A Comparison of Criteria for Maximum Entropy / Minimum Divergence Feature Selection

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

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

Abstract

In this paper we study the *gain*, a naturally-arising statistic from the theory of MEMD modeling [2], as a figure of merit for selecting features for an MEMD language model. We compare the gain with two popular alternatives—empirical activation and mutual information—and argue that the gain is the preferred statistic, on the grounds that it directly measures a feature's contribution to improving upon the base model.

Introduction

Maximum entropy / minimum divergence (MEMD) modeling is a powerful technique for building statistical models of linguistic phenomena. It has been applied to problems as diverse as machine translation [2], parsing [10], word morphology [5] and language modeling [6, 11, 3, 9]. The heart of the method is to choose a collection of informative *features*, each encoding some linguistically significant event, and then to incorporate these features into a family of conditional models.

A fundamental issue in applying this technique is the criterion used to select features. The work described in [3], for instance, incorporates every feature which either appears with above-threshold count in a training corpus, or which exhibits high mutual information. In [11] and [1], the authors select features based on a mutual information statistic. As we argue below, both these methods have drawbacks.

In this paper, we examine a statistic for selecting MEMD model features, called the *gain*. The gain was introduced in [4], and studied in greater detail in [5] and [2]. We present intuition, theory and experimental results for this statistic, as a criterion for selecting features for an MEMD language model. We believe our work marks the first time it has been used in MEMD language modeling, and the first side-by-side comparison with other selection criteria. Though our experimental results concern language models exclusively, we note that the gain can be used to select features for any MEMD model on a discrete space.

The language model we present is based on dependency grammars. It is similar to, but extends upon, the work reported in [3]. Two important differences between that work and ours are that ours is a true minimum-divergence model, and ours incorporates both link and trigger features.

The paper is organized as follows. In Section *Structure of the Model* we give a brief review of MEMD models in general, and of our dependency grammar model in particular. In Section *Linguistic Features* we describe and motivate the types of features we chose to investigate. In Section *Experimental Setup* we describe our experimental procedure. In Section *Selection of Features* we discuss feature selection; it is here that we develop the notion of gain. In Section *Additivity of the Gain* we discuss the additivity of gain, which measures the extent to which features contribute independently to a model. In Section *Tests and Results* we report our test results. Section *Summary* concludes the paper.

Structure of the Model

Use of a Linkage

Let $S = w^0 \ldots w^N$ be the sentence in question, 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. A typical sentence S, with its linkage K, appears in Figure 1. The relationship between the linkage of a sentence, and the familiar notion of a parse tree, is described in Section *Experimental Setup* below.

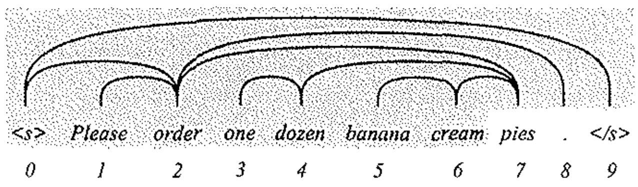

Figure 1: A Sentence S and its Linkage K. The shaded area represents the history h^7, which is the conditioning information available to the model at position 7. h^7 consists of the complete linkage K, and words w^0 through w^6 inclusive.

Our model, 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, for instance in a real-time speech recognition system. However, such a model can be used to select from a list of complete sentences.

The value $P(S \mid K)$ computed by our model is formed in the usual way as the product of individual word probabilities; that is

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

Here we have written $h^i = \langle w_0^{i-1}, K \rangle$ for the *history* at position i; this is the information the model may use when predicting the next word. Here and below the notation w_i^j, with $i \leq j$, stands for the word sequence $w^i \ldots w^j$. Thus for the models in this paper, the history consists of the words $w^0 \ldots w^{i-1}$, plus the complete linkage K.

Fundamentals of MEMD Models

The individual word probabilities $p(w^i \mid h^i)$ appearing in equation (1) above are determined by a *minimum divergence model.* Here we review of the fundamentals of such models; a thorough description appears in reference [2].

As above, let w stand for the word or *future* to be predicted, and let h stand for the history upon which this prediction is based. Suppose that $f(w\ h)$ is a binary-valued indicator function of some linguistic event. For instance, f may take the value 1 when the most recent word of h is the definite article *the* and the word w is any noun; otherwise f is 0. Or f might be 1 when h contains the word *dog* in any location and w is the word *barked.* Any such function $f(w\ h)$ is called a *binary feature function*; clearly we can invent a large number of such functions.

Now suppose C is a large corpus. C can be regarded as a very long sequence of word-history pairs $w^i\ h^i$, where w^i is the word at position i and h^i is the history at that position. We can use C to define the empirical expectation $E_{\tilde{p}}[f]$ of any feature function f; it is given by

$$E_{\tilde{p}}[f] = \sum_i f(w^i\ h^i)/N \quad (2)$$

where i runs over all the positions of the corpus, and N is the number of positions. The sum $A_f = \sum_i f(w^i\ h^i)$ is called the *empirical activation* of the feature f; it is the number of corpus positions where the feature is active (attains the value 1).

Finally, let $q(w \mid h)$ be some selected statistical language model, for instance a trigram model. We call q the *base model.* When q is a trigram, it predicts w based exclusively upon the two most recent words appearing in h. Note however that an arbitrary feature function f can inspect any word of h, or the linkage itself if it comprises part of h. It is the enlarged scope of information available to f that we hope to exploit.

