How much can part-of-speech tagging help parsing?

Mary Dalrymple
Centre for Linguistics and Philology
University of Oxford
Oxford OX1 2HG UK

mary.dalrymple@ling-phil.ox.ac.uk

June 19, 2005

Abstract

Folk wisdom holds that incorporating a part-of-speech tagger into a system that performs deep linguistic analysis will improve the speed and accuracy of the system. Previous studies of tagging have tested this belief by incorporating an existing tagger into a parsing system and observing the effect on the speed of the parser and accuracy of the results. However, not much work has been done to determine in a fine-grained manner exactly how much tagging can help to disambiguate or reduce ambiguity in parser output.

We take a new approach to this issue by examining the full parse-forest output of a large-scale LFG-based English grammar (Riezler et al., 2002) running on the XLE grammar development platform (Maxwell and Kaplan, 1993, 1996), and partitioning the parse outputs into equivalence classes based on the tag sequences for each parse. If we find a large number of tag-sequence equivalence classes for each sentence, we can conclude that different parses tend to be distinguished by their tags; a small number means that tagging would not help much in reducing ambiguity. In this way, we can determine how much tagging would help us in the best case, if we had the "perfect tagger" to give us the correct tag sequence for each sentence. We show that if a perfect tagger were available, a reduction in ambiguity of about 50% would be available. Somewhat surprisingly, about 30% of the sentences in the corpus that was examined would not be disambiguated, even by the perfect tagger, since all of the parses for these sentences shared the same tag sequence. Our study also helps to inform research on tagging by providing a targeted determination of exactly which tags can help the most in disambiguation.

1 Introduction

Systems that perform deep linguistic analysis generally operate by tokenizing the input string, performing morphological analysis, and handing off the tokenized, morphologically analyzed result as input to a syntactic parser. It is often argued that additional refinements in the input to the parser will improve performance: in particular, that including part-of-speech tagging as a preprocessing step will remove incorrect syntactic analyses from consideration by the parser, speeding up the parsing process and reducing ambiguity in the output of the parser. However, it is not clear exactly how much tagging can help to disambiguate or reduce ambiguity in parser output in naturally-occurring text. The two ends of the spectrum of possibilities are illustrated by two well-known ambiguous sentences in English.

The two parses of sentence (1) are associated with two different tag sequences:

(1) Time flies.

In the most natural reading for this sentence, *time* is a noun and *flies* is a verb, but there is another parse in which *time* is a verb and *flies* is a noun. If we knew that *time* was a noun (or that *flies* was a verb) in the current context, we could disambiguate the sentence completely.

Another type of ambiguity is illustrated by example (2):

(2) I watched the man with the telescope.

This sentence has multiple parses, but the ambiguity is not reflected in different part-of-speech tags for the words in the sentence; tagging will not help to disambiguate this example.

This study examines whether ambiguity in naturally-occurring English sentences is commonly reflected in different part-of-speech tags for different well-formed parses. If different parses tend to be associated with different tag sequences (the *Time flies* case), assigning the correct tag sequence as a preprocessing step before parsing will greatly reduce the number of parses that are produced. If different parses do not tend to be associated with different tag sequences (the *telescope* case), tagging will not help to reduce the output of the parser.

Previous research has attempted to answer this question by incorporating a tagger as a preprocessor to a parser, and seeing whether the accuracy of the parser improves as a result; some of this research will be discussed in the next section. This approach to the problem is potentially confusing, however, since any positive effects are inevitably obscured by the negative effect of tagging mistakes introduced by the tagger. No tagger is perfect: the current best-case scenario for taggers trained and tested on the Wall Street Journal corpus (Marcus et al., 1994a) is around 97.2% correct (Toutanova et al., 2003). Various ways of dealing with the problem of mistags have been suggested; for example, Copperman and Segond (1996) suggest that tagging should be used not as a preprocessor to a parser, but to eliminate unlikely parts of speech for a particular domain or genre from a general-purpose lexicon. This may be of help, but does not address the general question of how much tagging as a preprocessing step could in principle help in disambiguation.

