RECOGNITION PERFORMANCE OF A LARGE-SCALE DEPENDENCY GRAMMAR LANGUAGE MODEL

Adam Berger
Carnegie Mellon
School of Computer Science
Pittsburgh, PA 15232 USA
aberger@cs.cmu.edu

Harry Printz
IBM
Watson Research Center
Yorktown Heights, NY 10598 USA
printz@watson.ibm.com

ABSTRACT

In this paper, we describe a large-scale investigation of dependency grammar language models. Our work includes several significant departures from earlier studies, notably a larger training corpus, improved model structure, different feature types, new feature selection methods, and more coherent training and test data. We report word error rate (WER) results of a speech recognition experiment, in which we used these models to rescore the output of the IBM speech recognition system.

1. INTRODUCTION

One promising idea for advancing statistical language modeling is based upon dependency grammars, as reported in [4]. This work has the appeal of integrating, via the maximum entropy / minimum divergence (MEMD) technique, information from both syntax and ngrams. Moreover, unlike methods based upon context free grammars, grammatical information enters through the words themselves, rather than via abstract constituents like parse-tree node labels. A preliminary investigation of this idea, just cited, has yielded some positive results.

In this paper we report on further investigations of dependency grammar language models. Our work was inspired by the earlier study, but includes several significant departures, notably a larger training corpus, improved model structure, different feature types, new feature selection methods, and more coherent training and test data. We have conducted a preliminary study of the effect of using such models, in a rescoring paradigm, upon the word error rate (WER) of the IBM speech recognition system. We regret to report that we observed no significant performance improvement.

The rest of this paper is organized as follows. Section 2. is a brief review of MEMD models in general, and the elements of dependency grammar models in particular. Section 3. describes and motivates the types of features we chose to investigate. In Section 4. we discuss the actual selection of features. In Section 5. we describe our experimental setup, and in Section 6. we report the results of the tests we performed. Section 7. is a summary and description of future work.

2. MODEL STRUCTURE

Let $S = w^0 \ldots w^N$ be a sentence, comprised of words $w^0 \ldots w^N$, and let $K(S)$ or just K stand for its linkage. A linkage is a planar graph, in which the nodes are the words of S, and the edges connect linguistically related word pairs. An example of a typical sentence S, with its linkage K, appears in Figure 1.

Our model, written $P(S \mid K)$, is not a language model proper, since it is conditioned upon the linkage. In principle we can recover $P(S)$ as $\sum_K P(S \mid K)P(K)$; in practice we simply take $P(S) \approx P(S \mid K)$. Moreover since K itself depends upon S, the model cannot be applied incrementally, as in a real-time speech recognition system. However such a model can be used to select from a list of complete sentences.

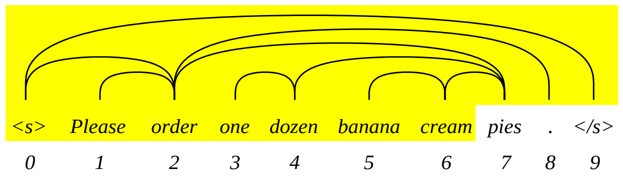


Figure 1: A Sentence S and its Linkage K. The shaded area represents the history h^7, which is the conditioning information available to the model at position 7. h^7 consists of the complete linkage K, and words w^0 through w^6 inclusive.

Our model $P(S \mid K)$ is a minimum divergence model, as described in references [10, 1]. Its value formed in the usual way as the product of individual word probabilities; that is

$$P(S \mid K) = \prod_{i=0}^{N} p(w^i \mid w_0^{i-1} K). \tag{1}$$

We write $h^i = \langle w_0^{i-1}, K \rangle$ for the *history* at position i; this is the information the model may use when predicting word w^i. For the models in this paper, the history consists of the words $w^0 \ldots w^{i-1}$, plus the complete linkage K. The individual word probabilities are determined as

$$p(w^i \mid h^i) = \frac{1}{Z(\bar{\alpha}, h^i)}\, q(w^i \mid h^i)\, e^{\bar{\alpha} \cdot \bar{f}(w^i, h^i)} \tag{2}$$

where

$$Z(\bar{\alpha}, h) = \sum_{w \in V} q(w \mid h)\, e^{\bar{\alpha} \cdot \bar{f}(w, h)}. \tag{3}$$