We can now enunciate the principle of minimum divergence modeling. Let $\bar{f} = \langle f_1 \ldots f_M \rangle$ be a vector of binary feature functions, with a known vector of empirical expectations $\langle E_{\tilde{p}}[f_1] \ldots E_{\tilde{p}}[f_M] \rangle$. We seek the model $p(w \mid h)$ of minimal Kullback-Liebler divergence from the base model $q(w \mid h)$, subject to the constraint that

$$\langle E_p[f_1] \ldots E_p[f_M] \rangle = \langle E_{\tilde{p}}[f_1] \ldots E_{\tilde{p}}[f_M] \rangle. \quad (3)$$

That is, the expectation of each f_i, according to the model p, must equal its empirically observed expectation on the corpus C.

By familiar manipulations with Lagrange multipliers, as detailed in [2], the solution to this problem can be shown to be

$$p(w \mid h) = \frac{1}{Z(\bar{\alpha}\ h)} q(w \mid h)\, e^{\bar{\alpha} \cdot \bar{f}(w\ h)} \quad (4)$$

where

$$Z(\bar{\alpha}\ h) = \sum_{w \in V} q(w \mid h)\, e^{\bar{\alpha} \cdot \bar{f}(w\ h)}. \quad (5)$$

Here $\bar{f}(w\ h)$ is a vector of 0s and 1s, depending upon the value of each feature function at the point $w\ h$. Likewise $\bar{\alpha}$ is a vector of real-valued exponents, which are adjusted during the training of the model so that equation (3) holds. V is a fixed vocabulary of words, and $Z(\bar{\alpha}\ h)$ is a normalizing value, computed according to equation (5). Finally $q(w \mid h)$ is the base model, which represents our nominal prediction of w from h. When q is the constant function $1/|V|$, the resulting model p is called a *maximum entropy model*; when q is non-constant, p is called a *minimum divergence model.* However the defining equations (4, 5) are the same, regardless of the nomenclature.

Use of a Base Model

In the work reported here, the base model q is decidedly *not* a constant: it is a linearly-interpolated trigram model, trained on a corpus of 44,761,334 words. This approach, while not novel [1], is one of the key departures of our work from [3].

This departure is significant for three reasons. First, it gives us a computationally efficient way to incorporate a large amount of valuable information into our model. To put this another way, we already know that the 14,617,943 trigrams, 3,931,078 bigrams and 56,687 unigrams that together determine q are useful linguistic predictors. But if we should try to incorporate each of these word-grams into a pure maximum entropy framework, via its corresponding feature function, we would be faced with an intractable computational problem.

Second, the use of raw word-gram feature functions, without some discounting of expectations, is believed to be problematic for maximum entropy models, since it can force solutions with unbounded exponents. By incorporating word-gram information via a linearly interpolated trigram model, we are less likely to encounter this problem.

Third, using a trigram base model raises a new and challenging version of the feature selection problem. How can we determine which features, when incorporated into the model, will actually yield an advance upon the trigram model? This is the central problem of this paper, which we proceed to address by using the gain statistic.

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.

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. Some words are in fact good predictors of themselves: seeing *Japanese* once in a sentence raises the likelihood it will appear again later. 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* [1, 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

$$f_{uv}(w\ h) = \begin{cases} 1 & \text{if } w = v \text{ and } h \ni u \text{ with } |uv| \geq d_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

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 $d_{\min} = 3$.

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. But here the acoustically confusable word *woes* appears, and so increasing the probability of *rose* at this position 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.

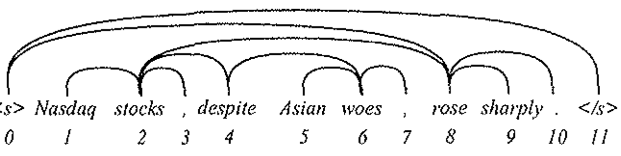
<s> Nasdaq stocks , despite Asian woes , rose sharply . </s>
0 1 2 3 4 5 6 7 8 9 10 11

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. The linkage shown here is the actual one 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 are precisely the linguistic facts 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

$$f_{\overset{\frown}{u\,v}}(w\ h) = \begin{cases} 1 & \text{if } w = v \text{ and } h \ni \overset{\frown}{u\,v} \text{ with } |uv| \geq d_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

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. In the example given above, the link feature $f_{\overset{\frown}{stocks\ rose}}$ attains the value 1 at position 8 only.

Experimental Setup

Here we describe the computation that underlies the work in this paper. Figure 3 is a schematic of the complete computation, which divides into three phases: (1) prepare the corpus and train a parser and base model, (2) identify and rank features, and (3) select features and train an MEMD model. Our experiments, which we report later, concerned phases (2) and (3) only. We include a discussion of phase (1) for completeness, and to place our experiments in context.

In the first phase we trained a parser and base model, and parsed the corpus text. By parsed we mean that for each sentence S of the corpus text T, we have its linkage $K(S)$ at our disposal. The parser we trained and then used was a modified version of the decision-tree parser described in [7]. Our parser training corpus consisted of 990,145 words of Treebank Release II data, and our base model corpus consisted of 44,761,334 words of Wall Street Journal data, both prepared by the Linguistic Data Consortium.

