Grammatical Category Disambiguation by Statistical Optimization

Steven J. DeRose

Brown University and the Summer Institute of Linguistics, 7500 W. Camp Wisdom Road, Dallas, TX 75236

Several algorithms have been developed in the past that attempt to resolve categorial ambiguities in natural language text without recourse to syntactic or semantic level information. An innovative method (called "CLAWS") was recently developed by those working with the Lancaster –Oslo/Bergen Corpus of British English. This algorithm uses a systematic calculation based upon the probabilities of co-occurrence of particular tags. Its accuracy is high, but it is very slow, and it has been manually augmented in a number of ways. The effects upon accuracy of this manual augmentation are not individually known.

The current paper presents an algorithm for disambiguation that is similar to CLAWS but that operates in linear rather than in exponential time and space, and which minimizes the unsystematic augments. Tests of the algorithm using the million words of the Brown Standard Corpus of English are reported; the overall accuracy is 96%. This algorithm can provide a fast and accurate front end to any parsing or natural language processing system for English.

Every computer system that accepts natural language input must, if it is to derive adequate representations, decide upon the grammatical category of each input word. In English and many other languages, tokens are frequently ambiguous. They may represent lexical items of different categories, depending upon their syntactic and semantic context.

Several algorithms have been developed that examine a prose text and decide upon one of the several possible categories for a given word. Our focus will be on algorithms which specifically address this task of **disambiguation**, and particularly on a new algorithm called **VOLSUNGA**, which avoids syntactic-level analysis, yields about 96% accuracy, and runs in far less time and space than previous attempts. The most recent previous algorithm runs in NP (Non-Polynomial) time, while VOLSUNGA runs in linear time. This is provably optimal; no improvements in the order of its execution time and space are possible. VOLSUNGA is also robust in cases of ungrammaticality.

Improvements to this accuracy may be made, perhaps the most potentially significant being to include some higher-level information. With such additions, the accuracy of statistically-based algorithms will approach 100%; and the few remaining cases may be largely those with which humans also find difficulty.

In subsequent sections we examine several disambiguation algorithms. Their techniques, accuracies, and efficiencies are analyzed. After presenting the research carried out to date, a discussion of VOLSUNGA's application to the **Brown Corpus** will follow. The Brown Corpus, described in Kucera and Francis (1967), is a collection of 500 carefully distributed samples of English text, totalling just over one million words. It has been used as a standard sample in many studies of English. Generous advice, encouragement, and assistance from Henry Kucera and W. Nelson Francis in this research is gratefully acknowledged.

1 Previous Disambiguation Algorithms

The problem of **lexical category ambiguity** has been little examined in the literature of computational linguistics and artificial intelligence, though it pervades English to an astonishing degree. About 11.5% of **types** (vocabulary), and over 40% of **tokens** (running words) in English prose are categorically ambiguous (as measured via the Brown Corpus). The vocabulary breaks down as shown in Table 1 (derived from Francis and Kucera (1982)).

0362-613X/88/010031-39$03.00

Number of words by degree of ambiguity:	
Unambiguous (1 tag)	35340
Ambiguous (2-7 tags)	4100
2 tags	3760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 ("still")

Table 1: Degrees of Ambiguity

A search of the relevant literature has revealed only three previous efforts directed specifically to this problem. The first published effort is that of Klein and Simmons (1963), a simple system using suffix lists and limited frame rules. The second approach to lexical category disambiguation is **TAGGIT** (Greene and Rubin (1971)), a system of several thousand context-frame rules. This algorithm was used to assign initial tags to the Brown Corpus. Third is the **CLAWS** system developed to tag the **Lancaster –Oslo/Bergen (or LOB) Corpus.** This is a corpus of British written English, parallel to the Brown Corpus. Parsing systems always encounter the problem of category ambiguity; but usually the focus of such systems is at other levels, making their responses less relevant for our purposes here.

1.1 KLEIN AND SIMMONS

Klein and Simmons (1963) describe a method directed primarily towards the task of initial categorial tagging rather than disambiguation. Its primary goal is avoiding "the labor of constructing a very large dictionary" (p. 335); a consideration of greater import then than now.

The Klein and Simmons algorithm uses a palette of 30 categories, and claims an accuracy of 90% in tagging. The algorithm first seeks each word in dictionaries of about 400 function words, and of about 1500 words which "are exceptions to the computational rules used" (p. 339). The program then checks for suffixes and special characters as clues.

Last of all, **context frame tests** are applied. These work on scopes bounded by unambiguous words, as do later algorithms. However, Klein and Simmons impose an explicit limit of three ambiguous words in a row. For each such **span** of ambiguous words, the pair of unambiguous categories bounding it is mapped into a list. The list includes all known sequences of tags occurring between the particular bounding tags; all such sequences of the correct length become candidates. The program then matches the candidate sequences against the ambiguities remaining from earlier steps of the algorithm. When only one sequence is possible, disambiguation is successful.

