $P(a_N|a_1...a_{N-1})$; transitions only exist between a state pair where the last $N-2$ symbols of the source state label match the first $N-2$ symbols of the destination state. The output distribution associated with states is determined by the last symbol of the state label - if

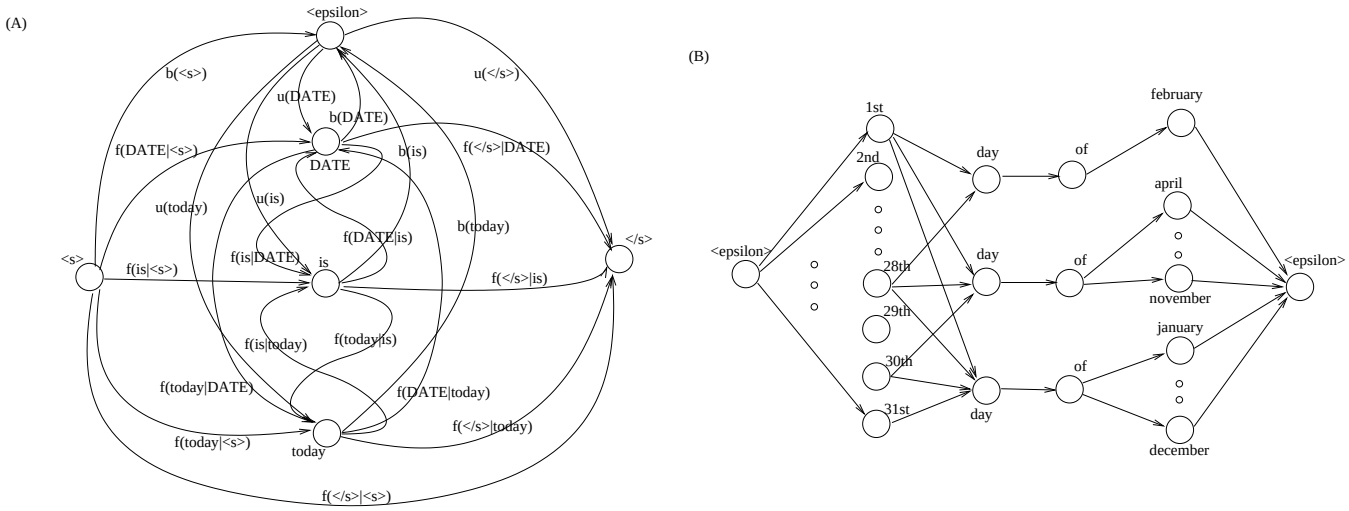

Figure 1: (A) A bigram language model containing a finite state grammar DATE. (B) The DATE grammar.

the last symbol for a state is a word then the output distribution trivially places all its mass on that word, if it is a grammar tag then the output distribution is that specified by the grammar corresponding to that tag, and if it is `</s>` or `</s>` then the output distribution places all its mass on a string of length zero; i.e. that state does not emit amything. The initial distribution π places a mass of 1.0 on the state labeled with a string of $N-1$ `<s>` symbols. Finally, all states whose last symbol is `</s>` make up the set of HMM final states.

2.1. Annotating Text with Grammar Tags

Under the HMM formulation, the annotation of an observed word string W can be stated as the problem of finding the most likely state sequence

$$Q^* = \arg\max_{Q \in \mathcal{Q}} P(Q|W), \tag{2}$$

where $\mathcal{Q}$ is the set of all possible HMM state sequences. In Q^*, if there is a state that is a grammar tag, the substrings of W that aligns with that state is annotated with that tag. For example, given the sentence `<s> today is 5th day of may </s>` the state sequence Q^* would be `today is DATE` where `DATE` aligns with substring `5th day of may` and hence the resulting tagged sentence would be `<s> today is DATE 5th day of may DATE </s>`.