This parser constructs a conventional parse tree. Since we needed linkages, we used the method of headword propagation to create them from the parser output; we now explain this method. To each parse tree we apply a small collection of headword propagation rules, which operate leaves-to-root. The result is a tree labeled with a headword at each node, where each headword is selected from the headwords of a node's children. (At the leaves, each word is its own headword.) The desired linkage is then obtained by drawing an arc from the headword of each child node to the headword of its parent, excluding self-loops. A conventional parse tree for the sentence of Figure 2 above, labeled with propagated headwords, appears in Figure 4.

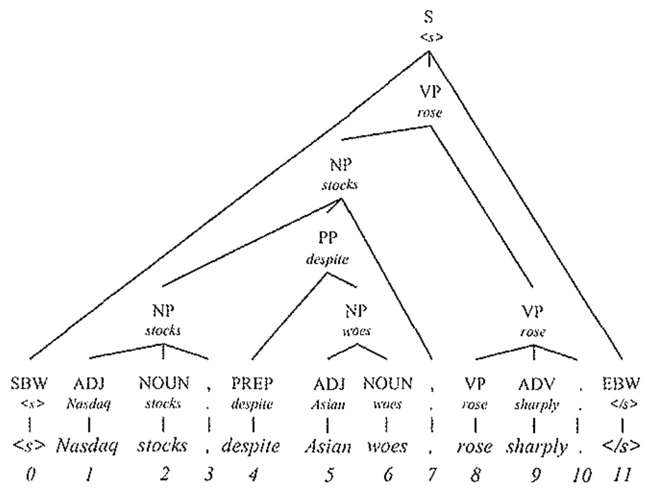

Figure 4: Conventional Parse Tree, with Propagated Headwords. The text explains how this headword-labelled tree can be transformed into the linkage of Figure 2.

For the base model q, we chose to use a linearly interpolated trigram language model, built from the same regularized WSJ corpus as the dependency grammar model itself.

In the second processing phase we identified and ranked features. The details of this phase, and in particular the figure of merit used for ranking, are the subject of Section *Selection of Features*. Here we explain its place in the overall scheme. By inspecting the parsed corpus $\mathcal{C}$, we identify a set F of trigger and link candidate features. These are then *ranked* according to the chosen statistic. In this paper we advocate the use of the gain as the rank statistic. The gain depends upon both the corpus and the base model, and for this reason these are shown as inputs to the box rank features in Figure 3. The output is the same set of candidate features, ranked according to the figure of merit. It happens that the gain computation also yields initial estimates of the MEMD exponents; abbreviated exps in the figure.

In the final phase of processing, we inspected the ranked list of features and selected those to incorporate into the model. We then used the selected features, their initial exponent estimates, the corpus, and the base model to train the MEMD model. Different choices of features yield different models; Section *Tests and Results* below gives details and performance of the various models we built.

Selection of Features

Once the model's prior 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 seldom activate, or may just repeat information that is already present in the prior.

In this section we describe a method for selecting precisely those features of greatest predictive power, over and above the base model q. The key idea of our method is to seek features that improve upon q's predictions of the training corpus itself. The measure of improvement is a statistic called the *gain*, which we define and motivate below. As we will demonstrate, computing the gain not only yields a principled way of selecting features; it can also be of great help in constructing the MEMD model that contains the selected features.

Our method proceeds in three steps: candidate identification, ranking, and selection. We now describe each step in greater detail.

Candidate Identification

By *candidate identification* we mean a pass over the training corpus (or some other corpus) to collect po-

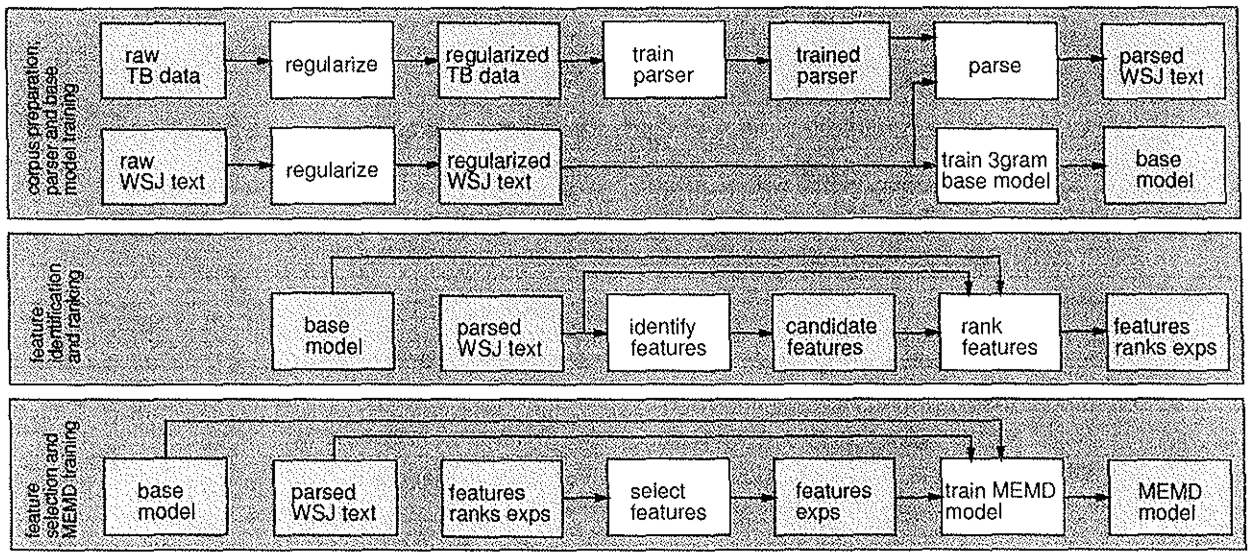

Figure 3: Corpus Preparation, Feature Ranking, and Model Training

tential features for the model. The result of this pass is a *candidate feature set*, denoted F. The candidate features are those that we will rank by gain in the next step.