The samples used for calibration and testing were limited. First, Klein and Simmons (1963) performed "hand analysis of a sample [size unspecified] of Golden Book Encyclopedia text" (p. 342). Later, "[w]hen it was run on several pages from that encyclopedia, it correctly and unambiguously tagged slightly over 90% of the words" (p. 344). Further tests were run on small samples from the *Encyclopedia Americana* and from *Scientific American.*

Klein and Simmons (1963) assert that "[o]riginal fears that sequences of four or more unidentified parts of speech would occur with great frequency were not substantiated in fact" (p. 3). This felicity, however, is an artifact. First, the relatively small set of categories reduces ambiguity. Second, a larger sample would reveal both (a) low-frequency ambiguities and (b) many long spans, as discussed below.

1.2 GREENE AND RUBIN (TAGGIT)

Greene and Rubin (1971) developed TAGGIT for tagging the Brown Corpus. The palette of 86 tags that TAGGIT uses has, with some modifications, also been used in both CLAWS and VOLSUNGA. The rationale underlying the choice of tags is described on pages 3–21 of Greene and Rubin (1971). Francis and Kucera (1982) report that this algorithm correctly tagged approximately 77% of the million words in the Brown Corpus (the tagging was then completed by human post-editors). Although this accuracy is substantially lower than that reported by Klein and Simmons, it should be remembered that Greene and Rubin were the first to attempt so large and varied a sample.

TAGGIT divides the task of category assignment into initial (potentially ambiguous) tagging, and disambiguation. Tagging is carried out as follows: first, the program consults an exception dictionary of about 3,000 words. Among other items, this contains all known closed-class words. It then handles various special cases, such as words with initial "$", contractions, special symbols, and capitalized words. The word's ending is then checked against a suffix list of about 450 strings. The lists were derived from lexicostatistics of the Brown Corpus. If TAGGIT has not assigned some tag(s) after these several steps, "the word is tagged NN, VB, or JJ [that is, as being three-ways ambiguous], in order that the disambiguation routine may have something to work with" (Greene and Rubin (1971), p. 25).

After tagging, TAGGIT applies a set of 3300 context frame rules. Each rule, when its context is satisfied, has the effect of deleting one or more candidates from the list of possible tags for one word. If the number of candidates is reduced to one, disambiguation is considered successful subject to human post-editing. Each rule can include a scope of up to two unambiguous words on each side of the ambiguous word to which the rule is being applied. This constraint was determined as follows:

> In order to create the original inventory of Context Frame Tests, a 900-sentence subset of the Brown University Corpus was tagged. . . and its ambiguities were resolved manually; then a program was run

which produced and sorted all possible Context Frame Rules which would have been necessary to perform this disambiguation automatically. The rules generated were able to handle up to three consecutive ambiguous words preceded and followed by two non-ambiguous words [a constraint similar to Klein and Simmons']. However, upon examination of these rules, it was found that a sequence of two or three ambiguities rarely occurred more than once in a given context. Consequently, a decision was made to examine only one ambiguity at a time with up to two unambiguously tagged words on either side. The first rules created were the results of informed intuition (Greene and Rubin (1972), p. 32).

1.3 CLAWS

Marshall (1983, p. 139) describes the LOB Corpus tagging algorithm, later named CLAWS (Booth (1985)), as "similar to those employed in the TAGGIT program". The tag set used is very similar, but somewhat larger, at about 130 tags. The dictionary used is derived from the tagged Brown Corpus, rather than from the untagged. It contains 7000 rather than 3000 entries, and 700 rather than 450 suffixes. CLAWS treats plural, possessive, and hyphenated words as special cases for purposes of initial tagging.

The LOB researchers began by using TAGGIT on parts of the LOB Corpus. They noticed that

> While less than 25% of TAGGIT's context frame rules are concerned with only the immediately preceding or succeeding word. . . these rules were applied in about 80% of all attempts to apply rules. This relative overuse of minimally specified contexts indicated that exploitation of the relationship between successive tags, coupled with a mechanism that would be applied throughout a sequence of ambiguous words, would produce a more accurate and effective method of word disambiguation (Marshall (1983), p. 141).

The main innovation of CLAWS is the use of a matrix of **collocational probabilities**, indicating the relative likelihood of co-occurrence of all ordered pairs of tags. This matrix can be mechanically derived from any pre-tagged corpus. CLAWS used "[a] large proportion of the Brown Corpus", 200,000 words (Marshall (1983), pp. 141, 150).

