Link Grammar Prefix Measures for Spontaneous Speech Recognition

Doug Beeferman

December 20, 1995

Abstract

We show how the link grammar formalism can be applied as a left-to-right syntax model to spontaneous speech recognition tasks. We offer algorithms for robustly measuring the grammaticality of partial sentences with respect to a link grammar and examine the measure's behavior over the Wall Street Journal and Switchboard corpora. We discuss why link grammar is useful for tackling the problems that conversational speech poses and suggest various language modeling strategies that exploit the described techniques.

Software implementations of the techniques described in this paper can be reviewed at the reader's request by emailing `dougb+@cs.cmu.edu`. An interactive demonstration of the full-sentence robust parser is online at the URL "http://bobo.link.cs.cmu.edu/cgi-bin/grammar/build-intro-page.cgi".

1 Introduction

Speech recognition language models based only on local lexical dependencies can be improved by incorporating more grammatical structure, either from longer-range lexical relationships or syntactic constraints [15]. Integrating a syntax model into a speech recognition system typically means sacrificing one or more of the following:

- Coverage and robustness: Constructing a context-free grammar (CFG) or recursive transition network (RTN) that captures a large subset of English and gracefully handles ungrammatical input leads to an explosion in production rules or states, respectively. The latter requirement is crucial for spontaneous (conversational) speech, in which extragrammaticality and disfluency infest nearly every utterance. Wide-coverage grammars for general English tasks are difficult to write in these formalisms and unwieldy to maintain.
- Efficiency: Parsing, whether applied incrementally to an input utterance or as a postprocess after the first recognition search pass (which is often guided by an efficient stochastic n-gram model), is generally very slow relative to the rest of the system's language processing.
- Coupling: It is prohibitively expensive to parse at every word transition in the search, and the temptation is instead to process "offline", either by reranking a top-N hypothesis list generated by the recognizer or by rescoring an output search lattice. The first approach has failed to improve recognition error rates significantly, and has often hurt [15]. Furthermore, it is still very slow in the cases of large N-best lists, as per-hypothesis parsing fails to exploit the fact that successive entries in the list often differ by as little as a single function word. The second approach often requires that the lattice be pared down or split into manageable sub-lattices, as in the case of GLR* [6], and besides losing information, these algorithms become grossly inefficient when either dimension of the lattice becomes large.

When we survey the current offerings in spontaneous speech recognition we find efficient systems for limited domains such as the RTN-based Phoenix system [13] applied to the ATIS task, or inefficient systems for larger domains such as the GLR*-based JANUS system applied to the Switchboard telephone task [3]. Both applications sacrifice some accuracy by loosely coupling the recognition and parsing stages [14].

Given infinite computing resources, it would seem desirable for the syntactic processing of the utterance to parallel the approach to language model integration pioneered by the Sphinx system [7], in which linguistic knowledge is applied as soon as possible *during* the recognition search. This allows unlikely states to be pruned earlier, saving precious state space for more likely candidates. Postprocessing a lattice in any form is bound to sacrifice some attainable accuracy over what could be done online, since the lattice is a function of a pruning scheme that uses a language model presumably weaker than the combination of the syntax and stochastic models.

The spontaneous speech recognition problem compounds all of the concerns above. Since the words are on average significantly shorter than words in read speech[1], the greater acoustic confusability means greater state space growth, and this leads to monstrously large output lattices.

The criteria that a syntax model be at once expansive, efficient, and applicable incrementally during the recognition search seem vexing, but we intend to show in this paper that there may be an alternative that sacrifices none of them. Link grammar, a formalism based on word-pair relationships introduced by Sleator and Temperley [9], is gaining attention in the speech recognition community because it addresses the first two issues: It is straightforward and inexpensive, relative to other models, to create wide-coverage link grammars, and there are efficient and robust parsing algorithms that process them.

This paper addresses the third concern for link grammar in the context of spontaneous speech. We discuss *prefix parsing* and other techniques for measuring the grammaticality of an input stream in an online fashion. We offer a recursive algorithm for determining whether a word sequence can be the prefix of some grammatical parse and suggest a way to count valid prefix parses. We explore a measure of prefix grammaticality confidence, called *disjunct fanout*, based on the geometric mean of disjuncts that survive disjunct pruning, and we compare this to the output of the robust parser for link grammar [4]. Finally we show how and why a prefix scoring mechanism for link grammar such as disjunct fanout can be useful for processing spontaneous speech.