Note that 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 prior, we had good reason to believe that adjacent words were well-modeled by this prior, and so we ignored links or triggers of span 2. 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. Moreover we did not include any trigger pair with an empirical activation 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 1,000,000 when sorted by mutual information). We supplied the resulting candidate set F, containing 1,538,998 features, to the next stage of the feature selection process.

Ranking

In this section we will motivate and develop the central feature of this paper, which is the notion of *gain*. First introduced in [2], and further developed in [5], the gain is a statistic computed for a given feature f, with respect to a base model, over some fixed corpus. We will argue that the gain is the appropriate figure of merit for ranking features.

Motivation At the heart of the issue lie the following two questions. First, how much does a feature f aid us in modeling the corpus? Second, to what extent does this feature help us to improve upon the base model? By giving quantitative answers to these questions, we will be led to the gain.

We begin by establishing some notation. Let $P(\mathcal{C})$ stand for the probability of the corpus, according to the base model q; that is $P(\mathcal{C}) = \prod_{i=0}^{N} q(w^i \mid h^i)$. For the model developed here, this should more properly be written $P(\mathcal{T} \mid \mathcal{K})$, where $\mathcal{T}$ represents the collected text of the corpus, and $\mathcal{K}$ consists of the linkage of each sentence of $\mathcal{T}$. However since our meaning is clear, for typographic simplicity we will use the shorter notation.

Now we remind the reader of the connection between MEMD training and maximum-likelihood estimation. Suppose we construct an exponential model, from base model q, that contains one single feature $f(w\ h)$. The form of this model will be

$$p_\alpha(w \mid h) = \frac{1}{Z(\alpha\ h)}\, q(w \mid h)\, e^{\alpha f(w\ h)} \tag{8}$$

where $Z(\alpha\ h)$ is the usual normalizer, and α is a free parameter. For any given value of α, the probability $P_{f\alpha}(\mathcal{C})$ of the entire corpus $\mathcal{C}$, as predicted by this model, is

$$P_{f\alpha}(\mathcal{C}) = \prod_{i=0}^{N-1} p_{\alpha^*}(w^i \mid h^i). \tag{9}$$

The MEMD trained value of α, denoted α^*, is determined as

$$\alpha^* = \operatorname*{argmax}_{\alpha} P_{f\alpha}(\mathcal{C}). \tag{10}$$

That is, the particular α that makes expression (8) the MEMD model is precisely the value α^* given by (10).

This fact is demonstrated in [5], along with a proof that the maximizing α^* is unique.

Thus the probability of the complete corpus, according to the MEMD model p_{α^*}, is just $P_{f\alpha^*}(C)$. When the identity of the feature is clear, we will abbreviate this by $P_{\alpha^*}(C)$.

We proceed to motivate and define the gain. At many positions of the corpus, the models q and p_{α^*} will yield the same value. But in those positions where they disagree, we would hope that p_{α^*} does a better job, in the sense that $p_{\alpha^*}(w^i \mid h^i) > q(w^i \mid h^i)$. That is, we wish that p_{α^*} distributes more probability mass than q on the word that actually appears in corpus position i. The extent to which this occurs is a measure of the predictive value of f, the feature that underlies p_{α^*}.

Of course, we do not want to gauge the value of f by a comparison of models on this or that particular corpus position. But we can judge the overall value of f by comparing $P_{\alpha^*}(C)$, the probability of the entire corpus according to a model that incorporates both q and f, with $P(C)$, the probability of the entire corpus according to q alone.

We can quantify the degree of improvement by writing

$$G_f(\alpha^*) = \frac{1}{N}\log\frac{P_{\alpha^*}(C)}{P(C)} = \frac{1}{N}\log P_{\alpha^*}(C) - \frac{1}{N}\log P(C). \tag{11}$$

We refer to $G_f(\alpha^*)$ as the *gain* of feature f. By the rightmost equality above, the gain measures the improvement in cross-entropy afforded by f, or more simply, the information content of f. When it is clear which feature we mean, we will write just $G(\alpha^*)$ for its gain. Likewise we will write G_f when we don't need to display the exponent. The seemingly ancillary quantity α^* is in fact of value, since it is an initial estimate of the feature's associated exponent, and may be used as a starting point in an MEMD training computation that includes this and other features.

Clearly, computing a feature's gain is intimately related to training an MEMD model containing this single feature. But because the model p_{α^*} involves only one feature, substantial computational speedup is possible. A fast algorithm for computing the gain appears in [8].

The notion of gain extends naturally to a set of features M. If $P_M(C)$ is the corpus probability according to a trained MEMD model built with feature set M, then we define $G_M = (1/N)\log(P_M(C)/P(C))$.

Comparison with Other Criteria A key advantage of the gain as a figure of merit is that it overcomes shortcomings of two competing criteria: the feature's empirical activation, and the mutual information of its history with its future. There are clear rationales for both alternatives, but also clear drawbacks.

