Grammatical Trigrams: A Probabilistic Model of Link Grammar*

John Lafferty† Daniel Sleator‡ Davy Temperley§
jlaff@watson.ibm.com sleator@cs.cmu.edu dt3@cunixa.cc.columbia.edu

Abstract

In this paper we present a new class of language models. This class derives from link grammar, a context-free formalism for the description of natural language. We describe an algorithm for determining maximum-likelihood estimates of the parameters of these models. The language models which we present differ from previous models based on stochastic context-free grammars in that they are highly lexical. In particular, they include the familiar n-gram models as a natural subclass. The motivation for considering this class is to estimate the contribution which grammar can make to reducing the relative entropy of natural language.

Introduction

Finite-state methods occupy a special position in the realm of probabilistic models of natural language. In particular, the simplicity, and simple-mindedness, of the trigram model renders it especially well-suited to parameter estimation over hundreds of millions of words of data, resulting in models whose predictive powers have yet to be seriously contested. It has only been through variations on the finite-state theme, as realized in cached models, for example, that significant improvements have been made. This state of affairs belies our linguistic intuition, as it beguiles our scientific sensibilities.

In the most common probabilistic model of context-free phrase structure grammar [8], the *parameters* are the probabilities $P_A(A \to B\ C)$ and $P_A(A \to w)$, where A, B and C are nonterminals, and w is a terminal symbol. For natural language, experience has shown that this model only weakly captures contextual dependencies, even if the set of nonterminals is sufficiently rich to encode lexical information, a goal toward which many unification-based grammars strive [4]. More to the point, the cross-entropies of language models constructed from probabilistic grammars have so far been well above the cross-entropies of trigram language models [3, 6, 14].

Link grammar is a new context-free formalism for natural language proposed in [13]. What distinguishes this formalism from many other context-free models is the absence of explicit constituents, as well as a high degree of lexicalization. It is this latter property which makes link grammar attractive from the point-of-view of probabilistic modeling.

Of course, several grammatical formalisms besides link grammar have been proposed which are highly lexical. One such example is lexicalized tree adjoining grammar [12], which is in fact weakly context sensitive in generative power. While this formalism is promising for statistical language modeling, the relative inefficiency of the training algorithms limits the scope of the associated models. In contrast, the motivation behind constructing a probabilistic model for link grammar lies in the fact that it is a very simple formalism, for which there exists an efficient parsing algorithm. This suggests that the parameters of a highly lexical model for link grammar might be estimated on very large amounts of text, giving the words themselves the ability to fully exercise their statistical rights as well as their grammatical proclivities. In this way one can hope to contest the unreasonable dominion that the insipid trigram holds over probabilistic models of natural language.

---

* Presented to the 1992 AAAI Fall Symposium on Probabilistic Approaches to Natural Language.

† IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598

‡ School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

§ Music Department, Columbia University, New York NY 10027

Link grammar

The best way to explain the basics of link grammar is to discuss an example of a *linkage*. Figure 1 shows how a linkage is formed when the words, thought of as vertices, are connected by labelled arcs so that the resulting graph is connected and planar, with all arcs written above the words, and not more than one arc connecting any two words. The labelled arcs are referred to as *links*.