The ambiguities contained within a span of ambiguous words define a precise number of complete sets of mappings from words to individual tags. Each such assignment of tags is called a **path.** Each path is composed of a number of tag collocations, and each such collocation has a probability which may be obtained from the collocation matrix. One may thus approximate each path's probability by the product of the probabilities of all its collocations. Each path corresponds to a unique assignment of tags to all words within a span. The paths constitute a **span network,** and the path of maximal probability may be taken to contain the "best" tags.

Marshall (1983) states that CLAWS "calculates the most probable sequence of tags, and in the majority of cases the correct tag for each individual word corresponds to the associated tag in the most probable sequence of tags" (p. 142). But a more detailed examination of the Pascal code for CLAWS revealed that CLAWS has a more complex definition of "most probable sequence" than one might expect. A probability called "SUMSUCCPROBS" is predicated of each word. SUMSUCCPROBS is calculated by looping through all tags for the words immediately preceding, at, and following a word; for each tag triple, an increment is added, defined by:

```
DownGrade(GetSucc(Tag2, Tag3), TagMark) *
  Get3SeqFactor(Tag1, Tag2, Tag3)
```

GetSucc returns the collocational probability of a tag pair; *Get3SeqFactor* returns either 1, or a special value from the tag-triple list described below. *DownGrade* modifies the value of *GetSucc* in accordance with RTPs as described below.

The CLAWS documentation describes SUMSUCCPROBS as "the total value of all relationships between the tags associated with this word and the tags associated with the next word. . . [found by] simulating all accesses to SUCCESSORS and ORDER2VALS which will be made. . . ." The probability of each node of the span network (or rather, tree) is then calculated in the following way as a tree representing all paths through which the span network is built:

```
PROB = DownGrade(GetSucc(lasttag,
currenttag), TagMark) *
  Get3SeqFactor(. . .))
PROB = PROB /(predecessor's
SUMSUCCPROBS) * (predecessor's PROB)
```

It appears that the goal is to make each tag's probability be the summed probability of *all* paths passing through it. At the final word of a span, pointers are followed back up the chosen path, and tags are chosen en route.

We will see below that a simpler definition of optimal path is possible; nevertheless, there are several advantages of this general approach over previous ones.

First, spans of unlimited length can be handled (subject to machine resources). Although earlier researchers (Klein and Simmons, Greene and Rubin) have suggested that spans of length over 5 are rare enough to be of little concern, this is not the case. The number of spans of a given length is a function of that length and the corpus size; so long spans may be obtained merely by examining more text. The total numbers of spans in the Brown Corpus, for each length from 3 to 19, are: 397111, 143447, 60224, 26515, 11409, 5128, 2161, 903, 382, 161, 58, 29, 14, 6, 1, 0, 1. Graphing the logarithms

of these quantities versus the span length for each, produces a near-perfect straight line.

Second, a precise mathematical definition is possible for the fundamental idea of CLAWS. Whereas earlier efforts were based primarily on ad hoc or subjectively determined sets of rules and descriptions, and employed substantial exception dictionaries, this algorithm requires no human intervention for set-up; it is a systematic process.

Third, the algorithm is quantitative and analog, rather than artificially discrete. The various tests and frames employed by earlier algorithms enforced absolute constraints on particular tags or collocations of tags. Here relative probabilities are weighed, and a series of very likely assignments can make possible a particular, a priori unlikely assignment with which they are associated.

In addition to collocational probabilities, CLAWS also takes into account one other empirical quantity:

> Tags associated with words. . . can be associated with a marker @ or %; @ indicates that the tag is infrequently the correct tag for the associated word(s) (less than 1 in 10 occasions), % indicates that it is highly improbable. . . (less than 1 in 100 occasions). . . . The word disambiguation program currently uses these markers top devalue transition matrix values when retrieving a value from the matrix, @ results in the value being halved, % in the value being divided by eight (Marshall (1983), p. 149).

Thus, the independent probability of each possible tag for a given word influences the choice of an optimal path. Such probabilities will be referred to as **Relative Tag Probabilities**, or **RTPs**.

Other features have been added to the basic algorithm. For example, a good deal of suffix analysis is used in initial tagging. Also, the program filters its output, considering itself to have failed if the optimal tag assignment for a span is not "more than 90% probable". In such cases it reorders tags rather than actually disambiguating. On long spans this criterion is effectively more stringent than on short spans. A more significant addition to the algorithm is that

> a number of tag triples associated with a scaling factor have been introduced which may either upgrade or downgrade values in the tree computed from the one-step matrix. For example, the triple [1] 'be' [2] adverb [3] past-tense-verb has been assigned a scaling factor which downgrades a sequence containing this triple compared with a competing sequence of [1] 'be' [2] adverb [3]-past-participle/adjective, on the basis that after a form of 'be', past participles and adjectives are more likely than a past tense verb (Marshall (1983), p. 146).