Here $\bar{f}(w, h)$ is a vector of 0s and 1s, which depends upon the value of each feature at the point w, h. Likewise $\bar{\alpha}$ is a vector of real-valued exponents, which are adjusted during the training of the model. V is a fixed vocabulary of words, and $Z(\bar{\alpha}, h)$ is a normalizing value, computed according to equation (3). Finally $q(w \mid h)$ is the base model, which gives our nominal prediction of w from h.

In the work reported here, the base model q is decidedly *not* a constant. This is what makes ours a minimum divergence model, rather than a maximum entropy model. This approach, while not novel [1, 10], is one of the first key departures of our work from [4].

We chose to use a standard deleted-interpolation trigram language model for q. This gives us access to the information present in q, but spares us from having to incorporate as features the 14,395,589 trigrams, 3,741,147 bigrams and 52,802 unigrams used to determine q. While in principle this could be done, the vast training computation would be infeasible for us. Moreover, using a linearly interpolated trigram as a base model allows us to circumvent a host of problems associated with discounting the empirical expectations of ngram features.

3. LINGUISTIC FEATURES

We now take up the question of how to exploit the information in the history h^i to more accurately estimate the probability of word w^i. We remind the reader that the base model already provides such an estimate, $q(w^i \mid h^i)$. But because in this case q is a trigram model, it discards all of h^i except the two most recent words, $w^{i-2}w^{i-1}$. Our aim is to find informative binary feature functions $f(w^i, h^i)$ that are clues to especially likely or unlikely values of w^i. We chose to use two different kinds of features: triggers and links.

3.1. Trigger Features

As every speaker of English is aware, the appearance of one given word in a sentence is often strong evidence that another particular word will follow. For instance, knowing that *computer* appeared among the words of h^i, one might expect that *nerds* is more likely than normal to appear among the remaining words of the sentence. Word pairs such as these, where the appearance of the first is strongly correlated with the subsequent appearance of the second, are called *trigger pairs* [11]. Note that the trigger property is not necessarily symmetric: we would expect a left parenthesis *(* to trigger a right parenthesis *)*, but not the other way around.

Our model incorporates these relationships through *trigger features.* Let u, v be some trigger pair. A trigger feature f_{uv} is defined as

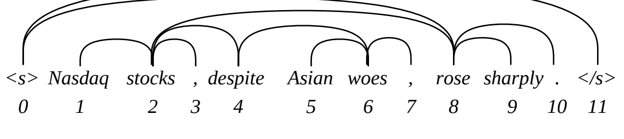
$$f_{uv}(w,h) = \begin{cases} 1 & \text{if } w = v \text{ and } h \ni u \text{ with } |uv| \geq d_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Here $h \ni u$, read "h contains u," means that u appears somewhere in the word sequence of h. The notation $|uv| \geq d_{\min}$ means that the *span* of this pair, defined as the number of words from u to v, including u and v themselves, is not less than a predetermined threshold $d_{\min}$. Throughout this work we have used $d_{\min} = 3$.

3.2. Link Features

One shortcoming of trigger features is their profligacy. In a model built with the feature $f_{computer\ nerds}$, an appearance of *computer* will boost the probability of *nerds* at every position at distance $d_{\min}$ or more to its right. This will be so whether or not a position is a linguistically appropriate site for *nerds*. Moreover, if a model contains a large number of trigger features, there will be many triggered words at each position, and their heightened probabilities will tend to wash each other out.

For instance, consider the sentence of Figure 2. The plausible trigger feature $f_{stocks\ rose}$ will boost the probability of *rose* at every word from position 4 onward, in particular at position 6, even though it is syntactically inappropriate there. Moreover, the acoustically confusable word *woes* appears at this position, and so increasing the probability of *rose* here could yield an error. Thus the boost that $f_{stocks\ rose}$ gives to *rose*, which we desire in position 8, is just as clearly not desired in position 6. Unfortunately the trigger is blind to the distinction between these two sites, and it boosts *rose* in both places.