An additional problem that has plagued several previous studies is incompatibility between the tags assigned by the tagger and the preterminal symbols used by the grammar. Requiring an additional mapping between the tagset for the tagger and the preterminal categories employed in syntactic analysis introduces additional errors and further obscures the results. Studies that train the tagger on the output of the parser do not suffer from this problem (Prins and van Noord, 2001, 2003), but must still contend with the harmful effect of mistags on the result.

We take a new approach to this issue by measuring the disambiguating effect of a "perfect tagger", a hypothetical tagger which never makes a tagging mistake. This allows us to evaluate the effect of tagging on disambiguation while abstracting away from the problems that are associated with any particular tagger. We do this by examining the full parse-forest output of a parser, and partitioning the output parses into equivalence classes based on the tag sequences for each parse. Roughly speaking, if we find a large number of tag-sequence equivalence classes for each sentence, we can conclude that different parses tend to be distinguished by their tags; a small number means that tagging would probably not help much in reducing ambiguity. In this way, we can determine how much tagging would help us in the best case, if we had the perfect tagger to give us the correct tag sequence for each sentence.

Given the tag sequences for all successful parses of the input, we can also answer more fine-grained questions about the utility of tagging: for example, which parts

of speech play the biggest role in creating multiple tag sequence equivalence classes for a sentence. We return to this question in Section 5 below.

It is important to note that the current study addresses only the question of the utility of tagging as it relates to ambiguity reduction. It does not answer the question of the effect of a tagger on *speed*; that is, it cannot determine whether a parser will perform faster on tagged input. This is because the data used in the study is a *parse forest* – a packed representation of all full, well-formed parses – and not a chart. Neither incomplete edges nor edges that fail to contribute to a full, well-formed parse are present for consideration in the data being examined. Since there is no way to determine how much work was done on tag sequences which do not ultimately contribute to a full and complete parse, there is no way for a study such as this one to address questions of increased efficiency or speed when parser input is pretagged. This question has been addressed in other studies, which conclude that tagging in a preprocessing step does in fact speed up the parsing process.

2 Previous work

One of the first studies to address the question of whether tagging helps in parsing was reported by Pulman (1992). In this study, a tagger was trained on the LOB corpus and used as a preprocessor to the Core Language Engine (Alshawi, 1992). This resulted in a loss in accuracy in parsing, though it did increase parsing speed. Accuracy was regained by the use of a multiple tagger, a tagger that returns more than one tag for each word. However, to regain the original level of accuracy, each word had to be assigned a large enough number of tags that most of the speed gain obtained from pretagging the input was lost. Interestingly, this result goes against the findings of Charniak et al. (1996), whose work indicated that a multiple tagger does not in fact significantly increase accuracy when used as a preprocessor to a probabilistic context-free phrase structure grammar relative to a single tagger, which assigns only one tag per word.

Subsequently, Wauschkuhn (1995) reported on a study in which two German corpora were studied; one was hand-tagged, and the other was statistically tagged, with an error rate of 3.5% to 4%. Both of these corpora were parsed twice: once with tags, and once without tags but with a morphological analyzer. There was no gold standard for either corpus, so the metric of success that was used is the number of sentences receiving a single parse in each case. This study suffered from several problems. First, the tags assigned by the morphological analyzer were not the same as the tags used for hand-tagging, which made comparison of the results difficult. Second, we do not expect tagging alone to completely disambiguate a sentence; a sentence may be structurally ambiguous, even with the same tags, so using a metric which defines success as obtaining a single parse does not seem appropriate. Third, the grammar used in the test seems to be quite small, perhaps too small for a fair trial: the majority of sentences got either zero or one parse for both the tagged and untagged corpus.

A subsequent study was conducted by Voutilainen (1998) on the basis of a system which uses a finite-state syntactic disambiguator to discard impossible syntactic analyses. Voutilainen added a morphological disambiguator to discard impossible tags before syntactic analysis is performed. The conclusion of the study was that tagging helps to reduce ambiguity, but increases the number of sentences with no parse. As in Wauschkuhn's study, no gold standard was available to determine whether the correct parse was among those parses that were discarded, which makes it hard to determine the benefit of the addition of a tagger to the system.