2 Link grammar

Link grammar is equivalent in expressive power to context-free grammar but it has a description scheme that lends itself to natural lan-

[1]Words in the Switchboard conversational corpus average 3.1 phonemes, compared to 4.2 phonemes in the *Wall Street Journal* (WSJ0) corpus.

guages like English. Constraints are formulated at the lexical level in terms of *linkage requirements*. In particular, each word in a link grammar *dictionary* is defined by a formula that encodes the combinations and orderings of left and right *connectors* of different types that must be linked with matching connectors in other words in the sentence. A sentence is said to be grammatical with respect to the grammar if there exists a valid *linkage* for the word sequence—a choice of matching connectors to link together such that each word's linkage requirements are satisfied, with the following two global restrictions:

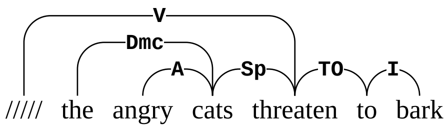
- Connectivity: The words and links form a connected graph
- Planarity: The graph can be drawn so that the links appear above the words without crossing

Here we introduce a simple example dictionary that will be used throughout the paper. Each line consists of words that share a common definition. Linkage requirements are specified in terms of the connector names and operators, including conjunction ("&"), disjunction ("or"), and optional ("{}"). Matching rules and details of the dictionary syntax are described in [9].

```
WALL-WORD: V+;
dog cat squirrel person: {A-} & D- & Ss+;
dogs cats squirrels people: {A-} & {Dmc-} & Sp+;
barks meows begs thinks: Ss- & {V-};
bark meow beg think: (Sp- & {V-}) or I-;
yak: ({A-} & D- & Ss+) or ((Sp- or I-) & {V-});
yaks: ({A-} & {Dmc-} & Sp+) or (Ss- & {V-});
barked meowed yakked burrowed thought: (S- or I- or T-) & {V-};
angry confused happy sad: A+;
may can would should: S- & I+;
wants threatens wishes hopes: ((Ss- & {V-}) or I-) & TO+;
want threaten wish hope: ((Sp- & {V-}) or I-) & TO+;
to: TO- & I+;
the: D+;
a: Ds+;
```

In the above dictionary we enforce agreement in number by giving right-pointing *Ss* and *Sp* connectors to singular and plural nouns, respectively, and left-pointing *Ss* and *Sp* connectors to verbs. We give *TO+* connectors to the verbs that take infinitives such as *threaten*, and *I-* connectors to the infinitives themselves so that they can left-link with *to*. Here *yak* and *yaks* may be used as nouns or verbs, so they are given the properties of both, ORed together. Finally, the "left wall word", indicated by slashes in the diagrams below, is used to enforce the presence of a verb in the sentence.

A sample linkage of the sentence "The angry cats threaten to bark" is given below.



A link grammar is stored internally in a *disjunctive* form that helps us to reason algorithmically about the parsing process: each word's formulaic definition is mapped to a logically equivalent set of *disjuncts*[2]. A disjunct is an ordered list of left and right connectors, all of which must be linked with matching connectors in the sentence to satisfy the word's linkage requirements. Exactly one disjunct per word is used in a valid linkage; this fact allows us to reason about a sentence in terms of disjunct sequences, the ramifications of which are explored in section 8.

The disjuncts for the words used in the example sentence are shown below, in the form (*left-connector-list, right-connector-list*), and those used in its linkage are in boldface.

`the:`	**(NULL, (D))**
`angry:`	**(NULL, (A))**
`cats:`	`((Dmc), (Sp)), (NULL, Sp),` **((Dmc, A), (Sp))**`, ((A), (Sp))`
`threaten:`	`((Sp), (TO)),` **((V, Sp), (TO))**`, ((I), (TO))`
`to:`	**((TO), (I))**
`bark:`	`((V, Sp), NULL), ((Sp), NULL),` **((I), NULL)**

3 The sentence parsing algorithm

Here we review the standard link grammar parsing algorithm for complete sentences and show how it can be extended to handle sentence prefixes. The reader is referred to [9] for details on the pseudocode. *left[d]* and *right[d]* denote the first connector in a linked list of connectors representing the left and right sides of a disjunct d. Parse counts the number of valid linkages of a sentence by summing, over each disjunct d of the first word in the sentence, the number of valid linkages that use d.