A similar move was used near conjunctions, for which the words on either side, though separated, are more closely correlated to each other than either is to the conjunction itself (Marshall (1983), pp. 146–147). For example, a verb/noun ambiguity conjoined to a verb should probably be taken as a verb. Leech, Garside, and Atwell (1983, p. 23) describe "IDIOMTAG", which is applied after initial tag assignment and before disambiguation. It was

> developed as a means of dealing with idiosyncratic word sequences which would otherwise cause difficulty for the automatic tagging. . . . for example, *in order that* is tagged as a single conjunction. . . . The Idiom Tagging Program. . . can look at any combination of words and tags, with or without intervening words. It can delete tags, add tags, or change the probability of tags. Although this program might seem to be an *ad hoc* device, it is worth bearing in mind that any fully automatic language analysis system has to come to terms with problems of lexical idiosyncrasy.

IDIOMTAG also accounts for the fact that the probability of a verb being a past participle, and not simply past, is greater when the following word is "by", as opposed to other prepositions. Certain cases of this sort may be soluble by making the collocational matrix distinguish classes of ambiguities—this question is being pursued. Approximately 1% of running text is tagged by IDIOMTAG (letter, G. N. Leech to Henry Kucera, June 7, 1985; letter, E. S. Atwell to Henry Kucera, June 20, 1985).

Marshall notes the possibility of consulting a complete three-dimensional matrix of collocational probabilities. Such a matrix would map ordered triples of tags into the relative probability of occurrence of each such triple. Marshall points out that such a table would be too large for its probable usefulness. The author has produced a table based upon more than 85% of the Brown Corpus; it occupies about 2 megabytes (uncompressed). Also, the mean number of examples per triple is very low, thus decreasing accuracy.

CLAWS has been applied to the entire LOB Corpus with an accuracy of "between 96% and 97%" (Booth (1985), p. 29). Without the idiom list, the algorithm was 94% accurate on a sample of 15,000 words (Marshall (1983)). Thus, the pre-processor tagging of 1% of all tokens resulted in a 3% change in accuracy; those particular assignments must therefore have had a substantial effect upon their context, resulting in changes of two other words for every one explicitly tagged.

But CLAWS is time- and storage-inefficient in the extreme, and in some cases a fallback algorithm is employed to prevent running out of memory, as was discovered by examining the Pascal program code. How often the fallback is employed is not known, nor is it known what effect its use has on overall accuracy.

Since CLAWS calculates the probability of every path, it operates in time and space proportional to the product of all the degrees of ambiguity of the words in the span. Thus, the time is exponential (and hence Non-Polynomial) in the span length. For the longest span in the Brown Corpus, of length 18, the number of paths examined would be 1,492,992.

2 The Linear-Time Algorithm (VOLSUNGA)

The algorithm described here depends on a similar empirically-derived transitional probability matrix to that of CLAWS, and has a similar definition of "optimal path". The tagset is larger than TAGGIT's, though smaller than CLAWS', containing 97 tags. The ultimate assignments of tags are much like those of CLAWS. However, it embodies several substantive changes. Those features that can be algorithmically defined have been used to the fullest extent. Other add-ons have been minimized. The major differences are outlined below.

First, the optimal path is defined to be the one whose component collocations multiply out to the highest probability. The more complex definition applied by CLAWS, using the sum of all paths at *each* node of the network, is not used.

Second, VOLSUNGA overcomes the Non-Polynomial complexity of CLAWS. Because of this change, it is never necessary to resort to a fallback algorithm, and the program is far smaller. Furthermore, testing the algorithm on extensive texts is not prohibitively costly.

Third, VOLSUNGA implements Relative Tag Probabilities (RTPs) in a more quantitative manner, based upon counts from the Brown Corpus. Where CLAWS scales probabilities by 1/2 for RTP < 0.1 (i.e., where less than 10% of the tokens for an ambiguous word are in the category in question), and by 1/8 for $p < 0.01$, VOLSUNGA uses the RTP value itself as a factor in the equation which defines probability.

Fourth, VOLSUNGA uses no tag triples and no idioms. Because of this, manually constructing special-case lists is not necessary. These methods are useful in certain cases, as the accuracy figures for CLAWS show; but the goal here was to measure the accuracy of a wholly algorithmic tagger on a standard corpus. Interestingly, if the introduction of idiom tagging were to make as much difference for VOLSUNGA as for CLAWS, we would have an accuracy of 99%. This would be an interesting extension. I believe that the reasons for VOLSUNGA's 96% accuracy without idiom tagging are (a) the change in definition of "optimal path", and (b) the increased precision of RTPs. The difference in tag-set size may also be a factor; but most of the difficult cases are major class differences, such as noun versus verb, rather than the fine distinction which the CLAWS tag-set adds, such as several subtypes of proper noun. Ongoing research with VOLSUNGA may shed more light on the interaction of these factors.