More recently, Prins and van Noord (2001, 2003) addressed this question by

adding a morphological disambiguator trained on the output of the parser. This method solves one of the problems that plagues other approaches, that of incompatibility between the tagset used by the tagger and the preterminal categories of the grammar. Prins and van Noord also were able to evaluate their results against a gold standard, showing whether the correct parse is present in the output when a tagger is used. Like the work reported by Pulman (1992), Prins and van Noord concluded that tagging does in fact help to reduce ambiguity if a multiple tagger is used. Prins and van Noord also show conclusively that tagging as a preprocessing step can increase parsing efficiency, with a twentyfold speedup for the Alpino system that they tested. Nevertheless, their approach, like most other approaches, fails to distinguish between the beneficial effects of tagging and the harmful effects of tagger errors.

A study which is close in some respects to the experiment reported here was conducted by Kaplan and King (2003), using the XLE parsing platform and PARGRAM English grammar, described below. Kaplan and King attempted to simulate the effect of a "perfect tagger" by using the preterminal category sequence from the Penn Treebank to tag the input string in parsing sentences in the Wall Street Journal corpus. However, the Penn Treebank preterminal categories and the preterminal categories of the PARGRAM English grammar are not compatible, which necessitated the introduction of a mapping function to mediate between the two tagsets. Kaplan and King concluded that parsing with input annotated with tags from the Penn Treebank speeds up parsing, but decreases parsing accuracy and coverage. Incompatibility between the Penn Treebank preterminals and the PARGRAM preterminals, with "tagging" errors introduced by errors in the mapping function, was a major source of difficulties for their approach.

Another closely related study was carried out by Toutanova et al. (2002), who investigated several techniques for disambiguation in parsing sentences from the Redwoods HPSG treebank (Oepen et al., 2002). One of the disambiguation techniques they investigated was adding a tagger trained on the gold standard treebank as a preprocessing step. They compared these results with the effects of a "perfect tagger" which, as in the Kaplan and King experiment, assigned the tags that appear as preterminals in the Redwoods treebank gold standard. They reported results which are very similar to the findings of the current study; we will return to a discussion of their work in Section 4 below.

3 The current study: Methodology

To answer the question of how much tagging can help parsing, we used a broad-coverage grammar of English to parse a large corpus. We then extracted the preterminal sequence for each parse – these are the "tags" that would be assigned by the perfect tagger – and grouped the preterminal sequences into equivalence classes. This allows us to answer several questions. First, by counting the number of tag sequence equivalence classes for each sentence in the corpus, we can determine the correlation between syntactic ambiguity and number of tag sequence equivalence classes: are most cases of ambiguity like the *Time flies* case, or like the *telescope* case? Second, we can determine which tags tend to give rise to different tag sequence equivalence classes; these are the tags that can help the most in disambiguation by tagging.

3.1 The grammar and parser

Our study uses the English grammar developed at the Palo Alto Research Center within the PARGRAM project, a large-scale multi-site LFG grammar development

project (Butt et al., 2002). The PARGRAM project encompasses large-scale grammars of English, French, German, Japanese, Norwegian, and Danish, with smaller grammars of Korean and Urdu also under development. The version of the PARGRAM English grammar used in this experiment is the result of about 9 person years of development, using the XLE grammar development and parsing platform for Lexical Functional Grammar.

Analysis of a string using the PARGRAM English grammar and the XLE parsing platform begins with the following steps:

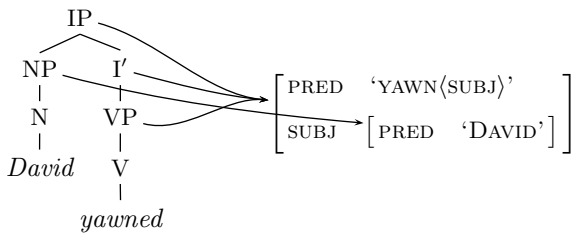
- Tokenization by finite-state transducer
- Finite-state morphological analysis, including part-of-speech information
- Guesser for forms not recognized by morphological analysis

The resulting input to syntactic rules is a chart consisting of all well-formed morphological analyses of all well-formed tokenizations. The version of the PARGRAM English grammar used in this study comprises 314 rules (left-hand side categories) with regular-expression right-hand sides. Lexical entries for most nouns and adjectives are constructed on the fly on the basis of the assigned syntactic category. The verb lexicon contains 9,652 stems and 23,525 subcategorization frame entries (Riezler et al., 2002).