[2]We use this terminology to be consistent with previous literature on link grammar; these should properly be called conjuncts.

Parse

```
t ← 0
for each disjunct d of word 0
    do if left[d] = NIL
          then t ← t + COUNT(0, N, right[d], NIL)
return t
```

Count takes as parameters the left and right words of a region and pointers into two linked lists of connectors, and returns the number of valid linkages of that region using all of the connectors in the two sublists.

Count(L, R, l, r)

```
if L = R + 1
   then if l = NIL and r = NIL
           then return 1
           else return 0
   else total ← 0
        for W ← L + 1 to R − 1
            do for each disjunct d of word W
                   do if l ≠ NIL and left[d] ≠ NIL and MATCH(l, left[d])
                         then leftcount ← COUNT(L, W, next[l], next[left[d]])
                         else leftcount ← 0
                      if right[d] ≠ NIL and r ≠ NIL and MATCH(right[d], r)
                         then rightcount ← COUNT(W, R, next[right[d]], next[r])
                         else rightcount ← 0
                      total ← total + leftcount * rightcount
                      if rightcount > 0 and l = NIL
                         then total ← total + rightcount * COUNT(L, W, l, left[d])
                      if leftcount > 0
                         then total ← total + leftcount * COUNT(W, R, right[d], r)
        return total
```

Count selects a splitting point (a word within the region and one of the word's disjuncts) and invokes itself recursively to find the number of parses of the regions to the left and to the right of the split word. The total count is the sum, for each such splitting point, of the number of combinations of left and right sub-parses that use one or both of the linked sides.

The parsing algorithm's running time is cubic in the length of the input sentence, and its maximal recursion depth is bounded by it. A number of speed optimizations are necessary for efficiency on the very long, rambling utterances that are occasionally found in conversational speech: Memoization of the subregion counts speeds up the enumeration; disallowing links longer than a certain length reduces the number of split candidates; and pruning of the disjuncts before the parse, discussed in section 4.3, greatly reduces the work to be done.

4 Prefix parsing

There are soft and hard problems in prefix parsing. A speech recognizer may wish to know at a given transition an answer to the binary question, *are the words processed thus far a prefix of a grammatical utterance?* The answer could constrain the recognition search, either rigidly (as is common with small-vocabulary systems) or by means of a penalty; or the answer could be integrated into a proper probability-generating language model.

A harder but more important question is *how likely are these words a grammatical prefix?* We can't hope to answer that question accurately with just a syntax model, if at all. The denominator is ill-defined: For all practical purposes there is an infinitude of possible sentence completions after a given prefix, and even if we believe this number to be bounded, treating each as equiprobable is the best we can do.

Consider a confidence measure that is based, rather than on how many different ways there are to complete the sentence, on how many different interpretations there are of the prefix. This figure is well-defined for context-free grammars and other formalisms that have prefix parsing algorithms.

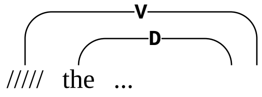
As we uncover a sentence, the number of prefix parses varies. After revealing the first word in Chomsky's famous "Time flies like an arrow", we have as many different prefix parses as there are part-of-speech labelings for *time*; after the second word we can interpret the prefix as an imperative, a subject-verb pair, or a noun phrase; and after the fifth word we have at least as many prefix parses as there are sentence parses. The dynamics of the count provides insight on the ambiguity of the prefix and the likely grammaticality of the utterance to follow; a count of zero implies that the sentence will surely be asyntactic.