Selecting by empirical activation ensures that we are choosing features that could significantly reduce the corpus perplexity, for they are active at many corpus positions, and hence can often alter the base model probability. But there is no guarantee that they change the MEMD model much from the base model, since the selected features might simply express regularities of language that the base model already captures. Of course there is no harm in this, but it does not yield a better model.

Likewise, the mutual information criterion could choose features that coincide with, rather than depart from, the base model. Moreover this criterion can suffer from inaccurate estimates of its constituent probabilities, when the feature is rare.

The gain remedies these problems. It finds features that cause the MEMD model to depart, in a favorable way, from the base model. And if a feature is rare, it is ignored, unless it is very valuable in those cases where it appears.

To test this claim, we computed the gain, empirical activation, and mutual information of the 538,998 candidate link features that we collected earlier from our corpus. We then plotted the gain against empirical activation, and against mutual information; these plots appear in Figure 5. It is clear that gain is only weakly correlated with these competing statistics. In Section *Models Trained* below, we compare the perplexities of models built by selecting features with these three criteria.

Final Selection

Ranking places the features of F in order, from most to least gainful. 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. Thus this last step—choosing where in the ranked list to draw the line—must be decided by hand by the modeler.

Since part of our aim was to compare the relative value of link and trigger features, we elected to build models containing the top T triggers and the top L links, for various values of T and L. We also built a model in which we simply retained the top 10,000 features by rank, without regard to their type.

For illustration, we provide in Table 1 a list of 25 selected trigger and link features, of the 1,538,998 in F, ranked by gain. The table also gives the value of α^* for each feature f; this number is reported as e^{α^*}, since this roughly corresponds the probability boost the future of each feature receives, when the feature is active.

Comparison with Feature Induction

In selection by ranking, we form a set F of candidate features, rank them by gain with respect to the base model q, and retain some number of top-ranked features to build the MEMD model p. We regard this approach as eminently reasonable. But there is this danger of inefficiency: we may incorporate two or more features that capture essentially the same linguistic information.

As a prophyllaxis against this, some authors [2] have advocated *feature induction*. Feature induction is an

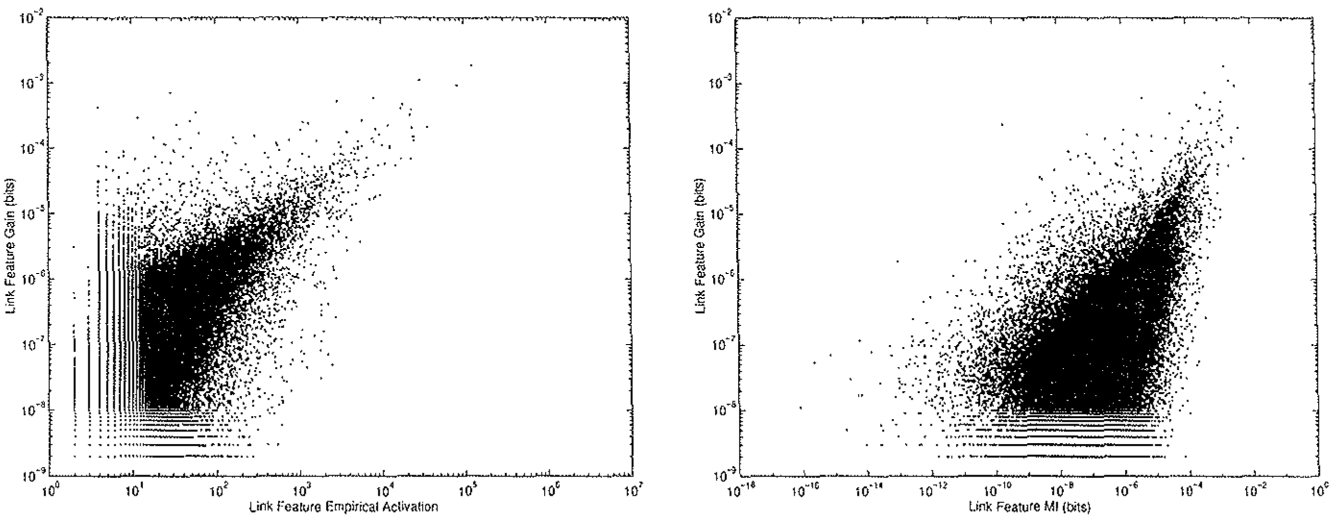

Figure 5: Comparison of 2Link Feature Gain with Empirical Activation and Mutual Information. Left: Scatterplot of feature gain against empirical activation. Right: Scatterplot of feature gain against mutual information.

iterative algorithm for choosing features; it selects one new feature on each iteration. One iteration consists of (1) complete training of an MEMD model using a current set of selected features, initially empty, (2) ranking all remaining candidates against this just-trained model, and (3) removing the single top-ranking feature from the candidate set, and adding it to the set of selected features. Feature induction terminates after incorporating some fixed number of features, or when the gain of the highest-ranked feature, with respect to the current model, drops below some threshold. In this way, if two features f and f' encode essentially the same information, only one is likely to be incorporated into the final model. This is so because after (let us say) feature f is selected, f' will probably have low gain with respect to the model that includes f.

We will show that at least for syntactic features, the feature induction computation is of little benefit. We begin our treatment of this issue by developing the notion of *gain additivity* in the next section. In Section *Empirical Study of Gain Additivity* we present results to support this claim.