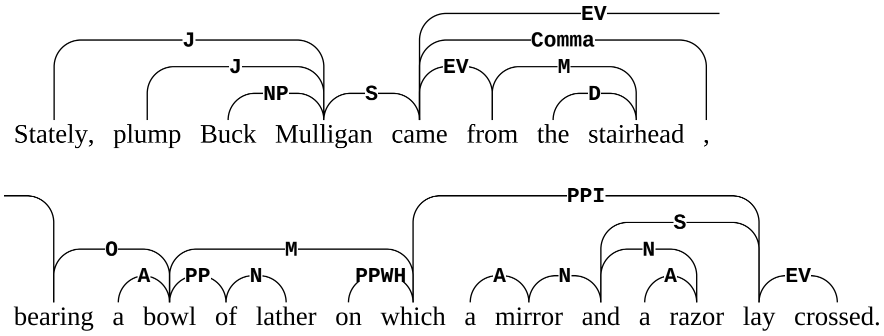

FIGURE 1

A *usage* of a word w is determined by the manner in which the word is linked to the right and to the left in a sentence. In Figure 1, for example, the word "came" is seen to be preceded by a subject, and followed by two adverbial phrases, separated by a comma. This usage of "came" is characterized by an `S` *connector* on the left, and two right `EV` connectors, separated by a `Comma` connector. We can thus say that one usage of the word "came" is $((\texttt{S}), (\texttt{EV}, \texttt{Comma}, \texttt{EV}))$. Similarly, a usage of the word "and" is $((\texttt{N}), (\texttt{S}, \texttt{N}))$; that is, it may coordinate two noun phrases as the subject of a verb. Of course, the labels in the above examples are quite simple; to incorporate more structure, it would be natural for the connectors to be represented by feature structures, and for linking to make use of unification.

A *dictionary* specifies all possible usages of the words in the vocabulary. A usage will also be referred to as a *disjunct*, and is represented by a pair of ordered lists

$$d = ((l_m, l_{m-1}, \ldots, l_1), (r_1, r_2, \ldots, n)).$$

The l_i's are left connectors and the r_i's are right connectors. Links are formed for a word W with disjunct d by connecting each of the left connectors l_i of d to a right connector $\widetilde{r}_i$ for some word L_i to the left of W, and and by similarly connecting each right connector r_j of d to the left connector $\widetilde{l}_j$ of some word R_j to the right of W. In Figure 1, for example, the left `N` connector in the disjunct $((\texttt{N}), (\texttt{S}, \texttt{N}))$ for the word "and" is connected to the right `N` connector in the $((\texttt{A}), (\texttt{N}))$ disjunct for "mirror." The lists of left and right-connectors are ordered, implying that the words to which $l_1, l_2, \ldots$ are connected are decreasing in distance to the left of W, and the words to which r_i are connected are decreasing in distance to the right of W. We will make use of the notation which for a disjunct $d = ((l_m, l_{m-1}, \ldots, l_1), (r_1, r_2, \ldots, r_n))$ identifies $\triangleleft l_i = l_{i+1}$ in case $i < m$, and $\triangleleft l_m = \textsc{nil}$. Similarly, we set $r_j \triangleright = r_{j+1}$ for $j < n$ and $r_n \triangleright = \textsc{nil}$. The first left connector of d is denoted by $\mathit{left}[d] = l_1$, and the first right connector is $\mathit{right}[d] = r_1$. Of course, it may be that $\mathit{left}[d] = \textsc{nil}$ or $\mathit{right}[d] = \textsc{nil}$. In short, a disjunct can be viewed as consisting of two linked lists of connectors.

A parse or linkage of a sentence is determined by selecting a disjunct for each word, and choosing a collection of links among the connectors of these disjuncts so that: the graph with words as vertices and links as edges is connected, the links (when drawn above the words) do not cross, each connector of each chosen disjunct is the end point of exactly one link, and the connectors at opposite ends of each link match. If no such linkage exists for a sequence of words, then that sequence is not in the language defined by the link grammar.

We refer the reader to [13] for more information about link grammars. That report describes a terse notation for use in writing link grammars, the workings of a wide-coverage link grammar for English, and efficient algorithms and heuristics for parsing sentences in a link grammar.

Link grammars resemble two other context-free grammatical formalisms: *categorial grammars* [11] and *dependency grammars* [7, 10]. Both link grammar and categorial grammar are highly lexical. The cancellation operator in a categorial grammar derivation is similar to linking process in a link grammar. In fact, it is possible to take a categorial grammar and generate an equivalent link grammar. (The reverse seems to be much more difficult.) Dependency grammars, like link grammars, involve drawing links between the words of a sentence. However, they are not lexical, and (as far as we know) lack a parsing algorithm of efficiency comparable to that of link grammars. Our approach to probabilistic modeling of grammar depends on the existence of an efficient parsing algorithm, and on having enough flexibility to represent the bigram and trigram models within the same framework.

