WS96 Project Report
DEPENDENCY LANGUAGE MODELING

Andreas Stolcke (SRI, Team Leader)
Ciprian Chelba (JHU)
David Engle (DoD)
Victor Jimenez (Univ. Politecnica Valencia)
Lidia Mangu (JHU)
Harry Printz (IBM)
Eric Ristad (Princeton)
Roni Rosenfeld (CMU)
Dekai Wu (Hong Kong UST)
collaborators
Fred Jelinek (JHU)
Sanjeev Khudanpur (JHU)

March 19, 1997

Abstract

This report summarizes the work of the Dependency Language Modeling group at the 1996 Summer Speech Workshop at the Center for Language and Speech Processing at Johns Hopkins University (WS96). We motivate and describe a novel statistical language model based on the syntactic dependencies between words. The model is formulated in the maximum entropy framework, which expresses statistical constraints on the frequencies of various type of dependencies, as well the standard N-gram statistics. We describe how this model was applied to the recognition of spontaneous English speech from the Switchboard corpus. Because of implementation constraints, only a reduced version of our model could be tested. The model gave a modest improvement over an N-gram baseline model. A byproduct of the project is the Maximum Entropy Modeling Toolkit (MEMT), a freely available software package for domain-independent maximum entropy modeling.

1 Introduction

Current language models for speech recognition are based on the statistics of N-grams, which are tuples of N adjacent words. In an N-gram model, a given word is predicted by its $N-1$ immediate predecessors. For the case $N = 3$, the so-called trigram model [10], this method is both computationally feasible and remarkably successful. The success of this approach indicates that for predicting a given word, its two immediate predecessors contain much useful information.

To improve on today's N-gram language models (LMs) we can pursue the following avenues:

(1) Use better models of linguistic structure

(2) Use more of the word history to make a prediction for the next word

(3) Use better estimators, smoothing approaches, etc.

Our group's primary focus was avenue (1): we built a LM based upon linguistic relationships between words, as determined by a *dependency grammar*. But we also pursued avenues (2) and (3), to the extent that this arose naturally from (1). Because a word's linguistic dependencies often reach further into the past than the one or two words of N-gram methods, our model effectively uses more history than a bigram or trigram model; in this way we explored avenue (2). And as a natural consequence of this approach, our model incorporated too much conditioning information to estimate its parameters by the ratios-of-counts method used for most N-gram models. This drove us to investigate avenue (3): we supplanted the ratios-of-counts method with the technique of *maximum entropy modeling*. This method permits the estimation of conditional probabilities that incorporate a large number of predictors, without fragmenting the model's training data.

The remainder of this report is structured as follows. Section 2 gives a more detailed account of our motivation and describes the overall approach. Section 3 explains the maximum entropy (ME) modeling approach and gives an overview of the software toolkit developed to implement general ME models. Section 4 gives a detailed description of the dependency model in its ME formulation. Section 5 discusses various approaches to the selection of dependency constraints. Section 6 describes speech recognition experiments and results using the model. Section 7 summarizes the work so far and discusses possible extensions.

2 Motivation and Approach

We begin by examining the shortcomings of a standard N-gram model using a specific example. Let S denote the sentence

$$\textit{<s> the dog I heard last night barked again </s>}$$

where the tokens <s> and </s> denote sentence begin and end, respectively. By the laws of conditional probability, we have the exact decomposition

$$P(S) = P(\textit{the} \mid \textit{<s>})P(\textit{dog} \mid \textit{<s> the})P(\textit{I} \mid \textit{<s> the dog}) \times \ldots \times P(\textit{</s>} \mid \textit{<s> the} \ldots \textit{barked again}) \qquad (1)$$

Unfortunately, this does not constitute a usable scheme. If V is the size of the vocabulary, then the ith generic factor in such a decomposition requires the estimation and storage of $V^{i-1}(V-1)$ independent parameters, and since typically $V \approx 25,000$, this is infeasible. Even if space were not an issue, we would not have enough training data to estimate such a model accurately.

N-gram models avoid this difficulty by retaining only the $N-1$ most recent words of history. For example, the trigram model yields the following approximation

$$P(S) \approx P(\textit{the} \mid \textit{<s>})P(\textit{dog} \mid \textit{<s> the})P(\textit{I} \mid \textit{the dog}) \times \ldots \times P(\textit{</s>} \mid \textit{barked again}) \qquad (2)$$

But this approach has two significant drawbacks. First, it is frequently linguistically implausible, blindly discarding relevant words that lie N or more positions in the past, and retaining words of little or no predictive value simply by virtue of their recency. Second, such methods make inefficient use of the training corpus, since the distributions for two histories that differ only by some triviality cannot pool data.

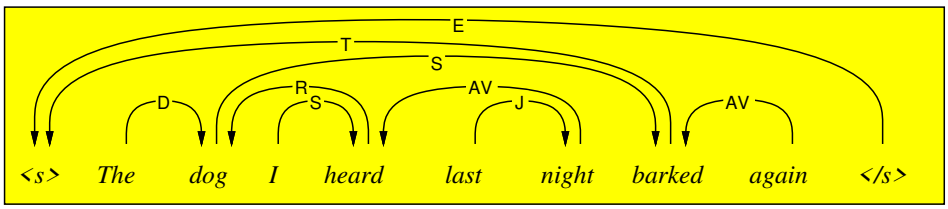

Figure 1: Link Structure for Example Sentence.

2.1 Using Linguistic Structure

There have been many attempts to exploit the linguistic structure of sentences for language modeling. The first forays in this direction were based on stochastic context-free grammars [11], which assume that words and phrases of a sentence form a tree-like structure. While these approaches have been successful in some applications [33, 17, 13], language models based on context-free grammars still cannot compete with N-gram models in domains with large vocabularies, relatively unrestricted speech, and large amounts of training data (such as the Wall Street Journal domain).

The main deficiency of context-free grammar models is that they condition their predictions on phrase labels, and not directly on words. But it is well-known that words themselves are the best predictors of other words. Current approaches in linguistically motivated language modeling therefore strive to use grammars that are *lexicalized*, that is, where all structural elements are directly anchored in words. This is true of so-called *dependency grammars* and a variant known as *link grammars* [29, 16].

A dependency grammar or link grammar represents grammatical structure by connecting linguistically related words with labeled arcs. The labels denote the type of grammatical relation, such as 'subject of verb' or 'adjective modifying noun'. Consider the link structure for the example sentence, shown in Figure 1.

The intuition underlying a dependency language model is that the grammatical structure can guide us to the words elsewhere in the sentence that are relevant to the prediction of a particular word. For example, the word *barked* is the main verb in the example above, bound to its subject by the link type S. Both the link type and the word *dog* at the other end should therefore help predict *barked*.

The earlier work reported in [5] employed just such an approach, using a link grammar. However, the particular grammar used there did not have a strong linguistic motivation, since it was based exclusively upon informative word pairs, and not upon their parts of speech. Moreover, at any given point, the model used no more than two words as predictors. As we discuss further below, our model can integrate many more predictors.

2.2 Parsimonious Use of the Larger History

It is evident from the example that links may connect words that lie outside of N-gram range, where N usually equals 2 or 3. In this case, N would have to be 5 to capture the word *is*, which the grammatical structure has identified as relevant for predicting *like*. But as we argued above, it is not feasible to build N-gram models with this much context. The problem is that while we want to use *some* of the words in the larger history, we also want to ignore many or most of the words in the history as not relevant for predicting the next word. The dependency structure should help us identify the relevant words.

2.3 Efficient Use of Data and Parameters

As we mentioned earlier, another problem with N-gram models is that they fragment the training data into bins, one for each distinct N-gram. This makes training data that is similar linguistically look very different to the model. Consider variants of the above sentence

the dog barked again
the dog I heard last night barked again
the dog I heard while I was taking out the garbage barked again

All these variants generate different N-grams ending in *barked*. The conditional probabilities for *barked* are therefore parameters that have to be independently estimated.

But to the dependency model, these sentences look very much alike. In particular, they all share the S link between *dog* and *barked*. If the model is properly parameterized, all the training examples should contribute to the estimate of $P(barked|\cdot)$ in similar contexts.

Such a parameterization is not trivial to achieve, however. Let w be the word we want to predict, and suppose that w is connected to a word x in the past by a link with label L. Though we can estimate a conditional probability $P(w \mid xL)$ as a ratio of counts, in general w will bear many links $x_1L_1, x_2L_2, \ldots x_nL_n$ to words in the past. To use all this conditioning information we need to estimate a probability

$$P(w \mid x_1L_1,\ x_2L_2,\ \ldots\ x_nL_n)$$

and we are once again faced with the problem of integrating a large number of predictor variables, without fragmenting our training data to the point of uselessness. Moreover, since w's one or two immediate predecessors are known to be good predictors, we may want to include them as well in the list of conditioning variables, further exacerbating the problem. It is for this reason that we used the maximum entropy modeling approach, described in Section 3, which provides an elegant solution to this difficulty.

2.4 Our Approach

We are now in a position to outline our approach to the actual building of a dependency-based language model; the following sections describe the various elements of that approach in more detail.

The goal is to obtain a model for $P(S)$, the probability of a sentence, or sequence of words, S. However, the model should take into account not just the sequence of words, but also its grammatical structure K. In the case of a link or dependency grammar, K is the *linkage*, or union of all links making up the parse, as depicted in Figure 1. Thus, the goal of the model is to compute the joint probability $P(S, K)$.

Since sentences can be ambiguous with respect to their grammatical analysis, we have to sum over all possible parses to obtain the sentence probability,

$$P(S) = \sum_K P(S, K). \tag{3}$$

Most sentences exhibit some degree of syntactic ambiguity, meaning that we would have to evaluate at least several of the highest probability parses to closely approximate the above sum. Instead, we chose to evaluate only the single most likely parse K^*. That is, we make the approximation

$$P(S) \approx \max_K P(S, K) = P(S, K^*), \tag{4}$$

where K^* is the parse yielding the highest overall probability.

We want to use the model $P(S)$ to compare only a few of the most likely recognition hypotheses (see Section 6.3), which are similar in the number of words and even share most of their words. Therefore, while $P(S, K^*)$ is not a good approximation of $P(S)$ we can assume that the probabilities of all recognition hypotheses to be compared will be underestimated by roughly the same factor, which can therefore be ignored.

2.4.1 Parsing and Tagging

We used an existing state-of-the-art parser developed by Michael Collins [2] to find the best parse K^* and then simply evaluate our model on the pair (S, K^*). In fact, we let the parser choose K^* according to *its own* model of parse probability, which is not necessarily the one that maximizes $P(S, K)$ according to *our* model. We felt justified in this shortcut since the parser used had been independently evaluated against hand-labeled data and found to have very high accuracy (see Section 6 for details).