4.1 Definitions

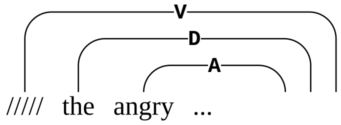
Returning to link grammars, we find that the model inspires a natural definition of a prefix parse. Define a valid *prefix linkage* to be a linkage that, like a sentence linkage, obeys the planarity restriction, but not necessarily the connectivity restriction. Furthermore, enforce in prefix linkages the left connection requirements of each word, but allow the right connectors of a disjunct to *dangle*, unconnected, past the right side of the word sequence. We will define the *prefix count* of a word sequence to be the number of such prefix linkages.

All such prefix linkages may potentially be completed by later words in the input stream, assuming it is lexically feasible [3].

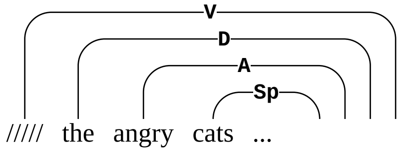
As there are possibly many sentence linkages for a given sentence, there can be several prefix linkages for a given grammatical prefix. Consider our example sentence. After the first word, there is only one prefix linkage:



After the second word there's still only one:

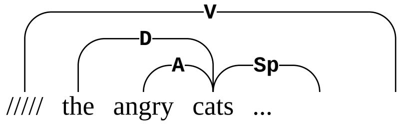


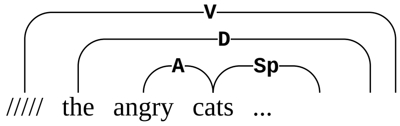
But when we hit *cats*, three possibilities arise depending on our choice of disjunct for the word: we could choose the second disjunct of *cats* and allow all the connectors in the prefix to dangle; or the third disjunct, and let just the *Sp* and *V* connectors dangle; or the fourth disjunct, and let *Sp*, *D*, and *V* dangle. It is not legal to choose the first disjunct, ((Dmc), (Sp)), since linking the *D* connector with the first word would isolate the second word, *angry*, under a link, making it impossible to connect *angry* to a future word. [4]



[3] There must exist words in the dictionary that satisfy the right connection requirements of the prefix words. The prefix count ignores this restriction. That is to say, it is an upper bound on what is in some sense the "true" prefix count.

[4] Note that true English grammar would permit only the second prefix linkage, as the other two attempt to link the article *the* with a word beyond the head of the noun phrase. In link grammar, this upper bound on the length of the *D* link is enforced not by the connection requirements of the determiner itself or of the head noun, but in some sense by their mutual presence in a complete sentence.





If we continue to process the sentence, we find that there are six prefix parses after *threaten*, after *to*, and after *bark*.

4.2 The algorithm

Here we describe an algorithm PrefixParse for enumerating the prefix linkages of a given word sequence. Analogously to Parse, it calls a routine PrefixCount for each disjunct in the first word. We arrive at PrefixCount by making the following changes and additions to Count:

```
PrefixCount(L, R, l, r)
if L = R + 1
   then if r = NIL
          then return 1
          else return 0
   else total ← 0
        for W ← L + 1 to R − 1
            do for each disjunct d of word W
 [...]
                      if leftcount > 0
                         then total ← total + leftcount * PrefixCount(W, R, right[d], r)
                      if next[right[d]] ≠ NIL
                         then rightdanglecount ← PrefixCount(W, R, next[right[d]], r)
                         else rightdanglecount ← 0
                      total ← total + leftcount * rightdanglecount
                      if l = NIL and (rightcount + rightdanglecount) > 0
                         then total ← total + (rightcount + rightdanglecount) *
                                          Count(L, W, l, left[d])
        if l ≠ NIL
          then total ← total + PrefixCount(L, R, next[l], r)
        return total
```

Everything counted by COUNT should also be counted by PREFIX-COUNT, as an acceptable sentence parse is also an acceptable prefix. In addition, we count cases in which the left connector of the region dangles by invoking PREFIXCOUNT recursively on the same region but with the next connector in the left disjunct's right connector list (lines 26 and 27). Furthermore we allow the right connector at the splitting point to dangle (lines 19 through 21).

Note that the base case of the recursion, in which the boundary words are right next to each other (lines 1 through 4), is changed: we no longer require that the left connector be NIL, and we count the case in which all the remaining connectors in the list L dangle.

The softer question, of whether there *exists* a prefix parse, may be computed more efficiently than inspecting the return value of PREFIX-PARSE by modifying PREFIXCOUNT to return true as soon as the total becomes nonzero, and false only at the end of the routine.