The Recognition Algorithm

An algorithm for parsing with link grammar is presented in [13]. The algorithm proceeds by constructing links in a top-down fashion. The recursive step is to count all linkages between a left word L and a right word R which make use of the (right) connector l for L and the (left) connector r for R, assuming that l and r are connected via links that have already been made. The algorithm proceeds by checking for each disjunct d of each word $L < W < R$, whether a connection can be made between d and l or r. There are three possibilities. It may be the case that $\mathit{left}[d]$ links to l and $\mathit{right}[d]$ is either NIL or remains unconnected. Or, $\mathit{right}[d]$ may link to r and $\mathit{left}[d]$ is NIL or unconnected. Alternatively, it may be that $\mathit{left}[d]$ is connected to l *and* $\mathit{right}[d]$ is connected to r.

As a matter of notation, we'll refer to the words in a sentence $S = W_0 W_2 \cdots W_{N-1}$ by using the indices $0, 1, \ldots N-1$. Also, we'll introduce the boundary word W_N for convenience, assigning to it the single disjunct $((\text{NIL}), (\text{NIL}))$. Each word $0 \leq W \leq N$ has an associated set $\mathcal{D}(W)$ of possible disjuncts. Let $c(L, R, l, r)$ be the number of ways of constructing a *sublinkage* between L and R using l and r, as described in [13]. Then $c(L, L+1, l, r)$ is equal to one in case $l = r = \text{NIL}$ and is equal to zero otherwise.

The following is a recursive expression for $c(L, R, l, r)$, on which the dynamic programming algorithm of [13] is based:

$$
\begin{aligned}
&c(L,R,l,r) = \\
&\sum_{L<W<R} \sum_{d \in \mathcal{D}(W)} \Big[\quad \mathit{match}(l, \mathit{left}[d])\, c(L, W, l\triangleright, \triangleleft \mathit{left}[d])\, c(W, R, \mathit{right}[d], r) \\
&\qquad + \quad \mathit{match}(l, \mathit{left}[d])\, \mathit{match}(\mathit{right}[d], r)\, c(L, W, l\triangleright, \triangleleft \mathit{left}[d])\, c(W, R, \mathit{right}[d]\triangleright, \triangleleft r) \\
&\qquad + \quad \delta_{\text{NIL}}(l)\, \mathit{match}(\mathit{right}[d], r)\, c(L, W, l, \mathit{left}[d])\, c(W, R, \mathit{right}[d]\triangleright, \triangleleft r) \Big]
\end{aligned}
$$

Here δ is the standard delta function and *match* is an indicator function, taking values 0 and 1, which determines whether two connectors may be joined to form a link. The function *match* must only satisfy $\mathit{match}(c, \text{NIL}) = \mathit{match}(\text{NIL}, c) = 0$ for any connector c, but is otherwise completely general, and could, for example, take into account unification of feature structures. The term $\delta_{\text{NIL}}(l)$ is included to prevent overcounting of linkages. Since there are at most $\binom{N}{3}$ triples (L, W, R) to be tried, the complexity of the parsing algorithm is $O(D^3 \cdot N^3)$, where D is an upper bound of the number of disjuncts of an arbitrary word in the grammar. The total number of linkages, or parses, of the sentence $S = W_0 \cdots W_{N-1}$ is $\sum_{d \in \mathcal{D}(0)} \delta_{\text{NIL}}(\mathit{left}[d])\, c(0, N, \mathit{right}[d], \text{NIL})$.

The Probabilistic Model

It is natural to develop a generative probabilistic model of link grammar. In using term *generative* we imply that the model will assign total probability mass one to the language of the grammar. The usual probabilistic model of context-free phrase structure grammar, given by the parameters $P_A(A \rightarrow B\ C)$ and $P_A(A \rightarrow w)$, also has this property.