<s> Nasdaq stocks , despite Asian woes , rose sharply . </s>
0 1 2 3 4 5 6 7 8 9 10 11

Figure 2: Links versus Triggers. The trigger feature for *stocks* and *rose* boosts the probability of *rose* at each position from 4 to 11, inclusive. The link feature also boosts *rose*, but only at positions 4 and 8. Shown here is the actual linkage computed by our parser.

These considerations have led us and others to consider features that use the linkage. The aim is to focus the effect of words in the history upon the particular positions that are appropriate for them to influence. Figure 2 shows how the linkage of this sentence connects *stocks*, the headword of the subject noun phrase, with *rose*, the main verb of the sentence; note there is no such link from *stocks* to *woes*. These is precisely the kind of linguistic fact that we wish to exploit, using an appropriate feature function. To do so, we will construct a feature function that (like a trigger) turns on only for a given word pair, and in addition only when the named words are connected by an arc of the linkage.

Because such features depend upon the the linkage of the sentence, we refer to them as *link features.* Such a feature $f_{\overset{\frown}{u\,v}}$, for words u and v, is defined as

$$f_{\overset{\frown}{u\,v}}(w,h) = \begin{cases} 1 & \text{if } w = v \text{ and } h \ni \overset{\frown}{u\,v} \text{ with } |uv| \geq d_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The notation $h \ni \overset{\frown}{u\,v}$, read "$h$ contains u, linking v," means that word u appears in the history's word sequence, that an arc of K connects u with the current position, and that word v appears in the current position.

4. FEATURE SELECTION

Once the base model and feature types have been chosen—choices generally dictated by computational practicality, and the information available in the training corpus—the key open issue is which features to incorporate in the model. In general we cannot and will not want to use every possible feature. For one thing, we usually have too many features to train a model that includes all of them: the processing and memory requirements are just too great. Moreover, rescoring with a model that has a very large number of features is itself time-consuming. Finally, many features may be of little predictive value, for they may just repeat information that is already present in the base model.

In this section we describe our method for selecting model features. Our method proceeds in three phases: candidate identification, ranking, and selection.

Candidate Identification By *candidate identification* we mean a pass over the training corpus (or some other corpus) to collect potential features for the model. The result of this pass is a candidate feature set F.

One or more criteria may be applied to decide which features, out of the many exhibited in the corpus, are placed into F in the first place. In the work reported here, we scanned the parsed corpus to collect potential features, both triggers and links. Since we were building a model using a trigram base model, we had good reason to believe that linguistic relations between adjacent words were well-modeled by this base, and so we ignored links or triggers of span 2. Also, to keep from being swamped with features of no semantic importance, and which arise purely because the words involved are common ones, we likewise ignored triggers where either word was among the 20 most frequent in the corpus. For similar reasons we did not include any trigger pair with a count below 6, nor any link pair with a count below 4. In this way we collected a total of 538,998 candidate link features (which were all those passing the criteria above) and 1,000,000 candidate trigger features (which were those passing the criteria above, and then the top million when sorted by mutual information). We supplied the resulting set F, containing 1,538,998 features, to the next stage of the feature selection process.

Ranking In the next phase, *ranking*, we determine the *gain* of each feature, and sort the features in order of decreasing gain. The gain of a feature f is a well-known [5] and well-studied [2, 3] figure of merit that arises naturally in the theory of MEMD models; it is defined as follows. Let $\mathcal{C}$ denote a corpus of N words, and let $P(\mathcal{C})$ denote the