4.3 Efficiency considerations

PREFIXPARSE has exponential worst-case running time [11, 12] without memoization. Even after applying optimizations analogous to the tricks used in the full-sentence parser it is much slower than PARSE—there is so much more to count.

In the standard sentence parsing algorithm, a stage of disjunct pruning before PARSE is invoked saves considerable enumeration time [5]. Disjuncts that cannot possibly exist in a valid parse are removed. The most obvious reason to exclude a disjunct d from a word w is if there is a connector in the left-connector list of d such that none of the words to the left of w have it as a right-connector, or if there's a connector in the right-connector list such that none of the words to the right of w have it as a left-connector. We can use this criterion in a "prefix pruning" stage before prefix parsing by enforcing only the first requirement in a single left-to-right pass; we can't enforce the right-hand-side requirements since we know nothing about what connectors could appear in the future. The full-sentence parser's pruning algorithm also quickly checks a number of potential violations of the connectivity and planarity requirements before proceeding, the specifics of which are outlined in [9]. These too can be incorporated into prefix pruning, though less effectively, with one left-to-right pass.

Since prefix pruning is necessarily weaker than the standard full-sentence pruning algorithm, many more disjuncts remain after this step. At a minimum there are as many prefix parses as the product of

[5] Actually there are two stages, called "pruning" and "power pruning", in [9], but we will consolidate them in this paper since the latter stage subsumes the former in effect.

the counts, for each word, of disjuncts that have empty left connector lists—the cases in which all right-connectors dangle. Most words in the general English grammars constructed so far have several such disjuncts; though most combinations of these in a sentence prefix make little sense, they cannot be ruled out until it is known that the sentence is finished, and so the total grows exponentially with the length of the prefix.

5 Relaxed prefix parsing

Let us turn our attention to speech recognition again and imagine that the algorithm above is invoked at every word transition in a large-vocabulary recognition search. Even after pruning, the strict prefix parsing algorithm is too slow for the critical path, no better than incremental versions of its CFG [10] and RTN brethren.

In many speech applications it is necessary or useful to know the syntactic structure of the final output hypothesis, but for transcription tasks, and in the absence of a parse disambiguation or scoring mechanism, it is wasteful to spend search time resolving structure. For we are not as concerned during recognition with what *kind* of syntactic structure is present in the input as we are with how *much* structure there is. More precisely we make the assumption that input utterances tend to be somewhat grammatical, and we want to take advantage of this in prediction by measuring just *how* grammatical partial hypotheses are, and favoring those with *more* structure.

In this section we suggest a measure of prefix grammaticality related to an obvious upper bound of the strict prefix count. In particular, consider the geometric mean of the disjunct count at each word after prefix pruning, or

$$\left(\prod_{i=1}^{n} d(w_i)\right)^{1/n},$$

where $d(w_i)$ is the number of disjuncts surviving at word i and n is the number of words in the prefix. The product itself is an approximate and loose upper bound on the prefix count, since a parse can be regarded as an assignment of each word to one of its disjuncts.[6]

The measure, which we will refer to as *disjunct fanout*, is similar to language model perplexity in that it estimates the average number of transition choices from any state. Like perplexity it may be efficiently computed in the log domain.

[6]For a given parse there is exactly one corresponding disjunct sequence; for a given disjunct sequence there is usually, but not necessarily, only one possible linkage.

the	cats	threaten	to	bark				
2.0	2.4	2.6	2.4	2.7				
the	bark	threaten	cats	to				
2.0	1.4	1.3	1.6	1.4				
the	confused	yaks	can	hope	to	yak		
2.0	1.4	1.8	1.9	2.2	2.1	2.5		
so	that	test	is	relevant	for	that	very	day...
5.0	11.4	21.7	46.3	22.3	22.3	24.0	20.6	28.4
well	and	I	worry	you	know	the	older	my
6.0	4.2	4.5	12.3	7.0	8.1	8.6	8.8	8.1
children	get	the	more...					
11.2	13.6	13.7	14.5					
I	mean	uh	I	build	a	uh	I	have
5.0	11.4	5.1	5.0	4.1	5.5	4.5	4.5	5.3
a	stock	English	garden	bench...				
6.3	8.3	9.2	12.2	15.5				

Figure 1: The disjunct fanout throughout various prefixes. The first three were processed with the small example dictionary, and the remaining are from the Switchboard corpus and were processed with a general English link grammar of 58,873 words. Observe how the measure drops off and then rises when a region of ungrammaticality is entered and exited.