Collins's parser does not operate directly on word sequences, however. Rather, it expects a sequence of part-of-speech tags T as input. It finds K^* that maximizes $P(S, K^*, T)$. We relied on another existing tool, Adwait Ratnaparkhi's maximum entropy part-of-speech tagger [25] to compute the best tagging T^* for a

given sentence S, and then applied the parser to T^*. Note that this represents another approximation since we are not jointly optimizing T and K. Again, we felt confident of this approximation because we were able to verify the high accuracy of the tagger on our data (Section 6).

We also had to deal with another technical inconvenience involving the parser. Collins's parser does not in fact output link structures of the type needed for our modeling, but rather computes traditional tree-structured parses. However, a tree parse can be converted to a dependency structure in the following way.

The idea is that each word has a dependency on the *head word* of the immediately enclosing constituent. Thus, adjectives are linked to the head nouns of the noun phrases they appear in. The nouns themselves are linked to the preposition or head verb they are the object (or subject) of. Head words are identified in a parse tree by simple heuristic rules that identify the head component for each type of constituent. For example, there is a rule stipulating that a noun phrase dominating a sequence of determiner, adjective(s), and noun has the noun as its head. This way, a link structure can be uniquely derived from each tree parse.

We also need to label the links as to the type of syntactic relation they represent. Since link labels will be used to condition the words appearing at the ends of the links, we can choose them in any way that makes useful grammatical distinctions. But these distinctions need not correspond to traditional linguistic ideas. A simple kind of label derived from the tree parse is the triplet of phrase categories appearing at the apex of the path linking the two words in the parse tree (the topmost node in the path, and its two children).

2.4.2 Maximum Entropy Modeling

To formulate and implement the probability model for $P(S, K)$, we used the *maximum entropy* (ME) framework. This framework frees us from the need to decompose the probability of $P(\text{word}|\text{history})$ into further conditional probabilities, thus avoiding the problems mentioned in Section 2.3.

Instead, we can focus on identifying statistics that reflect inherent constraints on the distribution of words. For example, we may choose the relative frequencies with which certain words appear linked (with specific link types). Once we have identified the statistics we believe are important in the domain, the distribution $P(S, K)$ is defined implicitly by the following two stipulations:

- The distribution should match the values for the statistics observed in the training data.
- Among all the models having this property, choose the one with the highest entropy.

The first stipulation says that our model should match the training data in all aspects deemed relevant; the second stipulation says it should be as noncommittal as possible in all other respects.

Section 3 describes how this implicit definition of a distribution can be made constructive, whereas Section 4 describes the types of statistics chosen to characterize our language model. In past applications of maximum entropy the modeling and the implementation have been intertwined for efficiency, leading to solutions that are hard to understand and debug. We therefore decided to separate these two aspects completely by creating a *Maximum Entropy Modeling Toolkit* that can train and evaluate ME models in a generic encoding, suitable for any domain. The model described in Section 4 is then mapped into this encoding. The encoding for a particular domain is not necessarily a trivial matter, and reflects the structural complexity of the domain. Still, the separation of tasks makes the overall implementation effort much less cumbersome.

3 Maximum Entropy Modeling

The maximum entropy framework [8, 9] is a powerful method for building statistical models. It is expressive, allowing modelers to easily represent their special insights into the data generating machinery. It is statistically efficient, because it models the intersection of complex events without increasing the number of parameters or fragmenting the training data. And it provides strong models, models that outperform their traditional variants with much less tweaking. For example, the maximum entropy trigram outperforms both the interpolated trigram [12] and the backoff trigram [14] in test set perplexity as well as in speech recognizer word error rate. For all these reasons, the maximum entropy framework has become the framework of choice for statistical language modeling [18, 1, 28].

This section has a dual purpose. First, it reviews the basics of maximum entropy modeling, as required to understand the discussion of our language model in Section 4. Second, it describes a generic software tool for training and using maximum entropy models—the Maximum Entropy Modeling Toolkit (MEMT), first designed and implemented by Eric Ristad at WS96.

3.1 The Maximum Entropy Modeling Framework

The fundamental problem of statistical modeling is to induce a joint probability model $p : X, Y \rightarrow [0, 1]$ from a finite corpus of observations $\{(x_1, y_1), \ldots, (x_T, y_T)\}$ drawn from a discrete joint domain X, Y.

In the maximum entropy approach to statistical modeling, we first define a set G of k binary features on X, Y:

$$G = \{g_i : X, Y \rightarrow \{0, 1\} \text{ for } i = 1, \ldots, k\} \tag{5}$$

For example, looking ahead to Section 4, useful features will include whether a certain dependency holds between two words, or whether two words appear next to each other.

Each binary feature $g_i(x, y)$ partitions the joint domain X, Y into two sets: those points (x, y) for which $g_i(x, y)$ is active and those for which it is not active. Next, we choose a vector $a = (a_1, \ldots, a_k)$ of target expectations for our features. The simplest way to do this is to choose each target expectation a_i to be the empirical expectation $f[g_i]$, that is, the expectation of the feature $g_i(x, y)$ with respect to the empirical distribution $f(x, y)$ defined by the training corpus:

$$f[g_i] = \sum_{x,y} f(x, y) g_i(x, y). \tag{6}$$

3.1.1 The Maximum Entropy Distribution

Together, the feature set G and their target expectations define a class $\mathcal{P}$ of all probability distributions whose feature expectations match the target expectations,

$$\mathcal{P} = \{p : p[g_i] = a_i \text{ for all } i = 1, \ldots, k\} \tag{7}$$

where $p[g_i]$ is the expectation of $g_i(x, y)$ with respect to the distribution $p(x, y)$,

$$p[g_i] = \sum_{x,y} p(x, y) g_i(x, y). \tag{8}$$

If the target expectations are the empirical expectations, then $\mathcal{P}$ contains all distributions that are equivalent to the empirical distribution defined by our training corpus, when viewed through the eyes of our features G. Defining the features and choosing their target expectations is the modeler's art.

Given such a class $\mathcal{P}$, we would like to find the distribution $p^*(x, y)$ in $\mathcal{P}$ that maximizes the entropy $H(p)$ with respect to all distributions in $\mathcal{P}$:

$$H(p) = -\sum_{x,y} p(x, y) \log p(x, y) \tag{9}$$

The maximum entropy distribution $p^*(x, y)$ is the one that is most faithful to our constraints, because it makes no additional assumptions beyond what has been specified [8, 9, 3]. We also need a compact way to represent this distribution.

3.1.2 The Exponential Model Class

Now consider the class $\mathcal{R}$ of all exponential models $m(x, y)$ defined over our features G

$$\mathcal{R} = \{m : m(x, y) = \frac{r(x, y)}{Z}\} \tag{10}$$

whose numerators $r(x,y)$ are a product of exponentials, $\alpha_i = e^{\lambda_i}$, and whose denominators Z are chosen to ensure that each m is a probability function.

$$r(x,y) = \prod_i \alpha_i^{g_i(x,y)} \tag{11}$$

$$Z = \sum_{x,y} r(x,y) \tag{12}$$

This model class $\mathcal{R}$ has as many free parameters as there are features. Fortunately, the intersection of the class $\mathcal{R}$ of exponential models over G with the class $\mathcal{P}$ of desired distributions is nonempty. Consequently, at least one of our desired distributions has a representation as an exponential model. Even better, the intersection of $\mathcal{P}$ and $\mathcal{R}$ contains the maximum entropy distribution $p^*(x,y)$. Best of all, the intersection of $\mathcal{P}$ and $\mathcal{R}$ is unique! Therefore, we need only find a single exponential model $m^*(x,y)$ in $\mathcal{R}$ that satisfies the linear constraints $\mathcal{P}$, and we are assured that this model is the distribution $p^*(x,y)$ in $\mathcal{P}$ of maximum entropy. Now if our target expectations are the empirical expectations, then it also happens that $m^*(x,y)$ is the maximum likelihood model in $\mathcal{R}$, and so the thorny problem of finding the maximum entropy distribution $p^*(x,y)$ in $\mathcal{P}$ reduces to the straightforward problem of finding the maximum likelihood model $m^*(x,y)$ in $\mathcal{R}$. This is the beauty of the maximum entropy framework [15].

3.1.3 Conditional Models

Various difficulties arise when these ideas are applied to discrete time series problems. The first difficulty is that we must assign probability to strings of arbitrary length, a task for which we cannot employ joint models over finite dimensional spaces. For discrete time series, we require a conditional model $p(y|x)$ instead of a joint model $p(x,y)$. Our model class $\mathcal{R}$ becomes

$$\mathcal{R} = \{m : m(y|x) = \frac{r(y|x)}{Z(x)}\} \tag{13}$$

where

$$r(y|x) = \prod_i \alpha_i^{g_i(x,y)} \tag{14}$$

$$Z(x) = \sum_z r(z|x). \tag{15}$$

Our constraint class $\mathcal{P}$ remains unchanged, although we now require a marginal distribution $p(x)$ on the X in order to calculate the feature expectations $m[g_i]$ with respect to a joint model $m(x,y) = p(x)m(y|x)$. There are many reasonable choices here, but computational efficiency will require us to use the empirical distribution $\tilde{p}(x)$ on X.

3.1.4 Computational Tricks

This is the second difficulty, namely, that maximum entropy modeling for joint spaces is computationally infeasible. Evaluating the probability $m(x,y)$ of a joint event (x,y) according to our model requires us to enumerate the entire X,Y space in order to calculate the denominator Z. And evaluating the expectation $m[g]$ of a feature $g(x,y)$ with respect to the joint model $m(x,y)$ requires us to enumerate the entire X,Y space again.

In a speech recognition application, Y is the vocabulary of all words and X is all history equivalence classes. The vocabulary of a typical speech recognition system has over 20,000 words, and its history equivalence classes X typically consist of all word bigrams, and so the X,Y joint space contains $|Y|^3 = 8\times10^{12}$ distinct events.

The infeasibility of repeatedly enumerating this space requires us to make two simplifications. The first simplification allows us to efficiently compute the expectation $m[g_i]$ of each feature $g_i(x,y)$, while the second simplification allows us to efficiently compute the denominator $Z(x)$ of a conditional exponential model without summing over all the symbols in Y.