Just as the basic operation of context-free phrase structure grammar is *rewriting*, the basic operation of link grammar is *linking*. A link depends on two connectors, a left connector l and a right connector r. These are the analogues of a nonterminal A which is to be rewritten for a phrase structure grammar. Given l and r, a link is formed by first choosing a word W to link to, followed by a choice of disjunct d for the word. Finally, an *orientation* is chosen for the link by deciding whether d links to l, to r, or to both l and r. In fact, we may also take into account the identities of the words L and R to which the connectors l and r are associated. This suggests the set of parameters

$$\Pr(\, W, d, O \mid L, R, l, r\,)$$

for a probabilistic model. Here O is a random variable representing the orientation of the link, which we will allow to have values $\leftarrow$, $\rightarrow$, or $\leftrightarrow$, in case d is linked to l, to r, or to both l and r. Of course, this probability may be decomposed as

$$\Pr(\, W, d, O \mid L, R, l, r\,) = \Pr(\, W \mid L, R, l, r\,) \Pr(\, d \mid W, L, R, l, r\,) \Pr(\, O \mid d, W, L, R, l, r\,)\ .$$

Since we are forming conditional probabilities on a set of events which is potentially quite large for a reasonable grammar and vocabulary for natural language, it may be impossible in practice to form reliable estimates for them. We thus approximate these probabilities as

$$\Pr(\, W, d, O \mid L, R, l, r\,) \approx \Pr(\, W \mid L, R, l, r\,) \Pr(\, d \mid W, l, r\,) \Pr(\, O \mid d, l, r\,)\ .$$

In addition, we require the joint probability $\Pr(W_0, d_0)$ of an initial word and disjunct.

The probability of a linkage is the product of all its link probabilities. That is, we can express a linkage $\mathcal{L}$ as a set of links $\mathcal{L} = \{(W, d, O, L, R, l, r)\}$ together with an initial disjunct d_0, and we assign to $\mathcal{L}$ probability

$$\Pr(\, S, \mathcal{L}\,) = \Pr(W_0, d_0) \prod \Pr(\, W, d, O \mid L, R, l, r\,)$$

where the product is taken over all links in $\mathcal{L}$, and where we have noted the dependence on the sentence S being generated. This probability is thus to be thought of as the probability of generating S with the linkage $\mathcal{L}$. The cross-entropy of a corpus $S_1, S_2, \ldots.$ with respect to the uniform distribution on individual sentences is then given by

$$H = -\gamma^{-1} \sum_i \log \sum_{\mathcal{L}} \Pr(\, S_i, \mathcal{L}\,)$$

for some normalizing term γ. In the following, we will describe an algorithm to determine a set of parameters which locally minimize this entropy.

Finite-state approximations

Link grammars may be constructed in such a way that the corresponding probabilistic model is a finite-state Markov chain corresponding to the n-gram model. For example, the link grammar whose corresponding probabilistic model is equivalent to the bigram model is depicted in Figure 2.

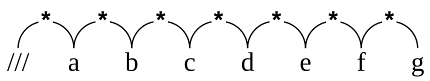

FIGURE 2: A BIGRAM MODEL

Suppose, as another example, that the grammar is made up of a dictionary where a word w has the set of disjuncts

$$\begin{gathered} ((*)\,,\,(wx)) \\ ((xw)\,,\,(wy)) \\ ((xw)\,,\,(\textsc{nil})) \end{gathered}$$

where x and y represent arbitrary words in the vocabulary. The disjunct $((xy),(yw))$ represents the assumption that any two words x and y may precede a word w. This information is passed through the left connector. The identity of the previous word y and the current word w is then passed through the right connector. The disjunct $((*),(w))$ represents the modeling assumption that any word can begin a sentence. Finally, the disjunct $((xy),(\textsc{nil}))$ allows any word to be the last word in a sentence. An artificial word ///, called "the wall," is introduced to represent the sentence boundary [13], and is given the single disjunct $((\textsc{nil}),(*))$. Given this set of disjuncts, each sentence has a unique linkage, which is represented in Figure 3. The resulting probabilistic model is precisely the familiar trigram model.

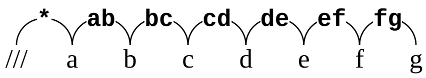

FIGURE 3: A TRIGRAM MODEL

Of course, since the generative power of link grammar is context-free, any finite state model can be represented. The point to be made with the above example, however, is that because of the lexical nature of the probabilistic model that is being proposed, finite-state language models such as the n-gram model and its derivatives can be easily and naturally represented in a probabilistic model of link grammar. Probabilistic link grammar thus provides a uniform framework for finite-state as well as linguistically motivated models of natural language.