3.2 Grammar output

The output of the XLE is a packed representation of well-formed pairs consisting of a *c-structure* or phrase structure tree and an *f-structure* or attribute-value structure:

(3)



The c-structure is a phrase structure tree representing surface phrasal relations and groupings; it appears on the left-hand side in (3). The f-structure is an attribute-value structure representing abstract functional syntactic relations like subject and object (Kaplan and Bresnan, 1982; Dalrymple, 2001); it appears on the right-hand side in (3). The mapping relating the two structures is represented by arrows from nodes of the c-structure to subparts of the f-structure. For the current study, f-structure information is not relevant, and we can therefore ignore the f-structure and think of the output simply as a packed parse forest.

Riezler et al. (2002) show that the coverage of the PARGRAM English grammar used in this study is very high, and the output is of very high quality. In a test of Section 23 of the Wall Street Journal corpus, 100% of the sentences received an analysis, though some analyses consisted of a set of well-formed fragments; 74.7% of sentences received a full (nonfragmentary) syntactic analysis. From Section 23, 700 sentences were randomly selected and a gold standard was hand-constructed (the PARC 700 Dependency Bank: King et al., 2003). These 700 sentences were then parsed using the English grammar, and the parse with the highest f-score[1] was

[1]F-score is an overall score representing a combination of precision and recall (van Rijsbergen, 1979):

$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

chosen; these parses had an average f-score of 84.1% relative to the gold standard. The average f-score for a randomly-selected parse relative to the gold standard was 78.6%, still quite high.

3.3 PARGRAM tags

Table 1 contains the 115 preterminal node labels used in the PARGRAM English grammar, referred to as "tags" in the following. These have been divided into the following groups:

(4) a. Verbal, including auxiliaries

b. Nominal

c. Adverbial

d. Prepositional

e. Punctuation

f. Other

With 115 tags, the PARGRAM tagset is more fine-grained than, for example, the Penn tagset, which has 48 tags (Marcus et al., 1994b). The relatively large size of the PARGRAM tagset does not in itself pose a problem for tagger training; as shown by Elworthy (1995), larger tagsets are not necessarily less accurate than smaller ones, at least for languages like English. Indeed, much larger tagsets have been used in tagger training: Toutanova et al. (2002) report reasonable results in training a tagger for English with the tagset for the HPSG Redwoods treebank. This tagset incorporates very detailed syntactic and semantic information, and assumes a tagset of 8,000 lexical tags.

Some of the PARGRAM English tag distinctions reflect subdivisions of standard phrase structure categories which are used in different syntactic contexts, such as the various verbal tags: V[fin] for finite verbs, V[perf] for perfect participles, V[prog] for progressive participles, and so on. These morphologically-encoded differences should in principle be possible for a standard tagger to discriminate. Other tags directly reflect structural syntactic ambiguities, however: for example, the word *and* can be tagged either as CONJ (used in non-nominal conjunction) or as CONJnp (used in conjunction of noun phrases). Such distinctions may well be very difficult for a tagger to make.

On the other hand, the PARGRAM tagset is less fine-grained than tagsets such as those used by Toutanova et al. (2002) in their experiments with the HPSG Redwoods treebank or by Bangalore and Joshi (1999) and Clark and Curran (2004) for supertagging in Combinatory Categorial Grammar and Tree Adjoining Grammar. These tagsets contain much more detailed syntactic information than the PARGRAM tagset about the syntactic environment in which a word can appear.

We hypothesize that when using a more fine-grained tagset which reflects syntactic ambiguities, different syntactic analyses of a string will tend to be reflected in different tags for the words in the string; conversely, if a coarser-grained tagset is used, syntactic ambiguity will tend not to be reflected in tags. If this is true, this study represents a middle ground for evaluation of syntactic disambiguation by tagging, since the PARGRAM tagset contains more syntactic information relevant for disambiguation than the Penn tagset, but less than is available from supertags.