3.1.5 Computing Expectations

The first simplification is to use the empirical distribution $\tilde{p}(x)$ as our marginal on X, which gives us the joint model $m(x,y) = \tilde{p}(x)m(y|x)$. The statistical consequence of this simplification is to require our conditional model $m(y|x)$ to match the target feature expectations on the observed contexts only. The computational consequence of this simplification is quite significant, however. Now our feature expectations are calculated as

$$\begin{aligned} m[g_i] &= \sum_{x,y} m(x,y)g_i(x,y) && (16) \\ &= \sum_{x,y} \tilde{p}(x)m(y|x)g_i(x,y) && (17) \\ &= \sum_{x} \tilde{p}(x) \sum_{y} m(y|x)g_i(x,y) && (18) \end{aligned}$$

The empirical distribution $\tilde{p}(x)$ is zero for novel contexts, and so the outer sum need iterate only over the observed contexts $\{x : \tilde{p}(x) \neq 0\}$. There are at most T such contexts in a corpus of size T. For the interior sum we need iterate only over those symbols y for which $g_i(x,y)$ is active. For all maximum entropy language models proposed in the literature, this is bounded by a small constant, typically one. Therefore we have reduced the entire $\sum_{x,y}$ of size $|X,Y|$ to a sum over the contexts observed x, which is of worst case size T. For a trigram model constructed from 1.4 million words of the Switchboard corpus with a vocabulary of 22,511 words, the size $|X,Y| = 22,511^3 = 11,407,339,418,831$, whereas there are only 234,009 distinct contexts in the Switchboard corpus. Thus, this first simplification reduces the computation of feature expectations by seven orders of magnitude.

3.1.6 Computing Denominators

The goal of the second simplification is to efficiently compute the denominator $Z(x)$ of the conditional exponential model $m(y|x)$. In order to calculate the feature expectations during parameter estimation, we must calculate $Z(x)$ for every observed context x. Since $Z(x)$ changes after each iteration during estimation, we must calculate $Z(x)$ for every context on each iteration. A straightforward approach to computing $Z(x)$ for each context x requires us to enumerate the alphabet Y of symbols, which is infeasible when the alphabet Y is large. Here we show how to reduce the size of this computation by several orders of magnitude for typical models. Our development takes advantage of the fact that most conditional events $y|x$ will only activate marginal features—features whose activation depends only on the symbol y and is independent of the context x. This property holds for models whose conditional space $Y|X$ is large and whose features are derived directly from a finite corpus of examples.

We begin by partitioning the set G of features into two sets: the marginal features G_m and the conditional features G_c. A marginal feature $g_i(x,y)$ is a feature whose activation depends only on the future y, completely independent of the context x. That is, $g_i(x,y)$ is a marginal feature if and only if $g_i(x,y) = g_i(w,y)$ for all w, x, and y. Conversely, $g_i(x,y)$ is a conditional feature if and only if $g_i(x,y) \neq g_i(w,y)$ for some w, x, and y.

Next, we impose two partitions on the set Y of symbols. The first partition is to divide the symbols Y into the set Y_m of symbols for which some marginal feature is active

$$Y_m = \{y : g_i(y) \neq 0 \text{ for some } g_i \in G_m\} \quad (19)$$

and the set $Y - Y_m$ for which no marginal feature is active. The second partition is to divide the symbols Y into the set Y_x of symbols for which some conditional feature is active in the context x

$$Y_x = \{y : g_i(x,y) \neq 0 \text{ for some } g_i \in G_c\} \quad (20)$$

and its complement $Y - Y_x$, the symbols for which no conditional feature is active in the context x. Note that Y_m and Y_x may intersect, as may their complements $Y - Y_m$ and $Y - Y_x$.

These partitions allow us to simplify the $Z(x)$ summation as follows.

$$
\begin{aligned}
Z(x) &= \sum_{y \in Y} r(y|x) && (21)\\
&= \sum_{y \in Y - Y_m} r(y) + \sum_{y \in Y_m} r(y) + \sum_{y \in Y_x} (r(y|x) - r(y)) && (22)\\
&= \sum_{y \in Y - Y_m} 1 + \sum_{y \in Y_m} r(y) + \sum_{y \in Y_x} (r(y|x) - r(y)) && (23)\\
&= |Y - Y_m| + \sum_{y \in Y_m} r(y) + \sum_{y \in Y_x} (r(y|x) - r(y)) && (24)
\end{aligned}
$$

where $r(y)$ is the product of the marginal parameters that apply to y,

$$r(y) = \prod_{g_i \in G_m} \alpha_i^{g_i(y)} \tag{25}$$

and $\sum_{y \in Y_m} r(y)$ is a constant independent of x. Note that $r(y)$ is unity if no marginal features apply to y.

These machinations reduce the computation of $Z(x)$ from a sum of $|Y|$ terms to a sum of $2|Y_x|$ terms. For the Switchboard trigram model introduced above, $|Y|$ is 22,511 whereas $|Y_x|$ is less than 204 on average, which gives us a net 55-fold speedup.

3.2 The Maximum Entropy Modeling Toolkit

Unfortunately, three obstacles have impeded the adoption of the maximum entropy framework in the LVCSR community. The first obstacle is that the theory of maximum entropy modeling is not widely understood in the LVCSR community outside of IBM Watson Research. The second obstacle is that a straightforward implementation of the maximum entropy framework will be computationally infeasible. For example, a recent implementation of a maximum entropy trigram for the Switchboard corpus [32] required over four months to train on a top-of-the-line workstation.[1] An efficient implementation of the maximum entropy framework requires careful programming and many little-known computational tricks. Consequently, there have been only a handful of successful maximum entropy models in the LVCSR community, and these have all occurred at IBM Watson Research [18, 24, 1]. The third obstacle to the use of maximum entropy techniques in the LVCSR community is that it is not obvious how to support arbitrary maximum entropy models in a clean and efficient manner. Consequently, no general-purpose software for maximum entropy modeling currently exists.

At WS96, we overcame these three obstacles by designing and implementing a general purpose Maximum Entropy Modeling Toolkit [26]. Our design was shaped by the four criteria of generality, efficiency, interoperability, and robustness. These criteria led to a file-based interface consisting of three file formats, along with three primary executables. Our prototype implementation is general, easy to use, and fast. The prototype toolkit was used by two of the four WS96 groups to create a wide range of maximum entropy language models.

3.2.1 Design Criteria

The first design criterion was that the toolkit must provide clean support for arbitrary features. The beauty and utility of the maximum entropy framework derives principally from its expressiveness. If we were to restrict the expressiveness available to the modeler, then we would defeat the central purpose of the toolkit. Consequently, the toolkit must support arbitrary features in a conceptually simple manner. This criterion shaped the overall design of the toolkit and also guided our choice of parameter estimation algorithms. In particular, it led us to use the recently developed improved iterative scaling (IIS) algorithm [23] for parameter estimation rather than the classic generalized iterative scaling (GIS) algorithm [4]. GIS requires that the

[1] Sanjeev Khudanpur, personal communication.

feature activations always sum to a constant. In contrast, IIS imposes no requirements on the features and it therefore provides a simpler interface.

The second design criterion is that the toolkit implementation must be highly efficient in both time and space. It must be of comparable efficiency to the best special-purpose code, or skilled users will not use the toolkit. The principal effect of this requirement is to require a careful implementation. This requirement has only two effects on the user interface. First, the user must identify all features whose behavior depends only on the dependent variable y, that is, the marginal features. Second, the user must enumerate all joint events $\langle x, y \rangle$ for which a given feature $g_i(x, y)$ is nonzero. Lacking such an extrinsic specification for the features, an efficient implementation would not be possible.

The third design criterion is one of interoperability. The toolkit should not restrict the user's choice of machine, operating system, or programming language. This criterion excludes binary file formats and discourages software interfaces (APIs with object code linking). The requirement of interoperability suggests a file-based user interface.

The fourth criterion is that the toolkit have a robust interface. It should be easy to monitor and debug the interface between the user and the toolkit. It is difficult to debug an object code interface. Consequently, this criterion, in conjunction with the desire for interoperability, strongly advocates a file-based design.

3.2.2 Architecture

Our four design criteria mandated a simple toolkit architecture with a file-based user interface. There are three file formats and three executables. The executables are used to verify the syntactic correctness of files, to estimate the parameters of a model using the training corpus, and to evaluate the testing corpus using a model.

`me.checker [-p parameters] [-e events] [-x expressions]`

> Given a parameters file `parameters`, an events file `events`, and/or an expressions file `expressions`, the executable `me.checker` verifies that the given file(s) satisfy all syntactic requirements. Detailed errors and warnings are written to `stderr` and a summary of the verification process is written to `stdout`. The executable `me.checker` returns success (0) if and only if all files are compatible.

`me.estimate model.in events n model.out`

> Given a parameters file `model.in` and an events file `events`, the executable `me.estimate` performs `n` iterations of improved iterative scaling [23] and writes the revised parameters to `model.out` after each iteration. Convergence information is written to `stdout` on each iteration, while errors and warnings are written to `stderr`.

`me.evaluate model expressions events results`

> Given a parameters file `model`, an expressions file `expressions`, and the corresponding events file `events`, the executable `me.evaluate` writes the outcome of the computation specified in `expressions` to `results`, in nats, that is, as negative log likelihoods in base e. Errors and warnings are written to `stderr`, and status information is written to `stdout`.

A parameters file contains the parameter value $\alpha_i = e^{\lambda_i}$ and the target expectation a_i for each feature $g_i(x, y)$. An events file defines the behavior of the features G on the training corpus. It enumerates all the conditional events $y|x$ that have occurred in the training corpus as well as those whose context x has occurred in the training corpus and whose future y activates some conditional feature. An expressions file contains a series of probability expressions, each of which evaluates to a probability value. The expressions file is needed to evaluate probability of the testing corpus with a maximum entropy model.

3.2.3 Experiences with the Toolkit

The MEMT was used by both the Dependency Modeling and the Pronunciation Modeling groups at WS96. Nearly all the research effort of these two groups was devoted to modeling — to designing and implementing features, and to preparing the training and testing data. Only a small fraction of the combined group effort was devoted to implementing the maximum entropy framework. Moreover, the MEMT was quite fast. For example, a large trigram word model containing 12,412 unigram features, 36,191 bigram features, and 120,116 trigram features was trained for 10 iterations on a single Sun UltraSparc workstation in less than 2 hours, 30 minutes. During last year's workshop, training such a word trigram model would have required 4 months.

The only performance problem in the MEMT is the infeasible size of events files. An events file completely specifies the behavior of our features on the training corpus. As such, it must not only summarize the training corpus, but it must also enumerate all conditional events $y|x$ for which some conditional feature is active and the context x was observed in the training corpus. This has the advantage of simplicity, but the disadvantage of infeasibility because such an events file may contain a great many unobserved events. In the worst case, the events file is size $O(TV)$ for a training corpus containing T observations and a vocabulary of size V. In many cases, this is infeasibly large. For example, a large word trigram model induced from 1.4 million words of Switchboard text over a vocabulary of 22,511 words contains 12,413 unigram features, 80,643 bigram features, and 120,116 trigram features. The corresponding events file is 1 GB because every observed context x generates on average 204.3 activating events, of which on average only 2.0 are actually observed in the training corpus. The remaining 202.3 events per observed context do not occur in the training corpus. Thus, the events file is two orders of magnitude larger than the training corpus. The next major release of the MEMT will include a redesign of the events file format to redress this flaw.