Additivity of the Gain

A natural question is whether a selected collection of features $M \subset F$ will be as informative as the sum of its parts. For instance, suppose the words *stocks* and *bonds* are both informative as triggers of the word *rose*. We might reasonably doubt that these are really independent predictors of *rose*, since *stocks* and *bonds* themselves tend to occur together. Put another way, since the gain is a numerical measure of the value of a feature, we are asking if the value of these (or any) two features, when both are used in a model, equals the sum of the individual value of each. In this section we give a theoretical treatment of this issue, introducing the notion of *additivity*.

To begin we consider why it might be plausible that the gains would add. Consider a set $M = \{f_1, f_2\}$ of just two features. By equations (8, 9, 11) above and the associated discussion we have

$$G_{f_1} = \frac{1}{N}\log\frac{P_{f_1\alpha_1^*}(C)}{P(C)}, \quad G_{f_2} = \frac{1}{N}\log\frac{P_{f_2\alpha_2^*}(C)}{P(C)}. \tag{12}$$

Let us write $P_{\bar{f}\bar{\alpha}^*}$ for the MEMD model defined by features $\bar{f} = \{f_1, f_2\}$ and exponents $\bar{\alpha}^* = \{\bar{\alpha}_1^*, \bar{\alpha}_2^*\}$, yielding a gain

$$G_{\bar{f}} = \frac{1}{N}\log\frac{P_{\bar{f}\bar{\alpha}^*}(C)}{P(C)}. \tag{13}$$

Note that $\bar{\alpha}_1^*$, $\bar{\alpha}_2^*$ are decidedly *not* necessarily equal to α_1^* and α_2^*, as determined by equation pair (12) above.

Now let us write

$$\frac{P_{\bar{f}\bar{\alpha}^*}(C)}{P(C)} = \frac{P_{f_1\alpha_1^*}(C)}{P(C)}\,\frac{P_{\bar{f}\bar{\alpha}^*}(C)}{P_{f_1\alpha_1^*}(C)} \tag{14}$$

which yields

$$G_{\bar{f}} = G_{f_1} + \frac{1}{N}\log\frac{P_{f_1 f_2\bar{\alpha}_1^*\bar{\alpha}_2^*}(C)}{P_{f_1\alpha_1^*}(C)}. \tag{15}$$

Here we have written $P_{\bar{f}\bar{\alpha}^*}(C)$ out in full as $P_{f_1 f_2\bar{\alpha}_1^*\bar{\alpha}_2^*}(C)$, and simplified using the definition of G_{f_1}. Thus the heart of the matter is how well the second term on the right hand side is approximated by G_{f_2}. We proceed to give a sufficient condition to ensure that the equation $G_{\bar{f}} = G_{f_1} + G_{f_2}$ is exact.

The key idea we will need for our argument is the *potential activation vector* of a feature f with respect to a corpus C, written $\bar{\phi}^C(f)$. In what follows we will relate

word pair		*gain* (mbits)	e^{α^*}	*active* ($\times 10^6$ words)
(	)	0.708	3.6	931
Mr.	*Mr.*	0.678	1.8	3351
Japanese	*Japanese*	0.472	8.1	276
his	*Mr.*	0.431	1.7	2501
Reserve	*Fed*	0.371	18.0	137
Motors	*G.*	0.264	9.8	140
Gorbachev	*Soviet*	0.261	15.6	104
Pennzoil	*Texaco*	0.257	47.7	69
Tokyo	*Japanese*	0.211	7.0	136
Exporting	*OPEC*	0.207	46.3	56
Lambert	*Drexel*	0.198	19.4	73
currency	*dollar*	0.191	3.9	233
prices	*million*	0.160	0.5	484
auto	*Ford*	0.153	10.6	75
Eastman	*Kodak*	0.148	163.2	31

trigger features

word pair		*gain* (mbits)	e^{α^*}	*active* ($\times 10^2$ words)
⟨*s*⟩	⟨/*s*⟩	9.639	10.7	16937
said	.	4.919	10.4	1561
⟨*s*⟩	*said*	2.920	3.8	1969
would	.	1.112	17.5	290
dollars	*cents*	0.934	70.6	230
yesterday	*closed*	0.261	67.1	39
rose	*to*	0.226	4.4	121
rose	*from*	0.197	5.3	84
its	*unit*	0.176	14.2	37
allow	*to*	0.164	38.1	36
A	*spokesman*	0.145	29.3	36
increased	*percent*	0.123	29.6	30
yield	*percent*	0.091	78.8	17
prevent	*from*	0.067	89.3	9
pence	*cents*	0.062	221.8	7

link features

Table 1: Selected Trigger and Link Features. These features are ranked according to gain, reported here in thousandths of a bit (mbits). The third column, e^{α^*}, represents the approximate boost (or deflation) of probability given to the second word of each pair, when the feature is active. The rightmost column lists the feature's empirical activation. Note that trigger features are active far more often than link features. The units used for column *active* differ by 10^4 words.

$\bar{\phi}^C(f)$ and the gain G. Note that both quantities are defined relative to a corpus. For typographic clarity, we elide the superscript from $\bar{\phi}^C$, with the understanding that our claims hold only when $\bar{\phi}$ and G share the same underlying corpus C.