In order to capture the trigram model in a traditional probabilistic context-free grammar, the following grammar could be used, where A_{xy} is a nonterminal parameterized by the "previous" words x and y.

$$\begin{array}{rcl} S & \rightarrow & w\; A_{wx} \\ A_{xw} & \rightarrow & w\; A_{wy} \\ A_{xw} & \rightarrow & w \end{array}$$

However, it would certainly be awkward, at best, to incorporate the above productions into a natural language grammar. The essence of the problem, of course, is that the Griebach normal form of a natural language grammar rarely provides a strong equivalence, but rather distorts the trees in a linguistically senseless fashion.

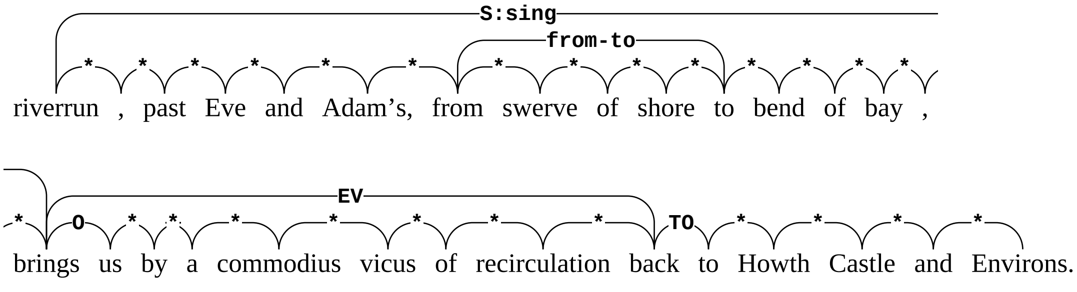

FIGURE 4: A BIGRAM/GRAMMAR MODEL

On the other hand, the corresponding finite-state links could be easily included into a link grammar for natural language in a manner which preserves the relevant structure. While the formalisms are equivalent from the point-of-view of generative power, the absence of explicit constituents as well as the head-driven nature of link grammar lends it well to probabilistic modeling. As an example, in the linkage displayed in Figure 3, subject-verb agreement, object-verb attachment, and adverbial clause attachment are handled using grammar, while the remaining words within each clause phrase are related by the bigram model. In addition, the logical relation between the words "from" and "to" is represented in a link. In this manner long-distance dependencies can be seamlessly incorporated into a bigram or trigram model.

The Training Algorithm

We have developed and implemented an algorithm for determining maximum-likelihood estimates of the parameters of probabilistic link grammar. The algorithm is in the spirit of the Inside-Outside algorithm [8], which, in turn, is a special case of the EM algorithm [2]. The algorithm computes two types of probabilities, which we refer to as *inside* probabilities $Pr_{\mathcal{I}}$ and *outside* probabilities $Pr_{\mathcal{O}}$. Intuitively, the inside probability $Pr_{\mathcal{I}}(L, R, l, r)$ is the probability that the words between L and R can be linked together so that the linking requirements of connectors l and r are satisfied. The term $Pr_{\mathcal{O}}(L, R, L, r)$ is the probability that the words outside of the words L and R are linked together so that the linking requirements outside of the connectors l and r are satisfied. Given these probabilities, the probability that the sentence $W_0, \ldots, W_{N-1}$ is generated by the grammar is equal to

$$\Pr(S) = \sum_{d_0 \in \mathcal{D}(W_0)} \Pr(W_0, d_0) \Pr_{\mathcal{I}}(0, N, \mathit{right}[d], \text{NIL}) .$$

The inside probabilities are computed recursively through the relations

$$\begin{aligned}
&\Pr_{\mathcal{I}}(L, R, l, r) = \\
&\sum_{L<W<R} \sum_{d \in \mathcal{D}(W)} \Big[\quad \Pr(\, W, d, \leftarrow \mid L, R, l, r\,) \Pr_{\mathcal{I}}(L, W, l\triangleright, \triangleleft \mathit{left}[d]) \Pr_{\mathcal{I}}(W, R, \mathit{right}[d], r) \\
&\qquad\qquad + \quad \Pr(\, W, d, \leftrightarrow \mid L, R, l, r\,) \Pr_{\mathcal{I}}(L, W, l\triangleright, \triangleleft \mathit{left}[d]) \Pr_{\mathcal{I}}(W, R, \mathit{right}[d]\triangleright, \triangleleft r) \\
&\qquad\qquad + \quad \Pr(\, W, d, \rightarrow \mid L, R, l, r\,) \Pr_{\mathcal{I}}(L, W, l, \mathit{left}[d]) \Pr_{\mathcal{I}}(W, R, \mathit{right}[d]\triangleright, \triangleleft r) \Big] .
\end{aligned}$$