4 Structure of the Model

We begin the description of our model by discussing the entities we manipulate, which are words and disjuncts. Then we exhibit the decomposition of the model into a product of conditional probabilities. We give a method for the grouping of histories into equivalence classes, and argue that it is both plausible and efficient. Since our model is obtained using the maximum entropy formalism, we describe the classes of constraints we imposed. Finally, we point out various practical obstacles we encountered in carrying out our plan, and discuss the changes they forced upon us.

4.1 Elements of the Model

Figure 2 depicts a dependency grammar parse K of our example sentence S, redrawn here for convenience. Note that (1) the linkage is a connected graph, (2) each graph edge is directed and bears a label, (3) the nodes of the graph are the words of the sentence, (4) the outdegree of every node, except <s>, is exactly 1, and (5) the graph is embedded in the half-plane above the sentence with no crossing edges. As motivated in Section 2.4, our aim is to develop an expression for the joint probability $P(S, K)$.

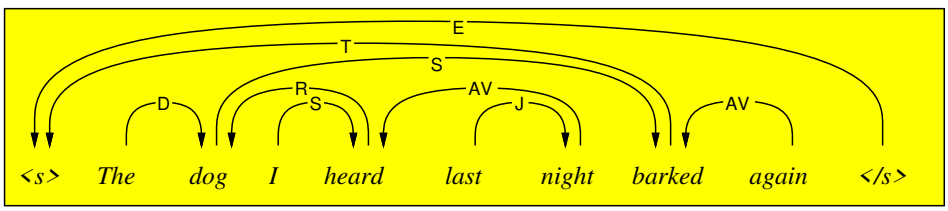

Figure 2: A Sentence S and its Linkage K.

For our purposes, every sentence S begins with the "word" <s>, and ends with the "word" </s>. We use angle brackets because these objects are of course not really words, though mathematically our model treats them as such. They are included for technical reasons: the start marker <s> functions as an anchor

for every parse, and the end marker </s> ensures that the function $P(S, K)$ sums to unity over the space of all sentences and parses.

A directed, labeled graph edge is called a *link*, and denoted L. Formally, each L consists of a triple of parse-node tags, as discussed in Section 2.4.1, plus an indication of the link's direction; we depict them more simply here for clarity. A link L that connects words y and z is called a *link bigram*, and written yLz.

Each word in the sentence bears a collection of links, emanating from it like so many flowers grasped in a hand. We refer to this collection as a *disjunct*, denoted d. A disjunct is a rule that shows how a word must be connected to other words in a legal parse. In linguist's parlance, a word and a disjunct together constitute a fully specified lexical entry.

For instance, the disjunct atop *dog* in Figure 2 means that it must be preceded by a determiner, and followed by a relative clause and the verb of which it is the subject, in that order. Intuitively, a disjunct functions as a highly refined part-of-speech tag. Note that in different sentences, or in different parses of the same sentence, a given word may bear different disjuncts, just as the word *dog* may function as a subject noun, object noun, or verb. A disjunct d is defined formally by two lists, *left*(d) and *right*(d), respectively the links it bears to the left and right.

4.2 Decomposition of the Model

Let us write w^i for the ith word of S, and d^i for its disjunct in a given linkage. It can be shown that if a sequence $d^0 \ldots d^n$ of disjuncts is obtained from a legal linkage K, then the linkage can be uniquely reconstructed from the sequence. Thus we have

$$P(S, K) = P(w^0 \ldots w^n d^0 \ldots d^n) = P(w^0 d^0 \ldots w^n d^n) \tag{26}$$

where it is understood that this quantity is 0 if the disjunct sequence does not constitute a legal linkage. Hence by the laws of conditional probability, we have the exact decomposition

$$P(S, K) = P(w^0 d^0) P(w^1 d^1 \mid w^0 d^0) P(w^2 d^2 \mid w^0 d^0 w^1 d^1) \times \ldots \times P(w^n d^n \mid w^0 d^0 \ldots w^{n-1} d^{n-1}) \tag{27}$$

Let us write h^i for the history at position i. Thus h^i consists of the constituents up to but not including $w^i d^i$; explicitly $h^i = w^0 d^0 \ldots w^{i-1} d^{i-1}$. Then we can render this decomposition more compactly as

$$P(S, K) = \prod_{i=0}^{n} P(w^i d^i \mid h^i) \tag{28}$$

A given factor $P(w^i d^i \mid h^i)$ in Equation 28 is the probability that word w^i, playing the grammatical role detailed by d^i, will follow the words and incomplete parse recorded in h^i. Figure 3 is a graphical representation of this notion, for position 7 of our example.

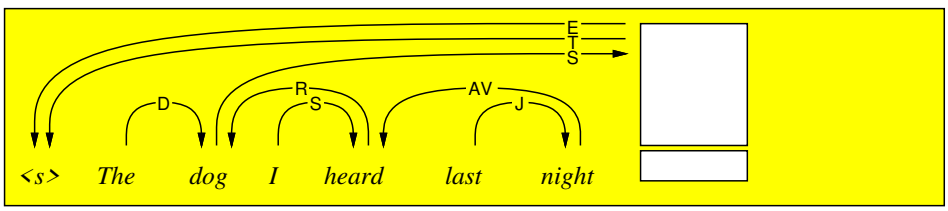

Figure 3: Meaning of $P(w^7 d^7 \mid h^7)$. h^7 is the sequence of all words and disjuncts to the left of position 7. Positions are numbered from the left, starting with 0.

4.3 Equivalence Maps of Histories

The problem is now to estimate the individual probabilities in the right hand side of Equation 28. The difficulty is exactly the one faced in word N-gram modeling—there are too many different histories, and not enough evidence.

We are driven to the solution used by N-gram modelers, which is to divide the space of possible histories into equivalence classes, via some map $\phi : h \mapsto [h]$, and then to estimate the probabilities $P(w\,d \mid [h])$. Approximating each factor $P(w^i d^i \mid h^i)$ by $P(w^i d^i \mid [h^i])$, equation (28) yields

$$P(S,K) = \prod_{i=0}^{n} P(w^i d^i \mid h^i) \approx \prod_{i=0}^{n} P(w^i d^i \mid [h^i]). \tag{29}$$

This expedient has the advantage of coalescing, into each class $[h]$, evidence that had previously been splintered among many different histories. It has the disadvantage that the map $h \mapsto [h]$ may discard key elements of linguistic information. Indeed, the trigram model, which throws away everything but the two preceding words, leads to the approximation $P(\textit{barked} \mid \textit{The dog I heard last night}) \approx P(\textit{barked} \mid \textit{last night})$. This is precisely what drove us to dependency modeling in the first place.

Reviewing the argument of Section 2.2, our hope in incorporating the incomplete parse into each h^i is that the parser will identify just which words in the history are likely to be of use for prediction. To return to our example, we have a strong intuition that *barked* is better predicted by *dog*, five words in the past, than by the preceding bigram *last night*. Indeed, none of the words of the relative clause—to which *barked* bears no links—would seem to be of much predictive value.

This intuition led us to the following design decision: the map $\phi : h \mapsto [h]$ retains (1) a *finite context*, consisting of some fixed number of preceding words (typically 0, 1 or 2), and (2) a *link stack*, consisting of the open (unconnected) links at the current position, and the identities of the words from which they emerge. The action of this map is depicted in Figure 4, where the information retained in $[h^7]$ is rendered in black.

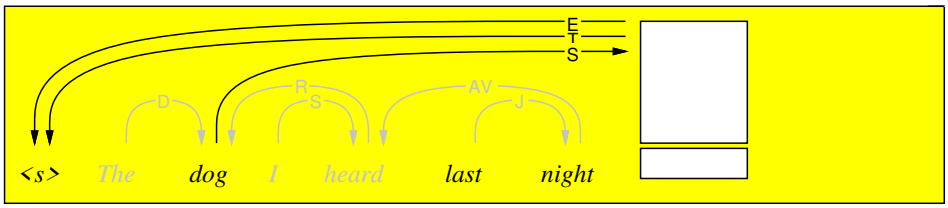

Figure 4: Meaning of $P(w^7 d^7 \mid [h^7])$. $[h^7]$ consists of the elements displayed in black.

We include the finite context in $[h]$ because trigram and bigram models are remarkably effective predictors, despite their linguistic crudeness. We include the link stack because it carries both grammar—it constrains the d that can appear in the next position, since *left*(d) must match some prefix of the stacked links—and semantics—we expect the word in the next position to bear some relation of meaning to any word it links to. Moreover, this choice for ϕ has the advantage of discarding the words and grammatical structure that we believe to be irrelevant (or at least less relevant) to the prediction at hand.

4.4 Maximum Entropy Formulation: Constraints

We chose to formulate our model using the method of constrained maximum entropy [1]. By familiar maneuvers with Lagrange multipliers, we know that the model must be of the form

$$P(w\,d \mid [h]) = \frac{e^{\sum_i \lambda_i g_i(w,d,[h])}}{Z(\bar{\lambda},[h])} \tag{30}$$

Here each $g_i(w,d,[h])$ is a *feature indicator function*, more simply *feature function* or just *feature*, and λ_i is its associated parameter. The constraint consists of the requirement

$$\hat{P}[g_i] = \tilde{P}[g_i]. \tag{31}$$

On the right hand side, $\tilde{P}$ stands for $\tilde{P}(w,d,[h])$, the joint empirical distribution. On the left hand side, $\hat{P}$ is the composite distribution defined by

$$\hat{P}(w,d,[h]) = P(w\,d \mid [h]) \cdot \tilde{P}([h]), \tag{32}$$

where $P(w\,d \mid [h])$ is one of the conditional maximum entropy models we are building, and $\tilde{P}([h])$ is the empirical distribution on history equivalence classes. We will sometimes abuse this nomenclature and refer to the feature function g_i itself as a constraint; it should be understood that the actual constraint is given by Equation 31.

4.4.1 N-Gram Constraints

Assuming that we retain two words of finite context, denoted oldest to youngest as h^{-2} and h^{-1}, we recognize five different kinds of feature. The first three kinds are based upon the familiar notion of N-grams, and are defined as

$$g_z(w,d,[h]) = \begin{cases} 1 & w = z \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

$$g_{yz}(w,d,[h]) = \begin{cases} 1 & w = z \text{ and } h^{-1} = y \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

$$g_{xyz}(w,d,[h]) = \begin{cases} 1 & w = z,\ h^{-1} = y \text{ and } h^{-2} = x \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

Typically, there are many such functions, distinguished from one another by the unigram, bigram or trigram they constrain. For example, $g_{\mathit{barked}}(w,d,[h])$ is 1 when w is *barked*, and 0 otherwise. The set $\{g_z\}$ of all such functions is called the set of *unigram constraints*. There is one such function for each word z whose unigram probability we choose to so constrain; we write $\mathcal{U}$ for the set of all these words.