Just as smoothing is required for stochastic language models to model unobserved sequences with nonzero probability, we have to account for the cases in which a word is left with no disjuncts after prefix pruning. Strictly speaking a sequence containing such a word must have a prefix count of zero, but any continuous grammaticality measure should take into account the structure of the remaining words. Smoothing is achieved simply by adding an implicit "null" disjunct to each word that cannot be pruned, and we will see in section 7 why this is consistent with other robustness approaches. The disjunct count for each word is therefore at least one, assuring that the product and geometric mean are positive.

Link grammar is ideal in that it is able to capture the structure of a word sequence even in the presence of out-of-vocabulary words, insertions and a number of other disfluency phenomena (see section 7), and this is reflected in our measure. The fluctuation of the disjunct fanout for some sentences in the Switchboard corpus is shown in figure 5.

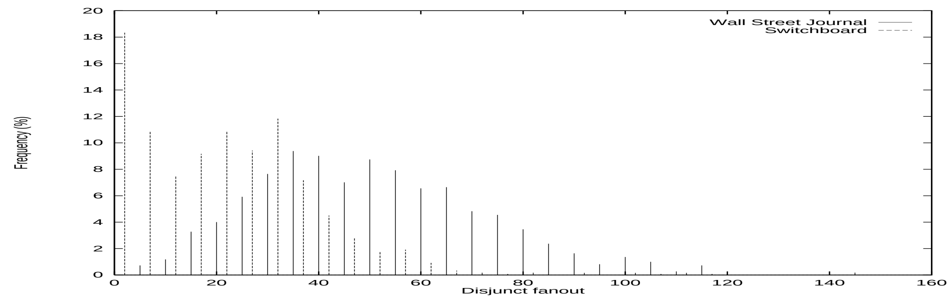


Figure 2: Histogram of disjunct fanouts for complete sentences in the WSJ0 and Switchboard corpora. Each bin corresponds to five units of fanout.

6 Measurements

The disjunct fanout is closely tied to how the link grammar dictionary is written, so it is not comparable across grammars. Its intended application is to help compare word transitions from the same source state in a speech recognizer, or to penalize partial or complete utterances with severity inversely related to the measure.

To investigate disjunct fanout as an indicator of grammaticality, we processed subsets of two large text corpora with the general English grammar: the *Wall Street Journal* corpus (WSJ0) from 1987 to 1989, and the Switchboard corpus. The former is representative of carefully written prose, the latter of spontaneous telephone speech.

The disjunct fanouts for the final word in 2000 sentences from each corpus were computed, and a histogram of the results is shown in figure 2. As expected, the WSJ0 sentences scored significantly higher.

It should be noted that the sentences in Switchboard are much shorter than in WSJ0, averaging for these tests 12.5 words compared to 22.6. While the disjunct fanout is normalized for sentence length, there is still a bias that favors longer sentences inherent in the measure: The left-to-right pruning passes eliminate more disjuncts from the beginning few words than from later words because the set of available right connectors continually grows. For this reason we found it useful to observe the disjunct fanout a fixed length n into each sentence. The results for $n = 5$ are shown in figure 3.