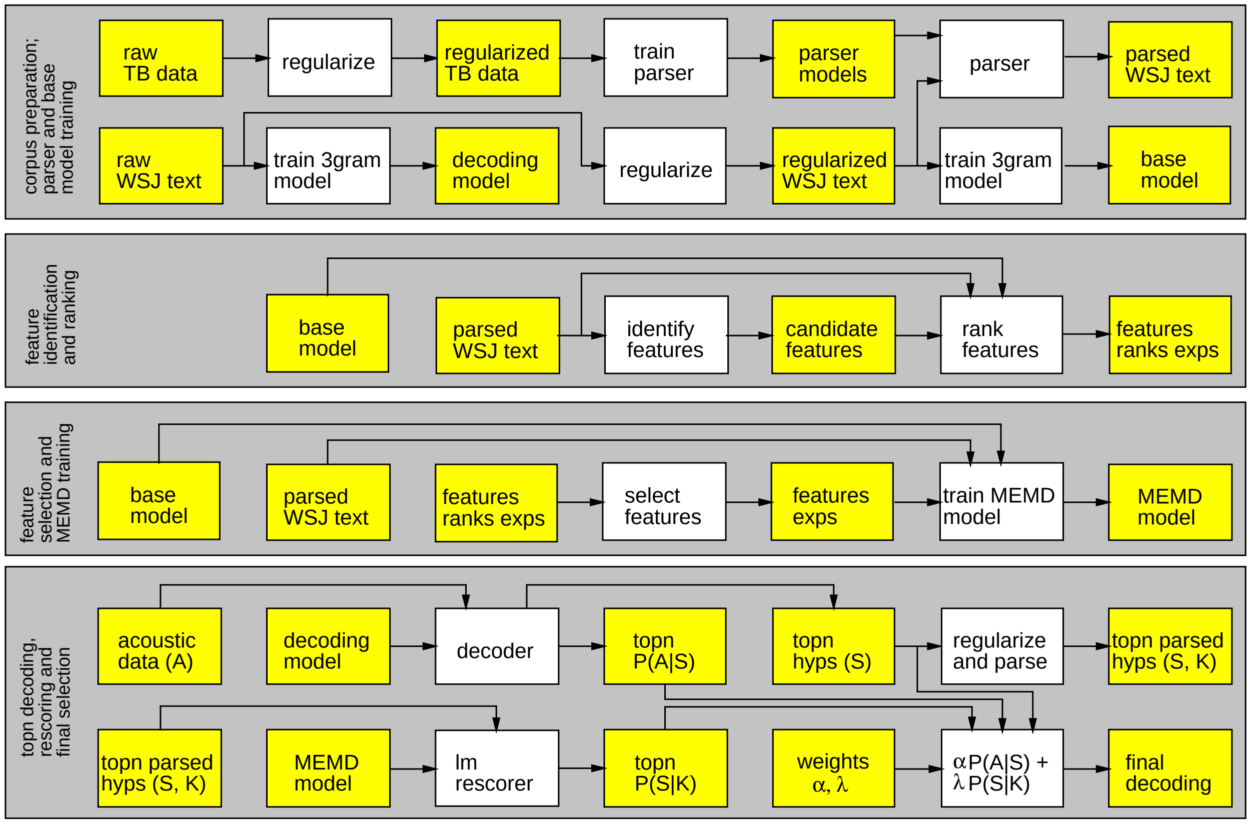


Figure 3: Corpus Preparation, Feature Selection, Model Training, Decoding and Rescoring. See Section 5 for a discussion.

corpus probability according to the base model. Now construct an MEMD model from the base model, constrained by the single feature f; let $P_f(\mathcal{C})$ denote the corpus probability, according to this single-feature model. Then the gain of f, written G_f, is defined as

$$G_f = \frac{1}{N} \log \frac{P_f(\mathcal{C})}{P(\mathcal{C})} = \frac{1}{N} \log P_f(\mathcal{C}) - \frac{1}{N} \log P(\mathcal{C}). \quad (6)$$

This quantity may be understood as follows. $P(\mathcal{C})$ and $P_f(\mathcal{C})$ respectively measure how well the base and single-feature models accord with the training corpus. Numerically, we wish that $P_f(\mathcal{C}) > P(\mathcal{C})$, for this means that overall, the model $P_f(\mathcal{C})$ assigns more probability mass to words as they actually appear in the corpus than does $P(\mathcal{C})$. Thus the more useful feature f is in predicting the corpus, the higher the value $P_f(\mathcal{C})$ will attain.