Likewise, we have the *bigram constraints* $\{g_{yz}\}$, identified by the set $\mathcal{B}$ of constrained bigrams, and the *trigram constraints* g_{xyz}, defined by the set $\mathcal{T}$ of constrained trigrams. If we retain only one word of finite context, we dispense with trigram constraints; if we retain no words of finite context, we dispense with bigram constraints as well. These notions are more fully described in [18, 27].

4.4.2 Other Constraints

We depart from other maximum entropy language models by incorporating two other kinds of constraints: (word, disjunct class) and link bigram. A (word, disjunct class) feature, denoted $g_{z[e]}$, is defined by

$$g_{z[e]}(w,d,[h]) = \begin{cases} 1 & w = z \text{ and } [d] = [e] \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

Here $[e]$ is an equivalence class of disjuncts, defined by some unspecified but linguistically plausible map $\psi : e \mapsto [e]$. The constraints arising from these feature functions are intended to capture the intuition that words tend to be used more as one part of speech than another. We write $\mathcal{D}$ for the set $\{z[e]\}$ of all constrained (word, disjunct class) pairs.

But the really novel element of our model is the link bigram constraint. It is here that we condition the probability of the predicted word w upon linguistically related words in the past, possibly out of trigram range. This is done via a link bigram feature function $g_{yLz}(w,d,[h])$, defined by

$$g_{yLz}(w,d,[h]) = \begin{cases} 1 & w = z \text{ and } [h] \sim d \text{ via } yLz \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

The notation "$[h] \sim d$," read "$[h]$ matches d," means that d is a legal disjunct to occupy the next position in the parse. This in turn is determined by the identity and order of links in *left*(d), and on the link stack of $[h]$. Specifically, if *left*(d) contains r links, then these must exactly match the links of the first r entries of the link stack of $[h]$, both lists given innermost to outermost. Figure 5 depicts matching and non-matching examples. The additional qualification "via yLz" means that at least one of the links must bear label L, and connect to word y. Returning to our example, we have $g_{\mathit{dog\mathrm{S}barked}}(w^7,d^7,[h^7]) = 1$, but $g_{\mathit{IS\,barked}}(w^7,d^7,[h^7]) = 0$. We write $\mathcal{L}$ for the set of all link bigram constraints.

In selecting N-gram constraints, we simply thresholded the counts of the associated N-grams. In selecting link bigram constraints, we either thresholded the counts or used an information-gain criterion. More on the selection of link bigram constraints appears in Section 6.6. For all constraints, we used ratios of counts, or ratios of smoothed counts, to compute empirical expectations.

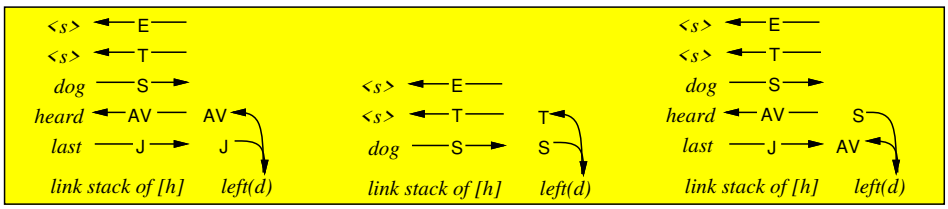

Figure 5: Matching and non-matching $[h], d$ pairs. Left, center: matching. Right: non-matching. Innermost links are at the bottom of the page, outermost at the top.

4.5 Comparison with Triggers

We caution the reader against equating link bigrams with triggers, a notion investigated in [18, 27]. When *dog* operates as a trigger, it elevates the probability of *barked* (and presumably depresses the probability of non-canine verbs like *watusied*) at every subsequent position of the sentence. But when *dog* acts through the link bigram *dog*S*barked*, it will operate only at a position bearing an S link to *dog*. In particular, this constraint will not operate at position 4 of our example, since the incomplete parse encoded in h^4 would not permit such a link. (Indeed, the subject of the relative clause is *I*, and the verb that follows is likely to be one more appropriate for humans.)

4.6 Practical Considerations

Unfortunately, the model just described proved infeasible to build. As described in Section 3.2, our domain-independent maximum entropy solver read uncompressed ASCII files as input, and required us to list the feature functions active for all actual *and potential* futures of any given history. Since we allowed any word to bear any disjunct, the space of futures $\{w\} \times \{d\}$ was very large. Thus, even though the number of observed histories was relatively low—it was bounded by the size of our modest training corpus—the solver's input rapidly exceeded the maximum file size imposed by the operating system.

Another approach was needed. Since the heart of the problem was the number of potential futures $\{w\} \times \{d\}$, we decided to move the disjunct into the model's history. Because the sequence $d^0 \ldots d^n$ is identified with the parse K, this maneuver yields a conditional model $P(S \mid K)$. This follows directly from the observation

$$P(w^0 \ldots w^n \mid d^0 \ldots d^n) = \frac{P(w^0 d^0 \ldots w^n d^n)}{P(d^0 \ldots d^n)} = \frac{P(S, K)}{P(K)} = P(S \mid K). \tag{38}$$

Note that we have the decomposition

$$P(w^0 \ldots w^n \mid d^0 \ldots d^n) = P(w^0 \mid d^0 \ldots d^n) P(w^1 \mid w^0 d^0 \ldots d^n) \times \ldots \times P(w^n \mid w^0 \ldots w^{n-1} d^0 \ldots d^n). \tag{39}$$

If we identify h^i with the sequence $w^0 \ldots w^{i-1} d^0 \ldots d^n$—that is, the preceding i words, and all the disjuncts—then approximating factor-by-factor as in Equation 29 we may write

$$P(S \mid K) = \prod_{i=0}^{n} P(w^i \mid h^i) \approx \prod_{i=0}^{n} P(w^i \mid [h^i]) \tag{40}$$

and proceed more or less as before.

Once again, we must define a map $\phi : h \mapsto [h]$. Since the argument h to ϕ now includes all the disjuncts of the sentence, we have a great deal more information at our disposal, which we may include in $[h]$ if we wish. However, we chose to retain in $[h]$ the same data as before, namely the finite context and the link stack. Indeed, we went beyond our former policy, by retaining only those links that could cause a link bigram to fire, and moreover sometimes discarding the link labels, except for the direction of the arrow. Note that the arguments to each constraint function g_i are now just w and $[h]$.

Discarding all this information forced us to dispense with the (word, disjunct class) constraint functions $g_{s[e]}$, since the only representation in $[h]$ of an individual disjunct is via the links it contributes to the stack.

Our rationale for this seemingly drastic maneuver is that we coalesce into the same class $[h]$ those histories that share a common element of meaning.

By adopting the expedient of building the conditional model of Equation 40, by restricting the finite context to either 0 words or 1 word, and by limiting the total number of constraint functions, we were able to build several small but nontrivial dependency models. Details of these models, and the decoding results we obtained, can be found in Section 6.6.

Of course, we are ultimately still interested in obtaining an estimate of $P(S, K)$. This can be recovered via the identity $P(S, K) = P(S \mid K)P(K)$, but we are then faced with the computation of $P(K)$. As it happens, though, the Collins parser generates an estimate of $P(K \mid S)$, and we use this quantity as an approximation to $P(K)$, yielding

$$P(S, K) \approx P(S \mid K)P(K \mid S). \tag{41}$$

This decomposition is decidedly illegitimate mathematically, and renders meaningless any perplexity computation based upon Equation 41. However, our aim is to reduce the word error rate, and the performance improvement we realize by incorporating $P(K \mid S)$ this way is for us an adequate justification.

In fact, for the decoding experiments, we used the geometrically averaged quantity

$$P(S, K) \approx P(n)^{\alpha} P(T \mid n, S)^{\beta} P(D \mid T, n, S)^{\gamma} P(S \mid D, T, n)^{\delta}. \tag{42}$$

Here n is the sentence length, T is the sequence of part-of-speech tags, D is the dependency parse, S is the sentence text, and α, β, γ and δ are experimentally determined mixing weights. Note that T, D and n taken together constitute a linkage K. Thus the quantity $P(S \mid D, T, n)$ is precisely the probability $P(S \mid K)$ that we have labored to develop here.

5 Feature Selection for the Dependency Model

Up to this point, the motivation for our models relies solely on qualitative linguistic assumptions about long-distance dependencies, as described in Section 2. While these assumptions certainly appear to be sensible, let us remember that the recent history of grammatically motivated language models is littered with surprising disappointments. It is true that our approach is safer than most, in that it concentrates its parameters on lexical dependencies and thus avoids the information loss inherent in phrase-structured probabilistic context-free grammars. Still, the recent experience suggests that it would be wise to ascertain through as many means as possible whether we are barking up the right tree.

With this in mind, we pursued a number of quantitative experiments in parallel with the implementation of our model. These were aimed at obtaining a quantitative feel for the amount of information residing in long-distance dependencies. Such measures equip us better to understand what kinds of potential perplexity reduction one might hope to unlock in the link structures.

5.1 Experimental Methodology

Experiments were conducted over the same parsed training corpus used for the maximum entropy model training. This corpus contains 1,437,972 links. Note that this is exactly the number of words in the corpus, under the pure dependency parses we derived from Collins's parser, in which each word is linked to exactly one other. This is a degenerate case of the link grammar model, where the number of links generally exceeds the number of words.

We proceeded in the phases described below. We first collected measurements on links and link bigrams. We then turned to the link selection problem, to discover what kinds of equivalence classing could be useful.

5.2 Basic Characteristics of Links

The raw link labels we used were somewhat unusual due to the conversion of parse trees produced by Collins's parser into dependency parses. Each link in the dependency parse corresponds to a pair of sibling nodes in the original parse tree. We therefore represented each link label as a 4-tuple (a, ll, lc, lr) constructed by taking the nonterminal labels of the left and right sibling nodes ll and lr along with nonterminal label lc

Distance $\geq$	Link Types
1	2477
2	1798
3	1335

Table 1: Number of distinct link types, as a function of minimum distance.

of their common parent node and an arrow direction a indicating whether ll or lr is the head word. For example, $(\leftarrow, \text{IN}, \text{PP}, \text{NP})$ is the label of a link to a preposition from the head word of a noun phrase that is its object.

An immediate concern of using such link labels is the unusually large space of possible link types $2|N \times N \times N|$ where N is Collins's nonterminal set. We therefore first checked the actual number of distinct link types occurring in our training corpus, as shown in Table 1. The numbers are far larger than one would expect in a "real" dependency grammar, but still only a small portion of the possible space is actually populated. Moreover, many link types occurred only at distance 1, i.e., between adjacent words, and thus are not likely to be the sort of long-distance dependencies we seek.

The frequency of link types is far from uniform; the partial histogram in Table 2 shows an exponential drop. In fact, over half of the 2,477 link types occur six times or less, which as it turned out made sparseness of training data a serious problem.