2.2. Maximum Likelihood Parameter Estimation

The parameter estimation of HMM based language models is carried out with text data that is annotated with grammar tags; i.e. both the observed word strings and their underlying state alignments are known. Using W_i to denote i^{th} training word sequence and Q_i to denote its underlying state label, the parameters are estimated so as to maximize the joint likelihood

$$\prod_i P(W_i, Q_i) = \prod_i P(Q_i) \prod_i P(W_i|Q_i) \tag{3}$$

Maximization of $P(Q_i)$ term simply corresponds to building an N-gram, appropriately smoothed, on the given state sequences. In case of the modified Kneser-Ney bigram of (1), the smoothed bigram is built with the following estimates [1]

$$f(a_2|a_1) = \frac{c(a_1a_2) - \lambda(c(a_1a_2))}{\sum_{a_1} c(a_1a_2)} \tag{4}$$

$$b(a_1) = 1 - \sum_{a_2} f(a_2|a_1). \tag{5}$$

$c(a_1a_2)$ is the count of word pair a_1a_2 in the training data, and $\lambda(c(a_1a_2))$ is a count dependent discounting weight. The unigram $u(a_2)$ in (1) is chosen so that the unigram marginals of the resulting bigram match the data marginals. $u(a_2)$ is sometimes further interpolated with a uniform distribution to give a non-zero probability to unseen words. For an excellent in-depth discussion of modified Kneser-Ney parameter estimation we refer the readers to Chen and Goodman [1].

Maximization of the $P(W_i|Q_i)$ term in (3) involves estimating the distribution specified by the grammars on the basis of training word sequences that align with those grammars. In the experiments reported in this paper we chose not to update the grammar probabilities and only focus on estimating the N-gram probabilities. We also note that in case the training data only contains word sequences without the underlying state alignment (grammar annotation), an expectation maximization algorithm can be carried out. However, this is also outside the scope of our current discussion and we shall not pursue it any further.

3. Conditional Maximum Likelihood Parameter Estimation

The conditional maximum likelihood estimation aims at maximizing the conditional likelihood of reference state sequence Q_i given the observed text W_i. I.e., it aims at maximizing

$$\prod_i P(Q_i|W_i) = \prod_i \frac{P(Q_i)P(W_i|Q_i)}{\sum_{Q_i'} P(Q_i')P(W_i|Q_i'))} \tag{6}$$

In contrast to the ML objective of (3), the CML objective is directly related to the annotation criterion of (2) and therefore it can be expected to have a better annotation performance.

The parameter estimation under Equation 6 is carried out using the extended Baum-Welch procedure of Gopalakrishnan et. al. [2].

While it is straight forward to derive the updates for $f(a_2|a_1)$, $b(a_1)$, and $u(a_2)$ components of the modified Kneser-Ney model of (1), it is unclear how to smooth the resulting values to avoid overtraining.

The smoothing method we tried was to only update the $f(a_2|a_1)$ portion of the bigram model while keeping $b(a_1)$ and $u(a_2)$ fixed at their original values. This choice of keeping the interpolation weights fixed was made arbitrarily, under our desire to keep the smoothing mass of each history unchanged from the ML estimated model of (5). This results in the following update

$$\hat{f}(a_2|a_1) = (1-b(a_1))\frac{c^n(a_1a_2) - c^d(a_1a_2) + Df(a_2|a_1)}{\sum_{a_2} c^n(a_1a_2) - c^d(a_1a_2) + Df(a_2|a_1)}, \tag{7}$$

where $c^n(a_1a_2)$ denotes numerator counts obtained from the reference state sequences in the training set as

$$c^n(a_1a_2) = \sum_i \sum_{a_1a_2 \in Q_i} \frac{f(a_2|a_1)}{f(a_2|a_1) + b(a_1)u(a_2)}, \tag{8}$$

and $c^d(a_1a_2)$ denotes denominator counts obtained from a list of K most likely state sequences corresponding to each training sentence

$$c^d(a_1a_2) = \sum_i \sum_{Q_i' \in \text{K-best}} P(Q_i'|W_i) \sum_{a_1a_2 \in Q_i'} \frac{f(a_2|a_1)}{f(a_2|a_1) + b(a_1)u(a_2)}. \tag{9}$$

The extended Baum-Welch procedure [2] suggests a D value to be used in (7). However, instead of using this value, we follow a strategy that is analogous to choosing D for CML estimation of acoustic models [7]. We select D as

$$D = \lambda D^* \tag{10}$$

$$D^* = \max_{a_1,a_2} \frac{c^n(a_1a_2) - c^d(a_1a_2)}{f(a_2|a_1)}, \tag{11}$$

where $\lambda \geq 1.0$ is an empirically selected parameter. We note that this choice of D simply ensures positivity of the numerator on RHS in (7), and consequently ensures validity of $\hat{f}(a_2|a_1)$. We also experimented with history a_1 dependent $D(a_1)$ values, something that is also found to be of value in CML estimation of acoustic models [7]. Details of these experiments are reported in Section 4.