7 Link grammar and spontaneous speech

While the short average word duration in spontaneous speech mentioned in section 1 contributes to *acoustic* confusability, a host of extra-

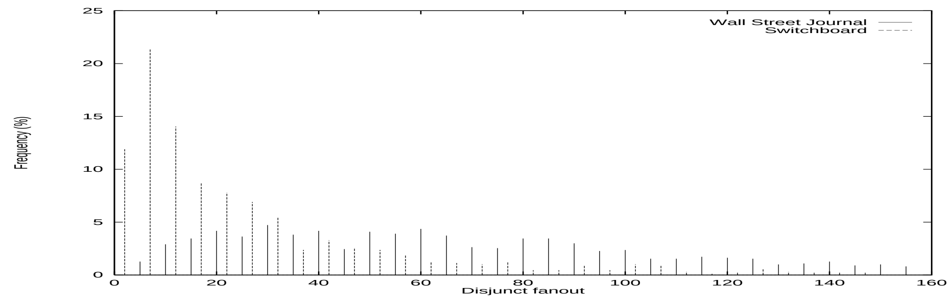


Figure 3: Histogram of disjunct fanouts for partial sentences (after $n = 4$ words) in the WSJ0 and Switchboard corpora

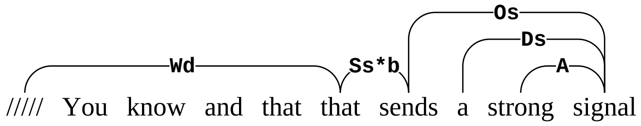
grammatical phenomena in conversational text streams make language modeling a nightmare. False starts, ellipsis, repetition, and a variety of inserted pause words such as "um", "uh", and "like", break the flow of syntax. This is compounded by the looser notion of grammaticality that speakers obey: even disfluency-free sentences do not necessarily parse.

A common approach in robust parsing is to search for and extract syntax from the largest grammatical subset of the input, ignoring out-of-vocabulary words and, ideally, disfluencies.

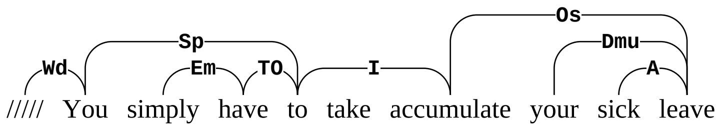
Grinberg, et al [4] have devised a robust parsing algorithm for link grammar that allows islands of connected components in an unconnected parse candidate to be joined by *null links*. Null links can connect any adjacent word pair, even out-of-vocabulary words, making extraction of grammatical structure possible for any input string. In the robust parser, linkages are assigned a cost that increases linearly with the number of null links used, essentially crowning as optimal the parse with the largest subset of words linked. The parser may thus be thought of as a cost calculation algorithm, with zero cost implying totally grammatical input, or as a filter that eliminates useless words in the input stream before language model scoring. Examples of how the link grammar parses a few ill-formed Switchboard sentences are shown in figure 4.

The concept of a null link is implicit in the aforementioned fanout-based confidence measure. The measure's advantage over the robust parser's cost function is continuity; its major relative weakness is coarseness.

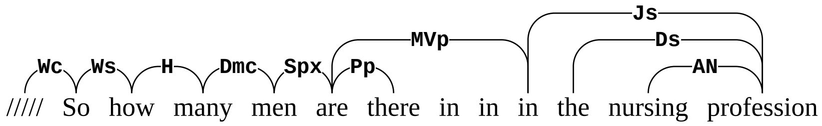
The robust parser and the fanout measure are intended to handle *inserted* disfluencies in text, as both schemes are based on deletion. Deletions are much less common than insertions in speech as they reduce meaning, but occasionally we find the omission of function



a. Filler words: "You know, and that... that sends a strong signal"



b. False starts: "You simply have to take... accumulate your sick leave"



c. Stuttering: "So how many men are there in in in the nursing profession?"

Figure 4: Examples of how the robust link parser reacts to three common problems in conversational speech: filler words, false starts, and repetition. The three sentences were taken from the Switchboard corpus and processed with a 58,873-word link grammar dictionary. Note how unmeaningful words are skipped while the essences of the sentences are extracted.

words such as articles. As one user of our World Wide Web-based link grammar demonstration software quipped self-referentially, "Newspaper headlines confuse link parser"; this quote indeed confounds the full sentence parser and has a very low disjunct fanout of 8.5. The fanout of the grammatical "Newspaper headlines confuse *the* link parser", by contrast, is 32.8.

8 Conclusions and future work

The power of link grammar lies in its representation of knowledge at the lexical level, with the disjunct as the key information-bearing unit. If a parse is interpreted as an assignment of words to disjuncts, then we can treat the disjunct sequence as a stream of discrete syntactic *usages*, each symbol of which expresses the unique relationship of the word to the others in the sentence more accurately than part-of-speech tags or any lexically-deterministic word classification scheme.