Last, the current version of VOLSUNGA is designed for use with a complete dictionary (as is the case when working with a known corpus). Thus, unknown words are handled in a rudimentary fashion. This problem has been repeatedly solved via affix analysis, as mentioned above, and is not of substantial interest here.

2.1 Choice of the Optimal Path

Since the number of paths over a span is an exponential function of the span length, it may not be obvious how one can guarantee finding the best path, without examining an exponential number of paths (namely all of them). The insight making fast discovery of the optimal path possible is the use of a **Dynamic Programming** solution (Dano (1975), Dreyfus and Law (1977)).

The two key ideas of Dynamic Programming have been characterized as "first, the recognition that a given 'whole problem' can be solved if the values of the best solutions of certain subproblems can be determined. . .; and secondly, the realization that if one starts at or near the end of the 'whole problem,' the subproblems are so simple as to have trivial solutions" (Dreyfus and Law (1977), p. 5). Dynamic Programming is closely related to the study of Graph Theory and of Network Optimization, and can lead to rapid solutions for otherwise intractable problems, given that those problems obey certain structural constraints. In this case, the constraints are indeed obeyed, and a linear-time solution is available.

Consider a span of length $n = 5$, with the words in the path denoted by v, w, x, y, z. Assume that v and z are the unambiguous bounding words, and that the other three words are each three ways ambiguous. Subscripts will index the various tags for each word: w_1 will denote the first tag in the set of possible tags for word w. Every path must contain v_1 and z_1, since v and z are unambiguous. Now consider the partial spans beginning at v, and ending (respectively) at each of the four remaining words. The partial span network ending at w contains exactly three paths. One of these must be a portion of the optimal path for the entire span. So we save all three: one path to each tag under w. The probability of each path is the value found in the collocation matrix entry for its tag-pair, namely $p(v,w_i)$ for i ranging from one to three.

Next, consider the three tags under word x. One of these tags must lie on the optimal path. Assume it is x_1. Under this assumption, we have a complete span of length 3, for x is unambiguous. Only one of the paths to x_1 can be optimal. Therefore we can disambiguate v. . . w. . . x_1 under this assumption, namely, as MAX $(p(v,w_i)* p(w_i,x_1))$ for all w_i.

Now, of course, the assumption that x_1 is on the optimal path is unacceptable. However, the key to VOLSUNGA is to notice that by making three such independent assumptions, namely for x_1, x_2, and x_3, we exhaust all possible optimal paths. Only a path which optimally leads to one of x's tags can be part of the optimal path. Thus, when examining the partial span network ending at word y, we need only consider three possibly optimal paths, namely those leading to x_1, x_2, and x_3, and how those three combine with the tags of y.

At most one of those three paths can lie along the optimal path to each tag of y; so we have 3^2, or 9,

The	AT		
man	NN	VB	
still	NN	VB	RB
saw	NN	VBD	
her	PPO	PP$	

Table 2: Sample Ambiguities

comparisons. But only three paths will survive, namely, the optimal path to each of the three tags under y. Each of those three is then considered as a potential path to z, and one is chosen.

This reduces the algorithm from exponential complexity to linear. The number of paths retained at any stage is the same as the degree of ambiguity at that stage; and this value is bounded by a very small value established by independent facts about the English lexicon. No faster order of speed is possible if each word is to be considered at all.

2.2. PROCESSING A SAMPLE SPAN

As an example, we will consider the process by which VOLSUNGA would tag "The man still saw her". We will omit a few ambiguities, reducing the number of paths to 24 for ease of exposition. The tags for each word are shown in Table 2. The notation is fairly mnemonic, but it is worth clarifying that PPO indicates an objective personal pronoun, and PP$ the possessive thereof, while VBD is a past-tense verb.

Examples of the various collocational probabilities are illustrated in Table 3 (VOLSUNGA does not actually consider any collocation truly impossible, so zeros are raised to a minimal non-zero value when loaded).

The product of 1*2*3*2*2*1 ambiguities gives 24 paths through this span. In this case, a simple process of choosing the best successor for each word in order would produce the correct tagging (AT NN RB VBD PPO). But of course this is often not the case.

Using VOLSUNGA's method we would first stack "the", with certainty for the tag AT (we will denote this by "p(the-AT) = CERTAIN)"). Next we stack "man", and look up the collocational probabilities of all tag pairs between the two words at the top of the stack. In this case they will be p(AT, NN) = 186, and p(AT, VB) = 1. We save the best (in this case only) path to each of man-NN and man-VB. It is sufficient to save a pointer to the tag of "the" which ends each of these paths,

	NN	PPO	PP$	RB	VB	VBD	
AT	186	0	0	8	1	8	9
NN	40	1	3	40	9	66	186
PPO	7	3	16	164	109	16	313
PP$	176	0	0	5	1	1	2
RB	5	3	16	71	118	152	128
VB	22	694	146	98	9	1	59
VBD	11	584	143	160	2	1	91

Table 3: Sample Collocational Probabilities

making backward-linked lists (which, in this case, converge).