The outside probabilities are computed by first setting

$$\Pr_{\mathcal{O}}(0, N, \mathit{right}[d], \text{NIL}) = \Pr(W_0, d)$$

for each disjunct $d \in \mathcal{D}(W_0)$ with $\mathit{left}[d] = \text{NIL}$. The remaining outside probabilities are then obtained as a sum of four terms,

$$\Pr_{\mathcal{O}}(L, R, l, r) = \Pr_{\mathcal{O}}^{\triangleleft \mathit{left}}(L, R, l, r) + \Pr_{\mathcal{O}}^{\mathit{left}}(L, R, l, r) + \Pr_{\mathcal{O}}^{\mathit{right}\triangleright}(L, R, l, r) + \Pr_{\mathcal{O}}^{\mathit{right}}(L, R, l, r)$$

where these probabilities are computed recursively through the following relations:

$$\begin{aligned}
&\Pr_{\mathcal{O}}^{\triangleleft \mathit{left}}(L, W, l\triangleright, \triangleleft \mathit{left}[d]) = \sum_{R>W} \sum_{r} \Pr_{\mathcal{O}}(L, R, l, r) \times \\
&\quad [\Pr(\, W, d, \leftarrow \mid L, R, l, r\,) \Pr_{\mathcal{I}}(W, R, \mathit{right}[d], r) + \Pr(\, W, d, \leftrightarrow \mid L, R, l, r\,) \Pr_{\mathcal{I}}(W, R, \mathit{right}[d]\triangleright, \triangleleft r)]
\end{aligned}$$

$$\begin{aligned}
&\Pr_{\mathcal{O}}^{\mathit{right}\triangleright}(W, R, \mathit{right}[d]\triangleright, \triangleleft r) = \sum_{L<W} \sum_{l} \Pr_{\mathcal{O}}(L, R, l, r) \times \\
&\quad [\Pr(\, W, d, \rightarrow \mid L, R, l, r\,) \Pr_{\mathcal{I}}(L, W, l, \mathit{left}[d]) + \Pr(\, W, d, \leftrightarrow \mid L, R, l, r\,) \Pr_{\mathcal{I}}(L, W, l\triangleright, \triangleleft \mathit{left}[d])]
\end{aligned}$$

$$\Pr_{\mathcal{O}}^{left}(L,W,l,\mathit{left}[d]) = \sum_{R>W}\sum_{r}\Pr_{\mathcal{O}}(L,R,l,r)\Pr(\ W,d,\rightarrow \mid\ L,R,l,r\ )\Pr_{\mathcal{I}}(W,R,\mathit{right}[d]\triangleright,\triangleleft r)$$

$$\Pr_{\mathcal{O}}^{right}(W,R,\mathit{right}[d],r) = \sum_{L<W}\sum_{l}\Pr_{\mathcal{O}}(L,R,l,r)\Pr(\ W,d,\leftarrow \mid\ L,R,l,r\ )\Pr_{\mathcal{I}}(L,W,l\triangleright,\triangleleft\mathit{left}[d])\,.$$

The expected number of times that, for example, a word W is linked to words L and R through connectors l and r in a given sentence S is then determined by

$$\begin{aligned}
Count(W,L,R,l,r) \quad = \quad & \Pr_{\mathcal{O}}(L,R,l,r)\Pr(S)^{-1}\sum_{d\in\mathcal{D}(W)}\Pr(\ W \mid\ L,R,l,r\ )\ \Pr(\ d \mid\ W,l,r\ )\Big\{ \\
& \Pr(\ \leftarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l\triangleright,\triangleleft\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d],r) + \\
& \Pr(\ \rightarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l,\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d]\triangleright,\triangleleft r) + \\
& \Pr(\ \leftrightarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l\triangleright,\triangleleft\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d]\triangleright,\triangleleft r)\Big\}\,.
\end{aligned}$$