Table 1: PARGRAM English grammar preterminals

Verbal tags	AUX[fut,fin]	AUX[modal,fin]	
	AUX[pass,base]	AUX[pass,fin]	
	AUX[pass,perf]	AUX[pass,prog]	
	AUX[perf,base]	AUX[perf,fin]	
	AUX[perf,prog]	AUX[prog,base]	
	AUX[prog,fin]	AUX[prog,perf]	
	AUXdo[base]	AUXdo[fin]	
	AUXsubj[fin]	V[base]	
	V[fin]	V[pass]	
	V[perf]	V[prog]	
	Vcop[base]	Vcop[fin]	
	Vcop[perf]	Vcop[prog]	
Nominal tags	N	NAME	
	NP[int]	NP[rel]	
	Ndate	Npart	
	PRON	PRON[int]	
	PRON[rel]	PRONemph	
	PRONfree	PRONheadless	
	PRONposs	PRONpp	
	DAY	HOUR	
	MN	MONTH	
	TITLE		
Adverbial tags	ADV	ADVadj	
	ADVadj[post]	ADVadj[pre]	
	ADVcomp	ADVcompmod	
	ADVcoord	ADVdate[any]	
	ADVdate[fin]	ADVdet	
	ADVfoc	ADVinf	
	ADVint	ADVnum	
	ADVpmod	ADVtime	
Prepositional tags	P		
	Padj		
	Pnum		
	Ppart		
	PP		
	PP[int]		
	PP[rel]		
	PPcl		
Punctuation/non-word tags	CCOLON	CDASH	COLON
	COMMA	CSEMI-COLON	DASH
	ELLIPSIS	HYPHEN	INT-MARK
	L-CRL	L-PRN	L-QT
	LD-QT	PERIOD	R-CRL
	R-PRN	R-QT	RD-QT
	SEMI-COLON	U-QT	TOKEN
Other tags	A	Adate	Aquant
	CONJ	CONJcomp	CONJnp
	CONJsub	C[inf]	C[int]
	C[pred]	C[that]	D
	D[int]	Dcomp	INITIAL
	LETTER	NEG[con]	NEG[full]
	NUMBER	PA	PART
	PARTinf	POSS	PRE-N
	PRE-V	PRECONJ	PREDET
	PREint		

3.4 The data

The current study examined sentences from two sections of the Wall Street Journal corpus (Marcus et al., 1994a). The first dataset consists of all of the sentences from Section 13; this was chosen as a large sample, but with no gold standard. The second dataset consists of 100 sentences from Section 23 in which the correct parse was marked; this was chosen as a small gold-standard sample for comparison with the larger sample. These sentences were parsed using the XLE parsing platform and the PARGRAM English grammar.

The results for Section 13 were:

(5)

Total sentences in Section 13	2481
Full and fragmentary parses	2429
Nonfragmentary parses with extracted tag sequences	2105

Of the 2481 sentences in Section 13, 2429 sentences obtained a complete (but possibly fragmentary) parse in the time allowed. Skimming was not used; in skimming mode, the parser may return a result containing only a subset of the parses licensed by the grammar (Riezler et al., 2002). Of these, tags were extracted from 2105 sentences. Tags were not extracted from sentences which obtained only a fragmentary parse, nor (because of resource limitations) from the 5 sentences with more than 80,000 full parses. For the sentences in Section 13, the correct or optimal parse was not marked, so there is no gold standard for evaluation of Section 13.

The 100 sentences chosen from Section 23 constitute a much smaller corpus. In creating this corpus, the sentences in Section 23 were parsed in sequence; sentences with a full and correct parse were banked, with the correct parse marked. This process continued until a 100-sentence treebank corpus had been produced. These 100 sentences all received a nonfragmentary parse, and tag sequences were extracted for each sentence.

(6)

Total sentences selected from Section 23	100
Nonfragmentary parses with extracted tag sequences, correct parse marked	100

The 100-sentence Section 23 corpus tends to contain slightly shorter and less ambiguous sentences:

(7)

	Num. sentences	Mean sentence length (words)	Mean num. parses	Median num. parses
Section 13	2105	21.6	429	12
Section 23	100	18.2	63	8

Comparison of the two corpora is therefore difficult; the bulk of the study described below is conducted on the basis of the larger Section 13 corpus.