Now we stack "still". For each of its tags (NN, VB, and RB), we choose either the NN or the VB tag of "man" as better. p(still-NN) is the best of:

p(man-NN) *p(NN,NN) = 186 *40 = 744
p(man-VB) *p(VB,NN) = 1 *22 = 22

Thus, the best path to still-NN is AT NN NN. Similarly, we find that the best path to still-RB is AT NN RB, and the best path to still-VB is AT NN RB. This shows the (realistically) overwhelming effect of an article on disambiguating an immediately following noun/verb ambiguity.

At this point, only the optimal path to each of the tags for "still" is saved. We then go on to match each of those paths with each of the tags for "saw", discovering the optimal paths to saw-NN and to saw-VB. The next iteration reveals the optimal paths to her-PPO and her-PP$, and the final one picks the optimal path to the period, which this example treats as unambiguous. Now we have the best path between two certain tags (AT and .), and can merely pop the stack, following pointers to optimal predecessors to disambiguate the sequence. The period becomes the start of the next span.

2.3 RELATIVE TAG PROBABILITIES

Initial testing of the algorithm used only transitional probability information. RTPs had no effect upon choosing an optimal path. For example, in deciding whether to consider the word "time" to be a noun or a verb, environments such as a preceding article or proper noun, or a following verb or pronoun, were the sole criteria. The fact that "time" is almost always a noun (1901 instances in the Brown Corpus) rather than a verb (16 instances) was not considered. Accuracy averaged 92–93%, with a peak of 93.7%.

There are clear examples for which the use of RTPs is important. One such case which arises in the Brown Corpus is "so that". "So" occurs 932 times as a qualifier (QL), 479 times as a subordinating conjunction (CS), and once as an interjection (UH). The standard tagging for "so that" is "CS CS", but this is an extremely low-frequency collocation, lower than the alternative "UH CS" (which is mainly limited to fiction). Barring strong contextual counter-evidence, "UH CS" is the preferred assignment if RTP information is not used. By weighing the RTPs for "so", however, the "UH" assignment can be avoided.

The LOB Corpus would (via idiom tagging) use "CS CS" in this case, employing a special "ditto tag" to indicate that two separate orthographic words constitute (at least for tagging purposes) a single syntactic word. Another example would be "so as to", tagged 'TO TO TO". Blackwell comments that "it was difficult to know where to draw the line in defining what constituted an idiom, and some such decisions seemed

to have been influenced by semantic factors. Nonetheless, IDIOMTAG had played a significant part in increasing the accuracy of the Tagging Suite [i.e., CLAWS]. . ." (Blackwell (1985), p. 7). It may be better to treat this class of "idioms" as lexical items which happen to contain blanks; but RTPs permit correct tagging in some of these cases.

The main difficulty in using RTPs is determining how heavily to weigh them relative to collocational information. At first, VOLSUNGA multiplied raw relative frequencies into the path probability calculations; but the ratios were so high in some cases as to totally swamp collocational data. Thus, normalization is required. The present solution is a simple one; all ratios over a fixed limit are truncated to that limit. Implementing RTPs increased accuracy by approximately 4%, to the range 95–97%, with a peak of 97.5% on one small sample. Thus, about half of the residual errors were eliminated. It is likely that tuning the normalization would improve this figure slightly more.

2.4 LEARNABILITY

VOLSUNGA was not designed with psychological reality as a goal, though it has some plausible characteristics. We will consider a few of these briefly. This section should not be interpreted as more than suggestive.

First, consider dictionary learning; the program currently assumes that a full dictionary is available. This assumption is nearly true for mature language users, but humans have little trouble even with novel lexical items, and generally speak of "context" when asked to describe how they figure out such words. As Ryder and Walker (1982) note, the use of structural analysis based on contextual clues allows speakers to compute syntactic structures even for a text such as *Jabberwocky*, where lexical information is clearly insufficient. The immediate syntactic context severely restricts the likely choices for the grammatical category of each neologism.

VOLSUNGA can perform much the same task via a minor modification, even if a suffix analysis fails. The most obvious solution is simply to assign all tags to the unknown word and find the optimal path through the containing span as usual. Since the algorithm is fast, this is not prohibitive. Better, one can assign only those tags with a non-minimal probability of being adjacent to the possible tags of neighboring words. Precisely calculating the mean number of tags remaining under this approach is left as a question for further research, but the number is certainly very low. About 3900 of the 9409 theoretically possible tag pairs occur in the Brown Corpus. Also, all tags marking closed classes (about two-thirds of all tags) may be eliminated from consideration.