The counts for the parameters $\Pr(\ d \mid\ W,l,r\ )$ and $\Pr(\ O \mid\ d,l,r\ )$ are obtained in a similar way. For completeness, we list the expected counts below.

$$\begin{aligned}
Count(d,W,l,r) \quad = \quad & Pr(S)^{-1}\sum_{L,R}\Pr_{\mathcal{O}}(L,R,l,r)\Pr(\ W \mid\ L,R,l,r\ )\ \Pr(\ d \mid\ W,l,r\ )\Big\{ \\
& \Pr(\ \leftarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l\triangleright,\triangleleft\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d],r) + \\
& \Pr(\ \rightarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l,\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d]\triangleright,\triangleleft r) + \\
& \Pr(\ \leftrightarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l\triangleright,\triangleleft\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d]\triangleright,\triangleleft r)\Big\}\,.
\end{aligned}$$

$$\begin{aligned}
Count(\leftarrow,d,l,r) \quad = \quad & Pr(S)^{-1}\sum_{L,W,R}\Pr_{\mathcal{O}}(L,R,l,r)\Pr(\ W \mid\ L,R,l,r\ )\ \Pr(\ d \mid\ W,l,r\ )\times \\
& \Pr(\ \leftarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l\triangleright,\triangleleft\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d],r)
\end{aligned}$$

$$\begin{aligned}
Count(\rightarrow,d,l,r) \quad = \quad & Pr(S)^{-1}\sum_{L,W,R}\Pr_{\mathcal{O}}(L,R,l,r)\Pr(\ W \mid\ L,R,l,r\ )\ \Pr(\ d \mid\ W,l,r\ )\times \\
& \Pr(\ \rightarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l,\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d]\triangleright,\triangleleft r)
\end{aligned}$$

$$\begin{aligned}
Count(\leftrightarrow,d,l,r) \quad = \quad & Pr(S)^{-1}\sum_{L,W,R}\Pr_{\mathcal{O}}(L,R,l,r)\Pr(\ W \mid\ L,R,l,r\ )\ \Pr(\ d \mid\ W,l,r\ )\times \\
& \Pr(\ \leftrightarrow \mid\ d,l,r\ )\Pr_{\mathcal{I}}(L,W,l\triangleright,\triangleleft\mathit{left}[d])\Pr_{\mathcal{I}}(W,R,\mathit{right}[d]\triangleright,\triangleleft r)
\end{aligned}$$

The algorithm for obtaining these counts is derived from the dynamic programming algorithm given in [13]. The algorithm involves three passes through the sentence. The first pass computes the inside probabilities in much the same way that the basic recognition algorithm computes the number of linkages. A second pass computes the outside probabilities. Finally, a third pass updates the counts for the parameters of the model in a manner suggested by the above equations.

While the algorithm that we have outlined is in the spirit of the inside-outside algorithm, the actual computations in the two algorithms are quite different. First, the inside pass proceeds in a top-down manner for link grammar, while the usual inside-outside algorithm is based upon the bottom-up CKY chart parsing algorithm. On the other hand, while the outside pass for link grammar is top-down, it differs from the outside pass for the inside-outside algorithm in that the computation is structured exactly like the inside pass. Thus, there is a symmetry that does not exist in the usual algorithm. In addition, there is an efficient check on the correctness of the computation. This lies in the fact that for each word W in a given sentence S, the total count $\sum_{L,R,l,r} Count(W,L,R,l,r)$ must be equal to one, where the sum is taken over all L, R, l, and r which occur in a linkage of S.