5.3 Basic Characteristics of Link Bigrams

Table 3 shows the frequency of long-distance link dependencies, as the threshold on the minimum distance considered "long" is raised. We found that the majority of links (55%) occur between adjacent words. This is perhaps to be expected as it is consistent with the generally decent performance of simple bigram models. However, the fact that 22% of the links are "true" long-distance dependencies that fall outside trigram range is encouraging.

Table 4 shows how the number of distinct link bigrams yLz and their frequency of occurrence fall as the minimum threshold for "long" distance is increased. *Link bigrams* are considered distinct if they differ in any of the words y or z, or in any component of the link label $L = (a, ll, lc, lr)$. The same statistics are also shown for several equivalence classings on the link labels L. In *arrow bigrams*, link labels are replaced as mentioned earlier with a simple direction flag. For *word bigrams* we discard even the arrow direction, requiring only that the two words be linked. The link equivalence classes used for *Collins bigrams* are described later.

Note that Zipf's Law generalizes to link bigrams: the average number of occurrences is deceptive, as the median number of occurrences is one—213,627 out of 303,548 link bigrams occurred just once.

Since the dependency language model as we have formulated it already incorporates ordinary Ngram constraints (in the reduced histories), the link bigrams that could potentially contribute additional information are those that span distances greater than N. The particular models within our implementation constraints were restricted to bigram histories, so the rest of our experiments assume the minimum link distance to be 2, i.e., we study non-adjacent link bigrams.

In practice, the number of link bigram constraints is limited by the implementation. Our original plan was to select link bigrams by a mutual information criterion against an ordinary Ngram model. Unfortunately, experiments revealed problems with sparse counts, even against a bigram model. We therefore decided to rely primarily on frequency thresholding to select which distinct link bigram types to apply constraints on. Table 5 shows what kind of reduction can be expected as the frequency threshold varies. (Remember that all data henceforth is for non-adjacent links.)

f	Number of Link Types Occurring f Times
1	548
2	273
3	169
4	124
5	93
6	61
7	62
8	51
9	52
10	30
11	33
12	24
13	26
14	32
15	17
16	16
	... (some representative picks) ...
32	8
64	4
128	4
256	1
	... (mostly zero starting here) ...
516	1
1023	1
2073	1
4250	1
8204	1
16477	1
37846	1
57392	1
65210	1
112507	1
204722	1

Table 2: Partial histogram of link type frequencies.

Distance $\geq$	Frequency	Percent of total tokens
1	1437972	100.0%
2	648011	45.1%
3	322009	22.4%
4	182796	12.7%

Table 3: Frequency of long-distance link dependencies, as the distance grows.

Distance ≥	Link Bigrams Types	Link Bigrams Avg Occur	Arrow Bigrams Types	Arrow Bigrams Avg Occur	Word Bigrams Types	Word Bigrams Avg Occur	Collins Bigrams Types	Collins Bigrams Avg Occur
1	303548	4.74						
2	176661	3.67	160194	4.05	156476	4.14	163725	3.96
3	103247	3.12	94007	3.43	91937	3.50		
4			57912	3.16				

Table 4: Number of distinct link bigram types and their average number of occurrences, as a function of minimum distance.

Min Freq	Link Bigram Types	Arrow Bigram Types	Collins Bigram Types
2	46666	45771	
3	26327	26177	
4		18628	
5		14370	
6		11821	
7		10005	
10		6943	
12		5733	
13		5248	
14		4859	4847
15		4538	

Table 5: Number of distinct link bigram types and their average number of occurrences, as a function of minimum distance.

				Link	All Links		Distance ≥ 2		Distance ≥ 3	
a	ll	lc	lr	Function	Rank	Freq	Rank	Freq	Rank	Freq
$\rightarrow$	NP	S	VP	Subject/Verb	(1)	204722	(4)	27307	(4)	11444
$\leftarrow$	VB	VP	NP	Verb/Object	(8)	27567	(8)	14745	(8)	6753
$\leftarrow$	IN	SBAR	S	Subord. Conjunction/Clause	(11)	22547	(7)	22206	(6)	7608
$\leftarrow$	VB	VP	SBAR	Verb/SBAR-Complement	(38)	7883	(31)	4453	(26)	2673
$\leftarrow$	VB	VP	S	Verb/S-Complement	(39)	7820	(32)	4446	(28)	2443

Table 6: Collins's suggested most important links. The *Function* column gives an indication of the grammatical relation expressed by the link type.

5.4 Link Selection

As just mentioned, we did not have enough training data to proceed with our original plan for constraint selection. Our plan was subsequently revised to

1. *Link bigram selection.* Analyze link bigrams primarily by frequency, whose effect appears from inspection to swamp the mutual information scores.
2. *Link selection.* Apply both manual and mutual-information criteria to select worthwhile link types.

Recall that in our baseline model all link types are equivalence-classed into just two types, preserving only the left/right arrow distinction. The link selection problem is to select additional link types whose specific type identity will not be discarded. We now discuss a number of approaches to link selection that we investigated, beginning with manual selection based on linguistic intuition, and progressing toward increasingly automated quantitative methods.

Collins links. During his visit to the Workshop, Michael Collins suggested that certain links representing predicate-argument relations (as opposed to head-adjunct ones) should not be equivalence-classed away. Table 6 shows the five links he suggested, along with the ranks (by frequency) and frequencies of these links as found in the training corpus with several different minimum distances.

Top 40. Looking at the ranks of Collins's suggested links in Table 6, one observes that they all rank among the top 40 link types, regardless of the minimum distance threshold. This suggests selecting other links out of (some of) the most frequent links. For non-adjacent links (distance ≥ 2), which are most likely to be useful, the top 40 are shown in Table 7.

Pruned Top 40. Some of the top 40 links will clearly not add much information beyond the arrow equivalence class. For example, the identity of the noun modifier link $(\leftarrow, \mathrm{NN}, \mathrm{NN}, \mathrm{NN})$ is unlikely to help once the two participating nouns themselves are already known. A quick manual pruning of the list yields 24 types that would appear to be more sensible, shown in Table 8. This supersets Collins's suggested links.

Average link gain. From an information-theoretic perspective, a more principled approach to link selection should be to use information gain. In the simplest approach, we define the *average link gain* as

$$I(z; L|y[L]) \quad = \quad E_{P(L)} G(L)$$

where the individual *link gain*

$$G(L) \quad = \quad E_{P(yz|L)} \log \frac{P(Lz|y[L])}{P(z|y[L])P(L|y[L])}$$

Frequency	Link			
	a	ll	lc	lr
110513	←	TOPL	TOPC	S
43771	←	IN	PP	NP
37360	→	CC	S	VP
27307	→	NP	S	VP
24091	→	DT	NP	NN
23523	→	INTJ	S	VP
22206	←	IN	SBAR	S
14745	←	VB	VP	NP
13042	←	VBP	VP	NP
12525	←	VBZ	VP	NP
11807	←	VBP	VP	VP
9587	←	VBP	VP	SBAR
9343	←	VBD	VP	NP
8550	←	NP	NP	NP
8182	→	S	S	VP
7892	→	PRN	S	VP
7882	→	RB	S	VP
7847	←	WHADVP	SBAR	S
7780	←	VB	VP	PP
7565	←	WHNP	SBAR	S
7524	→	ADVP	S	VP
6333	←	VP	VP	CC
6058	→	DT	NP	NNS
5944	←	VP	VP	VP
5706	→	SBAR	S	VP
5037	←	VBP	VP	PP
4913	←	VBD	VP	PP
4558	←	VBP	VP	ADVP
4517	←	MD	VP	VP
4472	→	PP	S	VP
4453	←	VB	VP	SBAR
4446	←	VB	VP	S
4204	←	VBZ	VP	ADJP
4107	→	EDITED	S	VP
4064	←	VBG	VP	NP
4049	←	VBN	VP	NP
3835	←	VB	VP	ADVP
3510	←	VBZ	VP	S
3492	←	VBZ	VP	ADVP
3363	→	JJ	NP	NN

Table 7: Top 40 most frequent links. The link labels are derived from the nonterminal categories found in the Treebank annotation system [19].

Frequency	Link			
	a	ll	lc	lr
43771	←	IN	PP	NP
37360	→	CC	S	VP
27307	→	NP	S	VP
24091	→	DT	NP	NN
22206	←	IN	SBAR	S
14745	←	VB	VP	NP
13042	←	VBP	VP	NP
12525	←	VBZ	VP	NP
11807	←	VBP	VP	VP
9587	←	VBP	VP	SBAR
9343	←	VBD	VP	NP
8182	→	S	S	VP
7892	→	PRN	S	VP
7882	→	RB	S	VP
7847	←	WHADVP	SBAR	S
7565	←	WHNP	SBAR	S
6058	→	DT	NP	NNS
5706	→	SBAR	S	VP
4517	←	MD	VP	VP
4453	←	VB	VP	SBAR
4446	←	VB	VP	S
4064	←	VBG	VP	NP
4049	←	VBN	VP	NP
3510	←	VBZ	VP	S

Table 8: 24 selected links obtained by pruning the top 40.

	Link			Average Link Gain		Weighted Link Gain		Average Anchor Gain	
a	ll	lc	lr	Rank	Bits	Rank	Bits	Rank	Bits
$\rightarrow$	NP	S	VP	(2451)	0.182855	(7)	0.0260329	(199)	4.13835
$\leftarrow$	VB	VP	NP	(2289)	1.83543	(3)	0.0351866	(39)	6.55541
$\leftarrow$	IN	SBAR	S	(959)	6.78454	(46)	0.00909541	(389)	2.99731
$\leftarrow$	VB	VP	SBAR	(1010)	6.59419	(23)	0.014747	(83)	5.64445
$\leftarrow$	VB	VP	S	(937)	6.87514	(27)	0.0139335	(90)	5.52263

Table 9: Average link gain, weighted link gain, and average anchor gain, evaluated on Collins's links.

is the average mutual information between the predicted word z and the link type L, given that the predicting word y and the equivalence class $[L]$ are known. In other words, $G(L)$ indicates how much information about the next word z on average is gained by knowing the specific identity L of a link, compared with knowing only the equivalence class $[L]$ of L. Note that this criterion is designed to be essentially insensitive to the frequency of L. The gain is measured relative to a unigram LM. In our models, $[L]$ is just the arrow direction a of L.

One quick evaluation of a link selection criterion is to check how well it selects link types that we have some other reason to believe in. Table 9 shows the ranks assigned to Collins's suggested links by the average link gain criterion. The poor match is caused by the fact that average link gain tends to choose rare links, as the sparse counts cause rare links to be judged the most discriminative.

Weighted link gain. To address the problem with average link gain, we define the *weighted link gain*

$$\tilde{G}(L) \quad = \quad P(L)G(L)$$

which is sensitive to the frequency of L. Results are again found in Table 9. While the match is much better, from inspection of results it appears that this measure is swamped by the frequency, so we might perhaps as well use just the frequency instead of weighted link gain (particularly since the rank of link types, not their gain score, is what really matters).