As above, suppose the corpus C contains N positions, numbered 0 through $N-1$, with h^i the history at position i. Then we define $\phi_i(f)$, the ith component of $\bar{\phi}(f)$, by

$$\phi_i(f) = \begin{cases} 1 & \text{if } \exists w \in V \text{ such that } f(w\ h^i) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Thus, $\phi_i(f)$ is non-zero if and only if feature f does or could attain the value 1 at corpus position i. More succinctly, $\phi_i(f) = \max_{w \in V} f(w\ h^i)$; note that ϕ_i does not depend upon the word w^i that actually appears at position i. The potential activation vector $\bar{\phi}(f)$ is then defined componentwise as an N-element vector, the ith component of which is $\phi_i(f)$.

Lemma 1 *Let f_1 and f_2 be binary-valued features. If $\bar{\phi}(f_1) \cdot \bar{\phi}(f_2) = 0$, then*

$$G_{f_1 f_2} = G_{f_1} + G_{f_2}. \tag{17}$$

Proof: The set of corpus positions $I = \{0 \ldots N-1\}$ can be split into three sets

$$\begin{aligned} I_{f_1} &= \{i \mid \phi_i(f_1) = 1\} \\ I_{f_2} &= \{i \mid \phi_i(f_2) = 1\} \\ I_0 &= \{i \mid \phi_i(f_1) = 0 \text{ and } \phi_i(f_2) = 0\}. \end{aligned}$$

Since $\bar{\phi}(f_1) \cdot \bar{\phi}(f_2) = 0$, these three sets are mutually disjoint; by definition they cover I.

Observe that G_{f_1} depends only upon the positions that appear in I_{f_1}; likewise G_{f_2} depends only upon I_{f_2}. Moreover the maximization of $\bar{\alpha}_1$ in $\operatorname{argmax}_{\bar{\alpha}} \log P_{f_1 f_2 \bar{\alpha}_1 \bar{\alpha}_2}(C)$ depends only upon positions appearing in I_{f_1}, since the log sum splits into independent terms just as I splits into I_{f_1}, I_{f_2} and I_0. Indeed, the term that corresponds to I_{f_1} is precisely the nonconstant term in the maximization that yields α_1^*; thus $\bar{\alpha}_1^* = \alpha_1^*$. A similar argument holds for $\bar{\alpha}_2^*$. A simple calculation then yields the desired result. ∎

When $\bar{\phi}(f_1) \cdot \bar{\phi}(f_2) = 0$, we write $f_1 \perp\!\!\!\perp f_2$. If $M = \{f_j\}$ is a collection of features, and g is a feature such that $g \perp\!\!\!\perp f_j$ for each $f_j \in M$, we write $g \perp\!\!\!\perp M$. Finally, if for every $f_j \in M$, we have $f_j \perp\!\!\!\perp (M \backslash \{f_j\})$, where the right hand side stands for M with f_j removed, then we say the collection M is *ϕ-orthogonal.*

Theorem 1 *Let M be a ϕ-orthogonal collection of features. Then*

$$G_M = \sum_{f_j \in M} G_{f_j}. \tag{18}$$

Proof: By induction on the size of M. ∎

Of course, we do not mean to suggest that many practical feature collections are ϕ-orthogonal. And it should be clear that since ϕ is defined relative to a particular corpus C, it is entirely possible that a collection M that is ϕ-orthogonal for one corpus may not be for another.

Tests and Results

Our experiments were designed to address three issues. First, given a training corpus over 20 times larger than the Switchboard transcripts used in [3], we were curious to see how large a model we could feasibly train. Second, we wanted to conduct an experimental study of the gain as a criterion for feature selection, compared to empirical activation and mutual information. Third, we wished to investigate the addivity of the gain. To answer these questions, we trained a number of models, varying the number of features, and the selection criterion, and measuring the resources the training consumed, and the perplexities of the resulting models.

Models Trained

We trained a total of fifteen models; in all cases we trained on the complete corpus. We performed MEMD training using the *improved iterative scaling* algorithm of [5], using the relative change in conditional perplexity, R_t, as a stopping criterion. This quantity is defined as $R_t = (\pi_{t-1} - \pi_t)/\pi_{t-1}$, where π_t is the conditional perplexity (that is, $\pi_t = P_t(\mathcal{T} \mid \mathcal{K})^{-1/N}$, where $P_t(\mathcal{T} \mid \mathcal{K})$ is the corpus probability according to our model at training iteration t). We required $R_t < .01$ before stopping. We write π_M for the perplexity of the final model M.

Table 2 summarizes our models, the characteristics of the training computation, and the model perplexities. Column t_{seg} is the time to complete one improved iterative scaling iteration on one segment (1/40th) of the complete training corpus on an IBM RS/6000 POWERstation, model 590H. Column *mem* is the total data memory required to process one segment of the corpus. The columns for G_M, $\hat{G}_M$ and δ_M are discussed below.

We draw three conclusions from the perplexity results in this table. First, models constructed only with 2link features have lower perplexity than those constructed only with 2trig features, when we compare models of the same size. This is evident in the comparison between 10k.2trig and 10k.2link, and also between 50k.2trig and 50k.2link. We believe this reflects the higher additivity of 2link gains, a point we discuss further in the next section. However, another possible explanation is that the training converges faster for 2link features than for 2trig features.

Second, the best performance is obtained by including both feature types. This can be seen by comparing among models 10k, 10k.2trig and 10k.2link, and likewise among 50k, 50k.2trig and 50k.2link.

Finally, models selected by gain do better than those selected by mutual information or empirical activation. This is evident from the perplexities of models 10k, 10k.mi and 10k.eact, and likewise 50k.2link, 50k.2link.mi and 50k.2link.eact.