Smoothing

Obtaining reliable estimates of the parameters of probabilistic language models is always a fundamental issue. In the case of the models proposed above, this is especially a concern due to the large number of parameters. Several methods of "smoothing" the estimates naturally suggest themselves. One such approach is to form the smoothed estimates

$$\widetilde{\Pr}(\, W \mid L, R, l, r\,) = \gamma^{-1}\delta_{l,r}(W)\left[\lambda \Pr(\, W \mid L, R\,) + (1-\lambda)\Pr(\, W \mid L, R, l, r\,)\right]$$

where $\delta_{l,r}(W)$ is equal to one in case the word W has a disjunct that can link to either l or r, and zero otherwise, and γ is a normalizing constant. This method of smoothing is attractive since the probabilities $\Pr(\, W \mid L, R\,)$ can be obtained from unparsed text. In fact, since for a given sentence S there are $\binom{|S|}{3}$ ways of choosing words that may potentially participate together in a linking, if we assume that the sentences in a corpus have lengths which are Poisson-distributed with a mean of 25, then there is an average of 2604 word triples per sentence, or approximately 100 times the number of usual trigrams. We can view the probability $\Pr(\, W \mid L, R\,)$ as the *prior* probability that the triple (L, W, R) forms a *grammatical trigram*.

Having obtained the maximum-likelihood estimates of the parameters of our model, we may then obtain the *posterior* probabilities of grammatical trigrams as

$$\widetilde{\Pr}(\, W \mid L, R\,) = \sum_{l,r} \widetilde{\Pr}(\, W \mid L, R, l, r\,)\Pr(\, l, r \mid L, R\,)$$

Here the probabilities $\Pr(\, l, r \mid L, R\,)$ are obtained through the joint probabilities $\Pr(\, L, R, l, r\,)$ which are estimated through the expected counts

$$Count(L, R, l, r) = \Pr_{\mathcal{O}}(L, R, l, r)\Pr_{\mathcal{I}}(L, R, l, r).$$

Further refinements to the smoothed distributions can be made using standard methods of deleted interpolation [1].

Prospects

The above class of models can be extended in many different directions. For example, decision trees can be used to estimate the probabilities, as we have in done in various other problems [4, 5]. Increasing the complexity of the models in this manner can promote the generative power to the class of context-sensitive languages. From a less formal point of view, such an extension would allow the statistics to better capture the long-range dependencies which are inherent in any large corpus. But the essence of the class of probabilistic models that has been proposed is that the parameters are highly lexical, though simple. In proceeding to actually carry out a program for constructing such models, one can at least begin to reach for the gauntlet [6] that has been thrown down in the name of the maligned trigram.

References

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer. A Maximum likelihood approach to continuous speech recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-5, No. 2, pp. 179-190, 1983.

[2] L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 627(3):1-8, 1972.

[3] E. Black, J. Lafferty, and S. Roukos. Development, evaluation, and results for a broad-coverage probabilistic grammar of English-language computer manuals. To appear in *Proceedings of the ACL*, 1992.

[4] E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, and S. Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the DARPA Speech and Natural Language Workshop*, Arden House, February 1992.

[5] E. Black, F. Jelinek, J. Lafferty, R. Mercer, and S. Roukos. Decision tree models applied to the labelling of text with parts-of-speech. In *Proceedings of the DARPA Speech and Natural Language Workshop*, Arden House, February 1992.

[6] P. Brown, S. Della Pietra, V. Della Pietra, J. Lai, and R. Mercer. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(2):31-40, 1992.

[7] H. Gaifman. Dependency systems and phrase-structure systems. *Information and Control* 8, 1965, Pages 304–337.

[8] F. Jelinek, J. D. Lafferty, and R. L. Mercer. Basic methods of probabilistic context-free grammars. In *Speech Recognition and Understanding: Recent Advances, Trends, and Applications,* P. Laface and R. De Mori, editors. Springer Verlag, Series F: Computer and Systems Sciences, vol. 75, 1992.

[9] F. Jelinek and J. D. Lafferty. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315-323, 1991.

[10] I. A. Meľčuk. *Dependency Syntax: Theory and Practice*, State University of New York Press 1988.

[11] R. T. Oehrle, E. Bach, and D. Wheeler, Editors. *Categorial Grammars and Natural Language Structures.* D. Reidel Publishing Company, 1988.

[12] Y. Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of COLING-92*, Nantes, France, July 1992.

[13] D. Sleator and D. Temperley. Parsing English with a Link Grammar. Technical report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, 1991.

[14] C. E. Shannon. Prediction and entropy of printed English. *Bell Syst. Tech. J.*, Vol. 30, pp. 50-64, 1951.