Average anchor gain. Another gain measure that is potentially more interesting than the link gain measures comes from the *average anchor gain*

$$I(z;y|L) \quad = \quad E_{P(L)}G_A(L)$$

where the individual *anchor gain*

$$G_A(L) \quad = \quad E_{P(yz|L)} \log \frac{P(yz|L)}{P(y|L)P(z|L)}$$

indicates the informativeness of the identity of the anchor y if L is already known. What makes this criterion of greater interest than the preceding ones is that it captures the idea that it is not worth using L as a link type for link bigrams if the bulk of the information comes from L rather than y. Instead, a part-of-speech feature could be used in such cases.

For completeness, Table 9 again shows results against Collins's links. However, we do not believe in this case that evaluation against Collins's links is meaningful, given the very different nature of what average anchor gain is attempting to measure.

5.5 Directions

Many insights have been gained from the information gain studies. Unfortunately, the implementation of the maximum entropy model was completed too late in the workshop schedule to effectively evaluate or improve the various link selection alternatives. It was possible to run only a few preliminary experiments, on an insufficient scale to draw conclusions. Nonetheless, we believe the understanding and experience gained on the potential utility of dependency relations is of sufficient value on its own as to warrant further experiments of this nature.

6 Experiments and Results

We can now describe our experiments with various dependency language models and the results obtained. We also carried out some ancillary experiments that were not related to the dependency language model *per se*, but which were important to calibrate performance and to verify various components of the experimental setup.

6.1 Training and Testing Corpora

All experiments were carried out on the Switchboard corpus of spontaneous conversational English speech [6]. A hand-annotated and hand-parsed version of much of this corpus was created by the Penn Treebank project [19] and made available to us in preliminary form by the LDC. In particular, we relied on the linguistically segmented, the tagged, and the parsed (treebanked) versions of the Switchboard transcripts for various aspects of model training and testing.

6.2 Training Procedure

As explained in Section 2, the dependency model is applied to the single best parse for each sentence, as determined by Collins's statistical parser. The parser itself is fed the single best tag sequence corresponding to the word sequence in question. Therefore, we need to train tagger, parser, and eventually the dependency model in a way that is consistent with the test scenario. This means the parser has to be trained on the actual output of the tagger, including its errors, and similarly the dependency model has to be trained on automatically parsed training data.

The following bootstrap procedure was designed to ensure consistent training while making best use of the available data.

1. The tagger is trained using a portion of the hand-tagged training data (998,000 words).
2. The trained tagger is applied to the hand-parsed portion of the training data, a set disjoint from the one used in the previous step (226,000 words).
3. The parser is trained on the hand-parsed, automatically tagged training data.
4. All linguistically segmented training data (1.44 million words, including the training subsets used above) is automatically tagged and parsed, using the trained tagger and parser, respectively.
5. The parses thus obtained are converted into linkages, and feature-function empirical expectations are computed. In some experiments these empirical expectations are smoothed to give more probability mass to unseen events.
6. The training corpus events, features used, and their target expectations are encoded for the MEMT and the model is trained using the `me.estimate` program.

corpus	tagging	parsing	
	accuracy	precision	recall
Switchboard	95.3%	83.5%	85.3%
Wall Street Journal	96.6%	85.3%	85.7%

Table 10: Tagging and parsing performance on Switchboard and Wall Street Journal corpora.

Tagging and Parsing Accuracy Since our model depends on being provided the best possible linkage, which is derived from the parser (and indirectly from the tagger) output, we evaluated the accuracy of these two tools on the Switchboard corpus. This is especially important as spontaneous speech has so far not been a focus of tagging and parsing research. Both tagger and parser were developed using mainly the Wall Street Journal (WSJ) corpus.

Table 10 gives tagger and parser performance using standard metrics. For the tagger, this is the percentage of tags that were labeled correctly. For the parser, it is customary to evaluate the precision and recall of constituents (bracket pairs). For comparison purposes, we also give the corresponding figures for the Wall Street Journal corpus, as reported in [25] and [2], respectively.

Note, however, that the Switchboard tagging and parsing tasks were actually made more difficult by the way the data was conditioned, in particular by removing all case distinctions and punctuation. Case is used by the tagger, and punctuation benefits both tagger and parser.

As can be seen, the Switchboard performance of both tagger and parser is within 1–2% of the WSJ figures. The amount of training data for tagging was roughly the same in both cases (1 million words), but the Switchboard parser was trained on only 226,000 words, about 1/6 as much data as the WSJ parser. We consider these results encouraging.

6.3 Testing Procedure

The dependency model was evaluated as part of a speech recognition system that was based on the HTK software available at WS96. Since the dependency model does not operate in an incremental, left-to-right fashion, it was not possible to integrate the language model directly into the search. Instead, we employed a *rescoring* approach using N-best lists [22]. For each utterance we generated the 100 best hypotheses using a standard trigram model. These N-best lists were then reordered using language model scores obtained from the dependency model. To this end, each hypothesis was tagged and parsed. Events and features for the dependency structures were extracted and encoded for the MEMT. The `me.evaluate` tool was used to compute the linkage probability for each hypothesis.

The relatively small number of N-best hypotheses (100) was chosen to allow quick turnaround of rescoring experiments. The best word error rate achievable with these N-best lists was 30.0%, the worst possible error rate was 80.8%.

6.4 Segmentation Issues

To have a reasonable chance parsing recognition hypotheses, the language model needs to be evaluated on complete sentences or utterances. Unfortunately, the Switchboard corpus as supplied to us was not segmented in that way, since spontaneous speech has no simple cues for utterance boundaries. The automatic linguistic segmentation of spontaneous speech is the subject of ongoing research [30].

To work around this problem we relied on the hand-segmentation of the word-level transcripts that had been done as part of the treebanking effort [20]. A total of 1.4 million words from the training corpus had been annotated for linguistic segment (utterance) boundaries, from which we drew the training material for our tagger, parser, and the dependency model itself.

For testing, we used a set of 11 conversations that had been similarly hand-segmented (a subset of the standard development test set). The segmented word transcripts were aligned against a time-marked transcript of the corpus, to infer the time marks of utterance boundaries. The waveforms were then divided

Test / Language model	Training Data	
	1.8 M words (all)	1.4 M words (ling. annotated)
ac. seg. waveforms, 2500-best rescoring		
ac. seg. trigram	45.5	46.8
ling. seg. waveforms, 1000-best rescoring		
ac. seg. trigram	45.4	
ling. seg. trigram	44.5[2]	45.1
ling. seg. waveforms & LM, using parser vocabulary		
full backoff trigram		46.2

Table 11: Baseline results (word error rates) using various segmentation for training and test data.

at those locations. These divided waveforms then served as the input to the HTK-based recognizer, which produced the 100-best hypotheses described above, using standard acoustic and language models.

We rescored these linguistically segmented N-best lists with a standard backoff trigram model (trained on the linguistically segmented portion of the training data) to obtain a baseline result for our further experiments, obtaining a word error rate (WER) of 46.2%. Table 11 summarizes further experiments run to determine the effect of the segmentation on WER.

Note that moving from acoustic to linguistic segmentations in the test data (using matched training data) improves WER by 1%, although this improvement currently involves 'cheating' since the segmentations were obtained by hand. Also note that for practical reasons, we chose the vocabulary of our language model to match that of the Treebank. The major difference from the standard recognizer vocabulary is that the treebank treats contractions like *we're* and *that's* as two separate words. The resulting trigram model has a 1% higher WER than the standard one, which we attribute to the fact that the range of the N-gram is effectively diminished by the change in vocabulary.

6.5 Maximum Entropy N-Gram Model Results

As a first test for the MEMT implementation we built and tested a series of maximum entropy N-gram models, i.e., incorporating only N-gram constraints. At this point we ran into the memory and file size demands of the current MEMT, as described in Section 3.2.3. Specifically, the event file size for a model with the same number of parameters as a standard N-gram exceeded 2 gigabytes, the maximum allowed by the operating system.

As a workaround, we decided to limit the number of bigram constraints, including only those bigrams that occurred at least four times in the corpus. (Bigrams contribute the most to the overall number of events, due to their large number and because a single word of context matches a large number of training histories.) The minimum count for unigrams and trigrams was two. We refer to such a model as a "2-4-2 trigram." (For comparison, the standard Switchboard backoff model is a 1-1-2 trigram.) For smoothing in both backoff and ME models, the N-gram counts were discounted using the "absolute discounting" scheme described in [21].

Table 12 summarizes word error rates obtained for various N-gram models, of both the backoff and the maximum entropy varieties. All results were obtained using the same language model weight (12) and insertion penalty (-10).

For the bigram, backoff and ME models perform almost identically. In the case of the trigram, however, we were not able to achieve good performance with the ME approach: the ME model WER is 2.1% higher than a backoff model with the same number of parameters. This is especially surprising as a trigram ME model developed at last year's workshop (LM95) was competitive with the standard trigram [32]. That model used a somewhat different smoothing method, which might account for the difference; in any case,

[2] To extend the linguistic segmentation to the full training corpus, the automatic segmenter of [30] was used.

Language model	WER
2-4 backoff bigram	48.0
2-4 ME bigram	48.3
2-4-2 backoff trigram	46.3
1-1-2 backoff trigram	46.2
2-4-2 ME trigram	48.4
2-4-2 ME trigram (no smoothing)	48.7

Table 12: N-gram results for backoff and ME models.

Model	Link Types	N-gram constraints unigram	bigram	Link constraints	WER DM only	Tag+Parse+DM
bigram		12412	36191		48.3	47.4
unigram+c4	2	12412		37007	48.3	48.1
bigram+c7	2	12412	36191	10005	47.5	46.8
bigram+c2	821	12412	36191	46666	48.4	47.6
bigram+c5+MI	432	12412	36191	12130	48.6	47.5

Table 13: Results for dependency models.

this negative result needs further investigation. As shown in the last row of Table 12, we also tried using strictly empirical expectation targets (without discounting), but without improvement.

We also note that increasing the minimum counts from 1-1-2 to 2-4-2 for the backoff trigram model seems to make virtually no difference to the WER.

6.6 Dependency Model Results

Based on our results with N-gram constraints alone, and because of the size limitations imposed by the implementation, we decided to build all our dependency models with a bigram-only model as the baseline. That is, all dependency models were obtained with a finite context of one single word. Unigram and bigram constraints were augmented by link bigram constraints. We varied both the nature of these link bigram constraints, and our means for selecting them.

The selection of link bigram constraints occurred along three dimensions:

- **Link type equivalences** In one set of models, only the direction of a link was considered distinctive, so effectively there were only two link types (left-to-right and right-to-left). The second, more expressive, class of models used both the direction and the triplet of nonterminals to form the link labels.
- **Count thresholding** Similar to N-gram constraints, only links occurring in the training corpus with a certain minimum frequency were included in the model.
- **Mutual information thresholding** We experimented with a heuristic to select only links that exhibited high mutual information between the link label L and the predicted word z, conditioned on the antecedent word y being known.