From the packed parse forest for each sentence in both corpora, a file was produced containing the tag sequence (the sequence of preterminal categories) for each parse. The contents of an example tagfile for the sentence *Hess declined to comment* is given in (8). The sentence has three parses. In the first parse, *to* has category P (preposition) and *comment* has category N (*decline* is intransitive in this case, and *to comment* is a prepositional phrase). In the second and third parses, *to* has category PARTinf (infinitival particle) and *comment* has category V[base] (one parse is the expected one, where *to comment* is an infinitival clause and an argument of *declined*; in the other parse, *declined* is intransitive and *to comment* is an infinitival purpose modifier).

(8) NAME:Hess V[fin]:declined P:to N:comment PERIOD:.
NAME:Hess V[fin]:declined PARTinf:to V[base]:comment PERIOD:.
NAME:Hess V[fin]:declined PARTinf:to V[base]:comment PERIOD:.

Next, the tag sequences were grouped into equivalence classes. The tag sequences in (8) would be grouped into two equivalence classes, with the first parse in one class and the second and third parses in the other class. Example statistics for sentences 1000-1010 are given in (9).

(9)

Sentence	Number of words	Number of parses	Number of tag sequence equivalence classes
wsjS1000.tags	26	2	1
wsjS1001.tags	9	1	1
wsjS1002.tags	15	15	6
wsjS1003.tags	27	49	18
wsjS1004.tags	28	640	8
wsjS1005.tags	31	320	2
wsjS1007.tags	33	128	2
wsjS1008.tags	14	6	3
wsjS1009.tags	24	4	1
wsjS1010.tags	38	660	6

The results reported below are based on these equivalence class groupings.

4 Analysis

620 (29.47%) of the sentences in Section 13 had parses whose tag sequences all fell into one equivalence class; these sentences had between 1 and 320 parses. One sentence had parses falling into 600 equivalence classes (20 words, 7336 parses); all the others had parses falling into 234 or fewer equivalence classes.

(10)

Number of tag sequence equivalence classes	Number of sentences	Number of parses	Cumulative percentage
1	620	1–320	29.47%
2	526	2–4608	54.47%
3	102	3–4758	59.32%
4-10	591	4–13584	87.40%
11-20	143	12–30576	94.20%
21-50	84	48–62464	98.19%
51-100	21	176–82704	99.19%
101-234	17	448– 75152	99.99%
600	1	7336	100.00%
total	2105	903765	

In answering the question of whether tagging is useful in disambiguation, the most relevant statistic is the proportion of sentences whose tag sequences all fall into one equivalence class: tagging would not help to disambiguate 29.47% of the sentences in this corpus, while it would help with the remaining 70.53%.

The relation between degree of ambiguity and number of tag equivalence classes is given in (11):

(11)

Number of tag sequence equiv. classes	Number of parses	Mean number of parses	Median number of parses
1	1–320	7.18	2
2	2–4608	46.04	8
3	3–4758	103.96	12
4-10	4–13584	198.38	36
11-20	12–30576	943.30	179
21-50	48–62464	2154.96	584
51-100	176–82704	8672.48	2484
101-234	448– 75152	12448.82	7496
600	7336	7336	7336

Somewhat surprisingly, the number of tag equivalence classes does not correlate well with either the length of the input string or with the number of parses of the sentence. Only 5% of the variation in tag-sequence classes is accounted for by sentence length, and only 16% by the number of parses:

(12)

	Sentence length	Num. parses
Num. tag sequence equiv. classes	r=.2260	r=.3967
	r^2=.0510	r^2=.1574

This means that it is difficult to tell in advance whether tagging will be helpful: tagging only the longer sentences, for example, is not guaranteed to be the best strategy.

4.1 Ambiguity reduction from tagging

Our problem is to estimate the degree of ambiguity reduction that could be obtained if the correct tag sequence is specified for each sentence by the "perfect tagger".[2] Specifying a tag sequence amounts to choosing a particular tag sequence equivalence class for a sentence; when we choose an equivalence class as the correct one, we rule out the parses in other equivalence classes. Therefore, the size of the equivalence class containing the correct parse represents the degree to which ambiguity can be reduced by tagging.

For the sentences in Section 13, the correct parse is not marked, and so we do not know which equivalence class contains the right parse. One way to guess how much ambiguity reduction is available by tagging is to compute the average size of the tag sequence equivalence classes. For Section 13, the tag sequence equivalence classes are distributed as follows:

(13)

Mean size of equivalence classes, Section 13:	14.31%
Median:	4.16%

The equivalence classes have an average size of 14.31%, with a median size of 4.16%, meaning that we can rule out 85-95% of the parses by randomly choosing a tag sequence for an input string.