For this reason studies are underway to analyze texts both written and spoken in terms of disjunct sequences, with the eventual goal of building a proper language model. One simple trigram-based approach models the probabilities $p(d_n|d_{n-1}, d_{n-2})$ for all disjunct triples, and the *a posteriori* probabilities $p(w|d_i)$ for all (word, disjunct) pairs. Such a model, interpolated with a purely lexical short-range stochastic model, could capture everything offered by the latter with the added benefit of long-distance syntax dependencies.

By contrast, a generative model, suggested by Lafferty, et al [5], views a parse in terms of the links that are used and which words they connect, and models a parse probability as the product of the constituent link probabilities. A left to right implementation of the decoding algorithm for this model could employ the prefix parsing algorithm described in section 4 to enumerate the prefix linkages, and score the prefix with the sum of all prefix linkage probabilities. The drawback here is that it would be painfully slow for large grammars because of the combinatorial explosion described in section 4.3.

Link grammar offers wide coverage and efficiency, and its highly lexical orientation lends itself to meaningful and computationally inexpensive measures. Yet it has not been proven to aid speech recognition. We believe that carefully integrating grammaticality measures such as the quantity proposed in this paper with existing language modeling strategies, or devising complete syntax-aware language models such as the one described above, can go a long way toward reducing perplexity and lowering word error rates for both read and conversational tasks.

Still, to approach the estimates of the entropy of English offered by Shannon [8]; Cover and King [2]; and others, it is clear that semantic and pragmatic constraints need to be harnessed in addition to syntax and local lexical relationships [1]. Neither a syntax model nor a short-range stochastic model, nor any combination of the two, can tell us that cats don't bark.

References

[1] P. E. Brown, S. A. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, Mar 1992.

[2] T. M. Cover and R. King. A convergent gambling estimate of the entropy of English. *IEEE Transactions on Information Theory*, 24:413–421, Mar 1978.

[3] J. Godfrey, E. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research development. In *Proc. ICASSP-92*, pages I–517–520, 1992.

[4] D. Grinberg, J. Lafferty, and D. Sleator. A robust parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, Prague/Karlovy Vary, Czech Republic, 1995. Also issued as Technical Report CMU-CS-95-125, School of Computer Science, Carnegie Mellon University, 1995.

[5] J. Lafferty, D. Sleator, and D. Temperley. Grammatical trigrams: A probabilistic model of link grammar. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, Cambridge, MA, October 1992. Also issued as Technical Report CMU-CS-92-181, School of Computer Science, Carnegie Mellon University, 1992.

[6] A. Lavie and M. Tomita. GLR*: An efficient noise-skipping parsing algorithm for context free grammars. In *Proceedings of the Third International Workshop on Parsing Technologies*, pages 123–134, 1993.

[7] K.F. Lee, H.W. Hon, and R. Reddy. An overview of the SPHINX speech recognition system. *Journal of Acoustics, Speech, and Signal Processing*, 38(1), January 1990.

[8] C. E. Shannon. Prediction and entropy of printed English. *Bell Systems Technical Journal*, 30:50–64, 1951.

[9] D. D. K. Sleator and D. Temperley. Parsing English with a link grammar. Technical Report CMU-CS-91-196, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, 1991.

[10] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, International Computer Science Institute, 1947 Center St.. Berkeley, CA 94704, 1993. Revised November 1994.

[11] S. Vempala. Application of the master theorem for solving recurrence relations. Personal correspondence, December 1995.

[12] S. Vempala. On the exponential running time of PrefixCount. Barroom brawl, December 1995.

[13] W. Ward. Understanding spontaneous speech: the Phoenix system. In *Proc. ICASSP-91*, pages I–365–367, 1991.

[14] W. Ward. Extracting information in spontaneous speech. Presented at the International Conference for Spoken language processing, September 1994.

[15] S.R. Young, Alexander G. Hauptmann, Wayne H. Ward, Edward T. Smith, and Philip Werner. High level knowledge sources in usable speech recognition systems. *Communications of the ACM*, 32(2), February 1989.