Link bigrams were counted as events only if they involved an antecedent word y that was not within the scope of an N-gram constraint. The rationale for this was that the bigram constraints would already model those types of dependencies. This also helped reduce the total number of events.

Table 13 lists the models tried, the number and type of constraints they contained, and the WERs obtained. For each model, we tried two approaches to rescoring. According to Section 4.6, equation 42, the total language model score is a function of the weights for sentence length (α), tag probability (β), parse probability (γ), and words given the parse (δ). In the "DM only" approach, we ignore tag and parse

probabilities (setting $\beta = \gamma = 0$), and set the remaining weights to fixed values ($\alpha = -10$, $\delta = -12$, corresponding to the values used for the WS96 baseline results). In the "Tag+Parse+DM" approach we optimized all weights separately for each model, including tag and parse probabilities.

The first row repeats the baseline for these experiments, the bigram-only model. The **unigram+c4** model contains no word bigram constraints, only unigram and link bigram constraints, and reduces link types to the link direction. Link bigrams were thresholded using a minimum count of four, which results in roughly the same number of link constraints as there are bigram constraints in the bigram model. We see that link bigram constraints can effectively replace the word bigram constraints achieving similar performance.

The **bigram+c7** model uses both word and link bigram constraints, a minimum count of seven for the latter. We see a 0.8% reduction in word error compared to the bigram baseline. The two results are significantly different according to a Sign test ($p < 0.02$).

The **bigram+c2** model uses a minimum link bigram count of two, and distinguishes links by their nonterminal labels, as well as their direction. There were 821 distinct link types of this sort in the training data, generating 4.7 times as many constraints as in the simpler model with only two link types.

Finally, the **bigram+c5+MI** model incorporated a mutual information selection criterion, namely the *average link gain* measure introduced in Section 5.4. Only links whose average gain was at least 1 bit, and which had occurrent at least five times in the training data, were included in the model. Unfortunately, the latter two models did not produce an improvement over the simplest link bigram model, and did in fact worse than the baseline. Clearly, the more sophisticated link selection criteria are not well understood at this point, requiring further experimentation. For example, it could be that link bigram expectations need smoothing similar to the N-gram expectations, at least when the link space is more finely partitioned.

Adding the parser and tagger scores seemed to help uniformly, reducing WER by about 1% percent. On the one hand, this makes sense since a low parser or tagger score should be indicative of an ill-formed hypothesis. On the other hand, these results are biased since the weights were optimized for the test set. Using a larger test set, these experiments need to be redone using independent data to optimize the weights.

6.7 Discussion and Future Work

The results obtained so far are clearly preliminary in that only a few experiments could be carried out, and some obvious experiments are currently not feasible due to resource and implementation constraints. We plan to remove some of these constraints in future work.

The results so far are encouraging, mainly because a simple bigram-based dependency model realized a small (but statistically significant) improvement over a standard bigram. However, the resulting model is not competitive with state-of-the art trigram models, so it remains to be seen if adding dependency constraints to a trigram can give similar or better improvements. Also, smoothing in N-gram based ME models needs to be debugged.

Our preliminary attempts at a more informed selection of link constraints have not been successful. Still, there should be a method that can improve on the count-based thresholding.

We used spontaneous speech as our domain for political reasons, but we note that it would probably be better to first experiment with syntactically based LMs on a linguistically more "well-behaved" corpus, such as the read speech in WSJ. Otherwise, we have to deal with various spontaneous speech effects, such as disfluencies and syntactically incomplete constructions that might affect the dependency model in particular. For example, we might want to try to filter disfluencies prior to parsing. This requires a separate model for disfluencies, possible similar to [7, 31].

Finally, there is a fundamental concern about the applicability of an N-best rescoring approach to a complex linguistically based model like ours. The best achievable error rate in our N-best lists was 30%, making at least every third word incorrect on average. The language model therefore has to choose between hypotheses that are *pretty bad* and those that are *completely wrong*. A model like ours will most likely only be able to construct a small fraction of the internal linguistic structure on the basis of which it evaluates hyptheses, doing about equally poorly on all. We expect the model to be much more useful in discerning *somewhat incorrect* hypotheses from *completely correct* ones (for which the linguistic structure can actually be recovered). To come closer to this goal we need to investigate ways to use dependency-based models earlier in the recognition search, i.e., during lattice search or even during decoding.

7 Summary

We implemented a new family of language models, based on state-of-the-art parsing and syntactic dependencies, for large vocabulary, spontaneous speech recognition. In a preliminary test, these models matched or exceeded the performance of comparable N-gram models, as measured by word error rate. Our simplest dependency model gave a modest absolute reduction in word error rate of 0.8%, compared with a bigram model.

Resource and implementation limitations prevented us from building models that might outperform a conventional trigram language model. A truly searching test of our ideas remains to be done. Likewise, we were unable to conduct a thorough exploration of constraint selection methods.

We designed, implemented and tested the prototype of a domain-independent toolkit for maximum entropy modeling (MEMT). It proved to be fast, but its current implementation requires unreasonable amounts of file space and physical memory, which prevented us from exploring trigram-based dependency models. Nevertheless, we expect this tool to be valuable for further research in statistical modeling in the LVCSR community. Moreover, we anticipate that future versions will have greatly reduced disk and memory requirements.

Finally, we have started to analyze the statistics of grammatical dependencies and their potential utility for language modeling, although this work is far from finished and could not be completed in time to inform the language model experiments carried out during the workshop.

8 Acknowledgments and Credits

The work described here was a collaborative effort of all members of the group, as well as many others at WS96. We thank everyone for their support, especially CLSP for creating the best possible work environment. Special thanks go to Adwait Rathnaparki for making his tagger available, and to Michael Collins for a close collaboration concerning the use of his parser.

This report was written and edited by an assortment of group members. Eric Ristad wrote Section 3, with portions excerpted from [26]. Harry Printz wrote Section 4 and portions of Section 2, and helped with editing. Dekai Wu wrote Section 5. Andreas Stolcke wrote the remaining sections and was responsible for overall editing. Fred Jelinek and Ciprian Chelba commented on drafts and made numerous corrections and valuable suggestions.

References

[1] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] Michael John Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz, CA, June 1996.

[3] I. Cziszar. Why least squares and maximum entropy? An axiomatic approach to inference. *Annals of Statistics*, 19(4):2032–2067, 1991.

[4] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[5] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ureš. Inference and Estimation of a Long-Range Trigram Model. Technical Report CMU-CS-94-188, School of Computer Science, Carnegie Mellon University, Pittsburgh, September 1994.

[6] J. J. Godfrey, E. C. Holliman, and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, volume I, pages 517–520, San Francisco, March 1992.

[7] Peter A. Heeman, Kyung ho Loken-Kim, and James F. Allen. Combining the detection and correction of speech repairs. In *Proceedings of the International Conference on Spoken Language Processing*, Philadelphia, October 1996.

[8] E. T. Jaynes. *Probability Theory in Science and Engineering*, volume 4 of *Colloquium Lectures in Pure and Applied Science*. Socony Mobil Oil Company, Dallas, TX, 1958.

[9] E. T. Jaynes. Where do we stand on maximum entropy? In R. D. Levine and M. Tribus, editors, *The Maximum Entropy Formalism*. MIT Press, Cambridge, MA, 1978.

[10] Frederick Jelinek. Up from trigrams! The struggle for improved language models. In *Proceedings 2nd European Conference on Speech Communication and Technology*, pages 1037–1040, Genova, Italy, September 1991.

[11] Frederick Jelinek, John D. Lafferty, and Robert L. Mercer. Basic methods of probabilistic context free grammars. In Pietro Laface and Renato De Mori, editors, *Speech Recognition and Understanding. Recent Advances, Trends, and Applications*, volume F75 of *NATO ASI Series*, pages 345–360. Springer Verlag, Berlin, 1992. Proceedings of the NATO Advanced Study Institute, Cetraro, Italy, July 1990.

[12] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, 1980.

[13] Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchman, and Nelson Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 189–192, Detroit, May 1995.

[14] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, March 1987.

[15] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[16] John Lafferty, Daniel Sleator, and Davy Temperley. Grammatical trigrams: A probabilistic model of link grammar. In *Proceedings of the 1992 AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, Cambridge, MA, 1992.

[17] K. Lari and S. J. Young. Applications of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 5:237–257, 1991.

[18] Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models: a maximum entropy approach. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, volume II, pages 45–48, Minneapolis, April 1993.

[19] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[20] Marie Meteer et al. Dysfluency annotation stylebook for the Switchboard corpus. Distributed by LDC, `ftp://ftp.cis.upenn.edu/pub/treebank/swbd/doc/DFL-book.ps.gz`, February 1995. Revised June 1995 by Ann Taylor.

[21] Herman Ney and Ute Essen. On smoothing techniques for bigram-based natural language modelling. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, pages 825–828, Toronto, May 1991.

[22] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. R. Rohlicek. Integration of diverse recognition methodologies through reevaluation of n-best sentence hypotheses. In *Proceedings DARPA Speech and Natural Language Processing Workshop*, pages 83–87, Pacific Grove, CA, February 1991. Defense Advanced Research Projects Agency, Information Science and Technology Office.

[23] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. Technical Report CMU-CS-95-144, School of Computer Science, Carnegie Mellon University, Pittsburgh, May 1995.

[24] A. Ratnaparkhi, S. Roukos, and R. T. Ward. A maximum entropy model for parsing. In *Proceedings of the International Conference on Spoken Language Processing*, pages 803–806, Yokohama, September 1994.

[25] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, University of Pennsylvania, Philadelphia, May 1996.

[26] E. S. Ristad. Maximum entropy modeling toolkit. Technical report, Department of Computer Science, Princeton University, Princeton, NJ, January 1997. 1.5 Beta.

[27] Ronald Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, April 1994. Technical Report CMU-CS-94-138.

[28] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228, 1996.

[29] Daniel D. K. Sleator and Davy Temperley. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, School of Computer Science, Pittsburgh, 1991.

[30] Andreas Stolcke and Elizabeth Shriberg. Automatic linguistic segmentation of conversational speech. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1005–1008, Philadelphia, October 1996.

[31] Andreas Stolcke and Elizabeth Shriberg. Statistical language modeling for speech disfluencies. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, volume I, pages 405–408, Atlanta, May 1996.

[32] M. Weintraub, Y. Aksu, S. Dharanipragada, S. Khudanpur, H. Ney, J. Prange, A. Stolcke, F. Jelinek, and E. Shriberg. Lm95 project report: Fast training and portability. Research Notes No. 1, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, February 1996.

[33] J. H. Wright. LR parsing of probabilistic grammars with input uncertainty for speech recognition. *Computer Speech and Language*, 4:297–323, 1990.