Empirical Study of Gain Additivity

To investigate the additivity of the gain, we first computed the actual gain of each model M, defined as

$$G_M = \frac{1}{N} \log \frac{P_M(C)}{P(C)}. \qquad (19)$$

Here $P_M(C)$ is the probability of the corpus, as given by model M. Note that the gain and the perplexity are related by $G_M = \log(\pi_q/\pi_M)$, where π_q is the perplexity of the base model. We then compared G_M with the gain as predicted by summing the individual feature gains, written

$$\hat{G}_M = \sum_{f \in M} G_f. \qquad (20)$$

Table 2 reports both these values, and also their *defect* δ_M, which is defined as $\delta_M = \hat{G}_M - G_M$. The defect measures the extent to which the model fails to realize its potential gain. The smaller the defect, the more nearly the gains of the underlying features are additive.

We have argued that the additivity of the gain is related to the ϕ-orthogonality of the feature set, and we believe this is borne out by the figures in the table. Trigger features are clearly highly non-additive. This is to be expected, since in any collection of gainful trigger features, we would expect a large fraction of them to be potentially active at any one position.

By contrast, the link features appear to be very nearly additive. Moreover, the defect δ_M does not grow monotonically with the number of link features in the model. It would seem that the stanza of 300,000 lower-ranked link features are more nearly ϕ-orthogonal than the 200,000 higher-ranked ones. This is reasonable, since on balance the lower-ranked features are probably less often active, hence more likely to act independently of one another.

Summary

In this paper we have investigated the use of gain as a criterion for selecting features for MEMD language models. We showed how the gain of a feature arises naturally from consideration of the feature's predictive value in an MEMD model, compared to the predictions made by the base model. We argued that the gain is the prefered figure of merit for feature selection, since it identifies features that improve upon the base model.

We then applied this statistic to the problem of selecting features for a dependency grammar language model. We showed that when comparing models constructed from the same number of features, using gain as the figure of merit yields models of lower perplexity than either empirical activation of mutual information. Moreover, among models built exclusively from either trigger or link features, but having the same number of features, those built exclusively from links had lower perplexity. However, we achieved the lowest perplexity when we picked the most gainful features without regard to their type.

Finally, we showed that sets of link features have very low gain defect; this is defined as the gap between the set's true and predicted perplexity gains, where the prediction is the sum of individual feature gains. Thus the computationally expensive feature induction procedure appears dispensable, at least for link features.

model name M	t_{seg} (hrs)	mem (MB)	perplexity π_M	actual, predicted gain G_M (bits)	$\hat{G}_M$ (bits)	defect δ_M
baseline (q)			26.764			
10k	.5	20	22.769	.233196	.558733	.325537
10k.mi	.3	19	24.195	.145558	.159312	.013754
10k.eact	1.6	23	25.860	.049545	.143026	.093481
10k.2trig	.8	20	24.483	.128487	.454672	.326185
10k.2link	.4	18	23.835	.167206	.202876	.035670
50k	2.4	37	21.647	.306100	1.140826	.834726
50k.2trig	2.6	38	23.706	.175015	1.007069	.832054
50k.2link	.9	21	23.114	.211472	.256284	.044812
50k.2link.mi	.8	21	23.379	.195054	.213165	.018111
50k.2link.eact	.9	21	23.324	.198452	.208937	.010485
100k	4.2	64	21.212	.335386	1.524190	1.188804
100k.2link	1.2	25	22.805	.230900	.278472	.047572
150k.2link	1.4	28	22.607	.243499	.291138	.047639
200k.2link	1.6	32	22.507	.249903	.299675	.049772
500k.2link	3.8	53	22.232	.267657	.316176	.048519

Table 2: Model Features, Training Characteristics, Perplexities, Gains. Models are named by the following convention. The first part of the name gives the number of features; the letter k denotes a factor of 1,000. Thus *10k* is a model built of the 10,000 highest-ranking features of the candidate set F. The notation *2trig* or *2link* means that we used only trigger or link features respectively. Thus *10k.2link* is built of the 10,000 highest-ranking 2link features of F. Additional letters identify the figure of merit used for the ranking: *eact* stands for empirical activation, *mi* stands for mutual information. If neither appears, the figure of merit was the gain.

We hasten to point out that our results concern perplexity only. It remains to be seen if these conclusions carry over to word error rate, in a suitable speech recognition experiment.

Acknowledgements

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," *Proc. 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] C. Chelba, et. al., "Structure and Performance of a Dependency Language Model," *Proc. of Eurospeech '97*, Rhodes, Greece, September 1997.

[4] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, L. Ureš, "Inference and Estimation of a Long-Range Trigram Model," *Second International Colloquium on Grammatical Inference*, Alicante, Spain, September 1994.

[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] R. Lau, R. Rosenfeld, S. Roukos, "Trigger-Based Language Models: a Maximum Entropy Approach," *Proc. of the International Conference on Acoustics, Speech and Signal Processing*, pp II: 45–48, Minneapolis, MN, April 1993.

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

[8] H. Printz, "Fast Computation of Maximum Entropy / Minimum Divergence Feature Gain," submitted to *International Conference on Speech and Language Processing*, Sydney, Australia, September 1998.

[9] 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.

[10] A. Ratnaparkhi, J. Reynar, S. Roukos, "A Maximum Entropy Model for Prepositional Phrase Attachment," *Proc. of the Human Language Technology Workshop*, Plainsboro, NJ, March 1994.

[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.