Formally, the gain measures the improvement in cross-entropy afforded by f, or more simply, the information content of f. Reference [3] contains an extensive discussion of the gain as a criterion for ranking features, and reference [9] describes a fast algorithm for computing the gain of a large number of features.

Selection Ranking places the features of F in order, from most to least gainful. It also provides an initial estimate of each feature's exponent. However, though it is clear that we wish to choose features from F in rank order, say retaining the top 10,000 or 100,000 features, the ranking algorithm does not indicate how many features to select. This is the final step in determining the model features, which we call *selection*. It consists of choosing where in the ranked feature list to draw the line, and must be decided by hand by the modeler.

5. EXPERIMENTAL SETUP

The overall structure of our experiment appears in Figure 3 above. The figure's top section shows the development of the parser and the two standard trigram language models we require. The parser, a modified version of the decision-tree parser of David Magerman [8], was trained on 990,145 words of Treebank Release II data [6]. The two trigram models were trained on 41,855,429 words of Wall Street Journal text [7]. The first one is trained on words as they appear in ordinary running text; its vocabulary contains 56,700 words. This is called the *decoding model.* The second one is trained on text that has been modified to create suitable input to the parser, for instance by detaching and exhibiting grammatical particles, like possessives and contracted negations (thus, *Bill's*→*Bill 's*, and *couldn't*→*could n't*). This serves as the base of our MEMD model; its vocabulary contains 52,817 words (it is smaller because it does not include possessives, which are a surprisingly large fraction of the first vocabulary).

The second section shows the first two steps of feature selection, which are candidate identification and ranking. The third section shows the final selection of features, and the training of the MEMD model. The word exps here is an abbreviation for exponents, which are the quantities adjusted in MEMD training. Finally, the bottom section shows the complete decoding and rescoring procedure.

6. TESTS AND RESULTS

6.1. Test Data

For acoustic test data we used 16 KHz-sampled recordings of male speakers, made in a quiet environment with a Sennheiser microphone. Speakers read verbalized-punctuation Wall Street Journal text [7]; the text of the test data did not overlap with the training data. We note this important difference between test and training data text: the training data appeared to consist of complete paragraphs of arbitrary length, and sometimes complete articles, drawn from the Wall Street Journal. Though the test data consisted of sentences drawn from the same publication at contemporaneous dates, no one speaker's selection contained more than four contiguous sentences from the same article. Since our trigger features were gathered from complete paragraphs or articles, and then ranked and trained on the same text, neither the selection, ranking nor training will accord well with the test data.

6.2. Model Training

We trained and tested eleven models, drawing features from the set F of 1,538,998 candidates described in Section 4. above. Our aims were to compare the benefit of different feature types (though see the caveat above regarding triggers), to investigate the effect of adding features, and of course ultimately to drive down the error rate. In all cases we trained on the complete corpus. We performed MEMD training using the *improved iterative scaling* algorithm of [5]. Additional information about our training method may be found in [3].

6.3. Model Tests and Performance

The acoustic test data were subdivided into two sets, *heldout* and *final test*, comprising 2553 and 11186 words of reference text respectively. Using the unmodified IBM speech decoder, we generated 100 hypotheses for each test utterance. For each MEMD model we studied, we determined the optimal value of α for rescoring the heldout data, using the geometrically weighted mixture $\alpha \log P(A \mid S) + \lambda \log P(S \mid K)$, with λ held fixed at 1. Here $P(A \mid S)$ is the acoustic probability, and $P(S \mid K)$ is the dependency grammar language model probability, for utterance A and hypothesis S. We then rescored the final test data using the α determined on the heldout data.

model	heldout data word errors		final test word errors	
	(#)	(%)	(#)	(%)
decoded	220	8.6	1161	10.4
baseline	206	8.1	1031	9.2
10k	206	8.1	1060	9.5
10k.2trig	205	8.0	1027	9.2
10k.2link	199	7.8	1070	9.6
50k	202	7.9	1052	9.4
50k.2trig	204	8.0	1020	9.1
50k.2link	197	7.7	1015	9.1
100k	200	7.8	1055	9.4
100k.2link	196	7.7	1042	9.3
150k.2link	199	7.8	1037	9.3
200k.2link	200	7.8	1034	9.2
500k.2link	202	7.9	1037	9.3

Table 1: Model Performance.