Again, however, we do not know for Section 13 which tag sequence equivalence class contains the correct parse. It may well be that the correct parse is usually contained in the *largest* tag sequence equivalence class (since most parses are contained in that class). This is the worst case for disambiguation by the tagger, since the smallest number of parses will be ruled out if the largest tag sequence equivalence class turns out to be the correct one. For Section 13, the average size of the *largest* tag sequence equivalence class is:

[2] I am grateful to Ron Kaplan for discussion of these issues.

(14) Mean size of *largest* equivalence class, Section 13: 55.55%
Median: 50.00%

If the largest tag sequence is taken to be the correct one, we can expect to rule out 45-50% of the potential parses for the sentence.

To decide which figure better reflects how much disambiguation can be expected from tagging, our corpus of 100 parses from Section 23 of the Wall Street Journal is relevant, since in this corpus the correct parse for each of the sentences has been hand-selected. For this corpus, the results are:

(15) Mean size of *correct* equivalence class, Section 23: 54.83%
Median: 50.00%

This result is surprisingly close to the result obtained by always choosing the largest equivalence class from the Section 13 corpus. If these data are representative, we can expect to rule out 45-50% of the potential parses for a sentence by choosing the correct tag sequence for the sentence.

Furthermore, this result accords well with results obtained by Toutanova et al. (2002) in their work on disambiguation with the HPSG LinGO grammar. As outlined above, Toutanova et al. investigated several disambiguation techniques in parsing sentences from the HPSG Redwoods treebank, including tagging the input in a preprocessing step. Since the Redwood corpus represents a very large gold standard corpus, they can identify the correct tag sequence for each sentence, and determine how much disambiguation the "perfect tagger" would provide. They report a correct tag sequence equivalence class size of 54.59%, very close to the results fond in the current study. This convergence of results is all the more surprising, given the very different granularity of the tagsets; the PARGRAM English tagset contains 115 tags, while the HPSG Redwoods tagset contains 8,000 tags.

One difference between the two studies is that Toutanova et al. report results only for sentences that have more than one parse, and for which disambiguation is therefore an issue. Because we do not know before parsing a sentence whether it is ambiguous or not, and because we are investigating the utility of a tagger as a preprocessor and not as a means of selecting the correct parse from the output of a parser, we have included all sentences in our corpus in the results we report above, not just the sentences that have more than one parse.

Our results are, of course, dependent on the grammar that was used in the experiment. Like the HPSG LinGO grammar, the PARGRAM grammar produces linguistically rich, detailed analyses in which subcategorization and other grammatical requirements must be satisfied; analyses which violate these requirements are ruled out by the grammar and do not appear in the output of the parser. It may be that tagging would play a greater role in ambiguity reduction for looser, less constrained grammars which do not encode or enforce such grammatical requirements.

5 Ambiguous words

Identifying the tags that are most often involved in cases of ambiguity provides useful information for developers of taggers as well as for grammar writers to tune large-scale grammars and reduce unnecessary ambiguity. In the following, we examine which tags are most often involved in distinguishing between different tag sequence equivalence classes.

The correct parse for the sentences in Section 13 is not marked, so we cannot compare the correct parse to the rest of the parses to determine which words are most often tagged incorrectly. However, we know from research conducted by Riezler et al. (2002) that the average quality of the analyses produced by the PARGRAM

English grammar is quite high, with a randomly-selected parse getting an average f-score of 78.6%. We can, then, perform the following experiment. We first pick an arbitrary parse as the standard to evaluate against. We then record the number of instances of disagreement relative to this arbitrarily-selected parse, where disagreement is defined as an instance of tag mismatch between a parse and the standard. Here we consider only parses with the same tokenization as the arbitrarily-chosen standard, discarding parses with different tokenizations.