An additional heuristic that we found to be quite useful was to consider only large $f(a_2|a_1)$ values for update in (7). This is because small $f(a_2|a_1)$ values often made D of (11) quite large, even in the case of history dependent D, which prevented the larger, more significant $f(a_2|a_1)$ values from changing much. We therefore introduced a threshold, t_f, below which the f values were not considered for update. The effect of t_f on (7) is to change the $(1 - b(a_1))$ term to $(1 - t(a_1))$ where

$$t(a_1) = b(a_1) + \sum_{a_2 : f(a_2|a_1) < t_f} f(a_2|a_1) \tag{12}$$

Similar to optimal λ value, the value of t_f was also empirically determined as discussed in Section 4.

4. Experiments and Results

The experiments reported in this paper were conducted on an IBM internal corpus collected using a natural language conversational system. People were calling the system to perform transactions - balance inquiry, fund transfer, etc. - on their retirement account. Bigram language models for this task were built with a vocabulary containing 2146 words and 17 stochastic finite state grammars.

The language model training data consisted of 39,715 sentences containing 330,310 un-annotated words. These sentences were manually annotated with grammar tags using 16313 tags in total. Reference state sequences (Q_i in (6)) were derived from this annotated data; these sequences contained 309588 tokens all together consisting of 293275 words and 16313 grammar tags.

A held out set containing 4549 sentences was used for tuning the heuristic parameters described in Section 3. This set contained 38,066 un-annotated words which were annotated with 1961 grammar tags. The resulting state sequences had 35619 tokens - 33658 words and 1961 grammar tags. Having determined the optimal parameter values, the held out set was combined with training set to obtain the final models used in testing.

The test set consisted of 5675 sentences containing 33662 un-annotated words. These sentences were annotated with 2160 grammar tags. The resulting state sequences contained 30351 tokens all together, consisting of 28191 words and 2160 grammar tags.

4.1. Performance Measures

The annotation using language models was evaluated under two measures. First, a hypothesis state sequence was obtained according to (2), and a string edit distance was computed between this and the reference state sequence. This edit distance, as a fraction of total reference states, shall be referred to as the *state error rate*.

While the state error rate is closely related to the CML objective that we optimized for, in some cases it may not be a good indicator of the annotation error. For example, consider the following reference and hypothesized annotations

Ref: `FUND equity index trust fund FUND`
Hyp: `equity index trust fund`

In this case the reference state sequence `FUND` would align with the hypothesized state sequence `equity index trust fund` and would result in four errors. To account for these instances, we computed a *bracketing error* which is a count of deletions, insertions, and substitutions of grammar tags. For instance, the example above has two bracketing errors. We refer to the total number of bracketing errors, as a fraction of the total number of reference "brackets" as *bracketing error rate*.

4.2. Optimization of Heuristic Parameters on Held Out Set

The selection of heuristic parameters: λ in (10), global or history dependent D values, and t_f in (12), was carried out to minimize the state error rate on the held out set. First a baseline modified Kneser-Ney bigram was trained on the training data using an IBM internal language modeling toolkit developed by Stanley Chen. This LM had a perplexity of 9.47, state error rate of 0.66%, and bracketing error rate of 10.07% on the held out set.

Two separate greedy searches, one for global D and one for history dependent D, were carried out to find optimal λ and t_f parameters. For both these searches, an arbitrary initial value was chosen and the search was conducted iteratively by keeping one parameter fixed and using a line search for the other to find minimum state error rate on the held out set. The results of optimization are presented in Table 1.

From the results of Table 1 we note that both global D and history dependent D result in significant improvements over baseline in both state error rate and bracketing error rate. The history dependent D performs nearly twice as good in comparison to global D. However, the updated models have a higher perplexity than the baseline model and their value for recognition and annotation of recognized text remains to be seen.

	parameters	perplexity	state error rate	bracketing error rate	objective function value
baseline		10.70	0.66%	10.07%	-3797.51
global D	$\lambda = 1.01, t_f = 1e-2$	10.73	0.60%	9.15 %	-3628.8
$D(a_1)$	$\lambda = 1.1, t_f = 5e-3$	11.90	0.53%	8.13%	-3088.08

Table 1: Results of parameter optimization on held out set under global and history dependent D values.