Also, since VOLSUNGA operates from left to right, it can always decide upon an optimum partial result, and can predict a set of probable successors. For these reasons, it is largely robust against ungrammaticality. Shannon (1951) performed experiments of a similar sort, asking human subjects to predict the next character of a partially presented sentence. The accuracy of their predictions increased with the length of the sentence fragment presented.

The fact that VOLSUNGA requires a great deal of persistent memory for its dictionary, yet very little temporary space for processing, is appropriate. By contrast, the space requirements of CLAWS would overtax the short-term memory of any language user.

Another advantage of VOLSUNGA is that it requires little inherent linguistic knowledge. Probabilities may be acquired simply through counting instances of collocation. The results will increase in accuracy as more input text is seen. Previous algorithms, on the other hand, have included extensive manually generated lists of rules or exceptions.

An obvious difference between VOLSUNGA and humans is that VOLSUNGA makes no use whatsoever of semantic information. No account is taken of the high probability that in a text about carpentry, "saw" is more likely a noun than in other types of text. There may also be genre and topic-dependent influences upon the frequencies of various syntactic, and hence categorial, structures. Before such factors can be incorporated into VOLSUNGA, however, more complete dictionaries, including semantic information of at least a rudimentary kind, must be available.

3 Accuracy Analysis:

3.1 CALIBRATION

VOLSUNGA requires a tagged corpus upon which to base its tables of probabilities. The calculation of transitional probabilities is described by Marshall (1983). The entire Brown Corpus (modified by the expansion of contracted forms) was analyzed in order to produce the tables used in VOLSUNGA. A complete dictionary was therefore available when running the program on that same corpus.

Since the statistics comprising the dictionary and probability matrix used by the program were derived from the same corpus analyzed, the results may be considered optimal. On the other hand, the Corpus is comprehensive enough so that use of other input text is unlikely to introduce statistically significant changes in the program's performance. This is especially true because many of the unknown words would be (a) capitalized proper names, for which tag assignment is trivial modulo a small percentage at sentence boundaries, or (b) regular formations from existing words, which are readily identified by suffixes. Greene and Rubin (1971) note that their suffix list "consists mainly of Romance endings which are the source of continuing additions to the language" (p. 41).

A natural relationship exists between the size of a dictionary, and the percentage of words in an average text which it accounts for. A complete table showing the

#Types	Freq Limit	#Tokens	%Tokens
1	69,971	69,971	6.9
2	36,411	106,382	10.5
3	28,852	135,234	13.3
4	26,149	161,383	15.9
5	23,237	184,620	18.2
11	9,489	255,503	25.2
135	683	508,350	50.1
236	383	558,024	55.0
408	229	608,933	60.0
693	145	660,149	65.1
1,120	96	710,137	70.0
1,791	62	760,838	75.0
2,854	39	812,448	80.1
4,584	22	862,357	85.0
8,478	10	918,046	90.5
16,683	4	965,382	95.2
50,406	1	1,014,232	100.0

Table 4: Number of Tokens by Frequency

relationship appears in Kucera and Francis (1967) pp. 300–307. A few representative entries are shown in Table 4. The "#Types" column indicates how many vocabulary items occur at least "Freq Limit" times in the Corpus. The "#Tokens" column shows how many tokens are accounted for by those types, and the "%Tokens" column converts this number to a percentage. (See also pp. 358–362 in the same volume for several related graphs.)

3.2 OVERALL ACCURACY

Table 5 lists the accuracy for each genre from the Brown Corpus. The total token count differs from Table 4 due to inclusion of non-lexical tokens, such as punctuation. The figure shown deducts from the error count those particular instances in which the Corpus tag indicates by an affix that the word is part of a headline, title, etc. Since the syntax of such structures is often deviant, such errors are less significant. The difference this makes ranges from 0.09% (Genre L), up to 0.64% (Genre A), with an unweighted mean of 0.31%. Detailed breakdowns of the particular errors made for each genre exist in machine-readable form.

Genre	Size	% Accuracy
A: Press Reportage	99,165	96.36
B: Press Editorial	60,716	96.09
C: Press Reviews	39,832	96.12
D: Religion	38,631	96.01
E: Skills/Hobbies	81,659	95.34
F: Popular Lore	108,617	95.99
G: Belles Lettres	169,789	96.35
H: Miscellaneous	69,508	96.66
J: Learned	179,927	96.38
K: General Fiction	67,083	95.72
L: Mystery/Detective	56,090	95.47
M: Science Fiction	13,956	95.40
N: Adventure/Western	67,673	95.58
P: Romance/Love Story	68,337	95.54
R: Humor	20,990	95.55
Informative Prose Total	847,844	96.20
Imaginative Prose Total	294,129	95.57
Overall Total	1,141,973	96.04

Table 5: VOLSUNGA Tagging Accuracy

4 CONCLUSION

The high degree of lexical category ambiguity in languages such as English poses problems for parsing. Specifically, until the categories of individual words have been established, it is difficult to construct a unique and accurate syntactic structure. Therefore, a method for locally disambiguating lexical items has been developed.