Results appear in Table 1. For each model, we report performance on both the heldout data and the final test data. Model *decoded* lists the error rate for the top-ranked hypothesis produced by the unmodified IBM speech decoder. However, this is not a fair basis for comparison. The fair comparison is made with model *baseline*, which is the result of reranking the topn decodings using the base model, after tuning its mixture weight α. For both heldout and final data, this yielded modest reductions in the error rate. These improvements are not attributable to dependency grammar modeling, because *baseline* is just a standard trigram model.

The remaining lines in the table list results for MEMD dependency grammar models. The notation *50k* indicates that the model contained the top 50,000 features, as ranked by gain. The notations *2trig* and *2link* indicate the model contained only trigger or link features respectively; absence of either indicates the model contained a mix of both feature types.

Our initial experiments, on heldout data, showed small but promising improvements, on the order of 5% relative WER reduction. However, the true measure of a model is its performance on final test data. Here our results may be described as mixed, at best. Only three models (*10k.2trig, 50k.2trig, 50k.2link*) showed any improvement at all, and these gains were on the order of 1.5% relative WER reduction.

7. SUMMARY

In this paper, we studied the construction and use of MEMD dependency grammar models in automatic speech recognition. Our work was inspired by [4], but differs in the use of a trigram base model, the inclusion of both trigger and link features, use of larger and more coherent training and testing corpora, feature selection by the gain statistic, rescoring of more accurate hypotheses, and performance comparison on final test data. In their current form, our models afforded no significant performance improvement. A detailed error analysis, which we have not yet conducted, is required to decide if these models warrant further investigation.

ACKNOWLEDGEMENT

We gratefully acknowledge support in part by the National Science Foundation, grant IRI-9314969.

REFERENCES

[1] D. Beeferman, A. Berger, J. Lafferty, “A Model of Lexical Attraction and Repulsion,” *Proceedings of the ACL-EACL'97 Joint Conference*, Madrid, Spain.

[2] A. Berger, S. Della Pietra, V. Della Pietra, “A Maximum Entropy Approach to Natural Language Processing,” *Computational Linguistics*, 22(1): 39–71, March 1996.

[3] A. Berger, H. Printz, “A Comparison of Criteria for Maximum Entropy / Minimum Divergence Feature Selection,” *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing*, 97–106, Granada, Spain, June 1998.

[4] C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke, D. Wu, “Structure and Estimation of a Dependency Language Model,” *Proceedings of Eurospeech '97*, Rhodes, Greece, September 1997.

[5] S. Della Pietra, V. Della Pietra, and J. Lafferty, “Inducing Features of Random Fields,” IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 19(4): 380–393, April 1997.

[6] Linguistic Data Consortium, *University of Pennsylvania Treebank Release II*, Philadelphia, PA.

[7] Linguistic Data Consortium, *Wall Street Journal Corpus, Release 0*, Philadelphia, PA.

[8] D. Magerman, *Natural Language Parsing as Statistical Pattern Recognition*, Ph.D. Thesis, Department of Computer Science, Stanford University, Palo Alto, CA, February 1994.

[9] H. Printz, “Fast Computation of Maximum Entropy / Minimum Divergence Feature Gain,” *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney, Australia, November 1998.

[10] P. S. Rao, S. Dharanipragada, S. Roukos, “MDI Adaptation of Language Models Across Corpora,” *Proc. of Eurospeech '97*, pages 1979–1982, Rhodes, Greece, September 1997.

[11] R. Rosenfeld, *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, April 1994.