We also measured the CML objective function value of (6), under baseline and updated language models. It is quite comforting to note that even though the model updates were carried out to minimize the held out set state error rate, the resulting models have a higher objective function value as compared to the baseline ML estimated model. The higher objective function for history dependent D as compared to that for global D correlates nicely with the better performance of the former model. However, these comparisons are obtained under only one iteration of CML estimation and the relative performance under multiple iteration remains to be seen.

4.3. Annotation Performance on Test Data

To test the CML estimation procedure on the test set we first combined the training and held out sets and built a modified Kneser-Ney bigram on this combined data. This model, when tested on the test set had a baseline state error rate of 0.71% and a bracketing error rate of 7.25%.

The results of CML update of the baseline model are presented in Table 2. The update under history dependent D values was carried out using $\lambda = 1.1$ and $t_f = 5e-3$, and for global D we used $\lambda = 1.01$ and $t_f = 1e-2$. From Table 2 we note that the history dependent D values yield a state error rate reduction of about 8.5% relative over the baseline model. Furthermore, as expected, the model with global D value was worse than the history dependent D model. However, we note that the relative gain in state error rate using history dependent D is much less than it was on held out set. This suggests that the global D may have a better generalization performance and a proper comparison may only be made after multiple iterations.

	state error rate	bracketing error rate
baseline	0.71%	7.25%
global D	0.67%	7.01%
$D(a_1)$	0.65%	6.74%

Table 2: Test set state and bracketing error rates under ML (baseline) and CML estimated models.

5. Conclusions

We have presented a conditional maximum likelihood estimation procedure for language models that combine N-grams and stochastic finite state grammars. This estimation aims at improving the accuracy of these language models in annotating text with constituent grammar tags. On a natural language based transactional task we find that the CML estimation results in over 7% relative improvement in annotation over the ML estimation procedure.

While the CML estimation results in significant improvement in annotation accuracy, there are several issues that are left unanswered. First of all, the effect of multiple iterations on objective function and annotation accuracy needs to be explored. Also, our choice of the smoothing method is somewhat adhoc and other, more principled, ways of smoothing need to be discovered. Furthermore, the effectiveness of the CML estimated models in speech recognition systems and in annotating the output of speech recognizers remains to be seen.

We note that while we have presented the case of bigrams incorporating FSGs, the methods discussed here naturally generaize to higher order N-grams and to N-grams that incorporate other stochastic grammars such as CFGs [5].

6. Acknowledgements

We would like to thank Mark Epstein for discussions that lead to the initiation of this work. We also thank Stanley Chen for providing his marvelous codebase infrastructure that enabled the experiments described in this paper, and to Satya Dharanipragada and Etienne Marcharet for providing useful comments on this paper.

7. References

[1] Chen, S. and Goodman, J., "An empirical study of smoothing techniques for language modeling," Tech. Rep. TR-10-98, Center for Research in Computing Technology, Harvard University, MA, 1998.

[2] Gopalakrishnan, P., Kanevsky, D., Nadas, A., and Nahamoo, D., "An inequality for rational functions with applications to some statistical estimation problems," IEEE Trans. Info. Theory, 37:107–113, 1991.

[3] Jelinek, F. *Statistical methods for speech recognition.* The MIT Press, Cambridge, MA, 1999.

[4] Nasr, A., Esteve, Y., Bechet, F., Spriet, T., and De Mori R., "A language model combining N-grams and stochastic finite state automata," Eurospeech 1999, 5:2175–2178, 1999.

[5] Wang, Y. and Acero, A., "Combination of CFG and N-gram modeling in semantic grammar learning," Eurospeech 2003, 4:2809–2812, 2003.

[6] Wang, Y., Acero, A., and Chelba, C., "Is word error rate a good indicator for spoken language understanding accuracy," ASRU 2003, 577–582, 2003.

[7] Woodland, P. and Povey D., "Large scale discriminative training of hidden Markov models for speech recognition," Computer Speech and Language, 16:25–47, 2002.