Early efforts to solve this problem relied upon large libraries of manually chosen context frame rules. More recently, however, work on the LOB Corpus of British English led to a more systematic algorithm based upon combinatorial statistics. This algorithm operates entirely from left to right, and has no inherent limit upon the number of consecutive ambiguities which may be processed. Its authors report an accuracy of 96–97%.

However, CLAWS falls prey to other problems. First, the probabilistic system has been augmented in several ways, such as by pre-tagging of categorially troublesome "idioms" (this feature contributes 3% towards the total accuracy). Second, it was not based upon the most complete statistics available. Third, and perhaps most significant, it requires non-polynomially large time and space.

The algorithm developed here, called VOLSUNGA, addresses these problems. First, the various additions to CLAWS (i.e., beyond the use of two-place probabilities and RTPs) have been deleted. Second, the program has been calibrated by reference to 100% instead of 20% of the Brown Corpus, and has been applied to the entire Corpus for testing. This is a particularly important test because the Brown Corpus provides a long-established standard against which accuracy can be measured. Third, the algorithm has been completely redesigned so that it establishes the optimal tag assignments in linear time, as opposed to exponential.

Tests on the one million words of the Brown Corpus show an overall accuracy of approximately 96%, despite the non-use of auxiliary algorithms. Suggestions have been given for several possible modifications which might yield even higher accuracies.

The accuracy and speed of VOLSUNGA make it suitable for use in pre-processing natural language input to parsers and other language understanding systems. Its systematicity makes it suitable also for work in computational studies of language learning.

References

Beale, Andrew David. 1985 Grammatical Analysis by Computer of the Lancaster-Oslo/Bergen (LOB) Corpus of British English Texts. *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*. University of Chicago Press, Chicago, Illinois: 293–298.

Blackwell, Sue A. 1985 A Survey of Computer-Based English Language Research. *ICAME News* 9: 3–28. (Available from the Norwegian Computing Centre for the Humanities, Harald Harfagres gate 31, P.O. Box 53, N-5014 Bergen University, Norway.)

Booth, B. M. 1985 Revising CLAWS. *ICAME News* 9:29–35.

Dano, Sven. 1975 *Nonlinear and Dynamic Programming*. Springer-Verlag, New York.

Dreyfus, Stuart E. and Law, Averill, M. 1977 *The Art and Theory of Dynamic Programming*. Academic Press, New York.

Francis, W. Nelson and Kucera, Henry. 1979 *Manual of Information to Accompany A Standard Corpus of Present-Day Edited American English, for Use with Digital Computers* ("Revised and Amplified" edition). Department of Linguistics, Brown University, Providence, Rhode Island.

Francis, W. Nelson and Kucera, Henry. 1982 *Frequency Analysis of English Usage: Lexicon and Grammar*. Houghton-Mifflin Company, Boston, Massachusetts.

Greene, Barbara B. and Rubin, Gerald M. 1971 *Automated Grammatical Tagging of English*. Department of Linguistics, Brown University, Providence, Rhode Island.

Hirst, Graeme. 1983 *Semantic Interpretation Against Ambiguity*. Ph.D. dissertation., Department of Computer Science, Brown University, Providence, Rhode Island.

Klein, S. and Simmons, R. F. 1963 A Computational Approach to Grammatical Coding of English Words. *JACM* 10: 334–47.

Kucera, Henry and Francis, W. Nelson. 1967 *Computational Analysis of Present-Day American English*. Brown University Press, Providence, Rhode Island.

Leech, Geoffrey; Garside, Roger; and Atwell, Erik. 1983 The Automatic Grammatical Tagging of the LOB Corpus. *ICAME News* 7: 13–33.

Marshall, Ian. 1983 Choice of Grammatical Word-Class Without Global Syntactic Analysis: Tagging Words in the LOB Corpus. *Computers in the Humanities* 17: 139–150.

Ryder, Joan and Walker, Edward C. T. 1982 Two Mechanisms of Lexical Ambiguity. In Mehler, Jacques; Walker, Edward C.T.; and Garret, Merrill, Eds., *Perspectives on Mental Representation*. Lawrence Erlbaum Associates, Hillsdale, New Jersey: 134–149.

Shannon, Claude E. 1951 Prediction and Entropy of Printed English. *Bell System Technical Journal* 30: 50–64.