Table 2 contains the confusion matrix for disagreements representing at least 1% of the total disagreements in the data. An explanation of the preterminal symbols referenced in Table 2 is given in (16):

(16)

A	adjective
ADV	adverb
CONJ	conjunction
CONJnp	conjunction for noun phrases
CONJsub	subordinating conjunction
C[that]	*that* as complementizer
D	determiner
N	noun
NAME	proper name
P	preposition
PA	*a* in constructions like *5 times a day*
PARTinf	*to* as infinitival marker
PRON	pronoun
V[base]	base (citation) form of verb
V[fin]	finite verb
V[pass]	passive participle form of verb
V[prog]	progressive form of verb

Three entries in the table are worthy of note.

Tag disagreement between the category A (adjective) and N (noun) accounted for 29.63% of cases of disagreement between the arbitrarily-selected standard and the other parses (summing together the 21.86% of cases where the arbitrarily-chosen standard had category A and the other parses had N, and the 7.77% of cases where the standard had N and the other parses had A). The reason for this is that there are many words that can be used either as an adjective or as a noun, and it is difficult to allow only an adjective + noun parse for these cases and disallow a noun-noun compound parse. For example, the most obvious parse for a phrase like *green box* is the one where *green* is an adjective, but in order to parse examples like *Green is my favorite color*, *green* is also listed in the lexicon as a noun; thus, *green box* gets a noun-noun compound parse like the parse for *music box*.

Tag disagreement between the category CONJ (conjunction) and CONJnp (a special category for noun phrase conjunction) accounted for 7.21% (5.10% + 2.11%) of cases of disagreement. On the PARGRAM English grammar analysis of coordination, the lexical category of the conjunction reflects structural ambiguity arising from differences in scope of coordination. The grammar writer could alternatively have made a different choice: to use the same preterminal category for conjunctions used within noun phrases as in coordination more generally. If that had been done, the number of parses would have remained the same (the same structural ambiguities for scope of coordination would have been available), but these would not have been reflected in different tag sequences for the different possibilities: given such a grammar, choosing a tag sequence would not disambiguate between different coordination possibilities.

Tag disagreement between the category V[prog] and the category N accounted for 8.31% (1.83%+6.48%) of cases of disagreement. This is because present par-

Table 2: Confusion matrix for tag mismatches >1%, Section 13

	A	ADV	C[that]	CONJ	CONJnp	N	NAME	P	PA	PARTinf	V[base]	V[fin]	V[pass]	V[prog]
A		1.45				21.86								3.61
ADV	2.11													
CONJ					5.10									
CONJnp				2.11										
CONJsub								3.56						
D									2.23					
N	7.77						1.67				1.73	4.56		6.48
P										1.67				
PRON			3.17											
V[fin]						4.68							2.75	
V[prog]	1.20					1.83								

ticiple forms like *swimming* can be analyzed either as progressive verbs (*He is swimming*) or as gerunds (*Swimming is fun*). Ambiguity can arise in a variety of situations: the most pernicious is in seemingly simple cases like *He is swimming*, which has besides the obvious present progressive reading, a reading where *is* is analyzed as a copula and *swimming* is a gerund, with a meaning something like *he is (the concept of) swimming*. Again, it is difficult to rule out such examples in a non-ad-hoc manner.

6 Conclusion

Examining the output of the PARGRAM English grammar allows us to assess the effect of incorporating a tagger into a large-scale processing system in the best case, abstracting away from errors that would inevitably be introduced by even the best tagger currently available. We have shown that a perfect tagger would reduce ambiguity by about 50%. Somewhat surprisingly, about 30% of the sentences in the corpus that was examined would not be disambiguated, even by the perfect tagger, since all of the parses for these sentences shared the same tag sequence. For at least some of the difficult cases, in particular ambiguity corresponding to scope of coordination, it is not at all clear whether a tagger could be expected to do better than a parser in any case.

7 Acknowledgments

Thanks to Steve Clark, Ken Kahn, David Kahn, Tracy King, Steve Pulman, audiences at University of Brighton, University of Sheffield, and University of Oxford, and especially to Ron Kaplan for very helpful discussion, and to Matty Dalrymple for help with Excel.

References

Alshawi, Hiyan (editor). 1992. *The Core Language Engine*. Cambridge, MA: The MIT Press.

Bangalore, Srinivas and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics* 25(2), pp. 237–266.

Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*.

Charniak, Eugene, Glenn Carroll, John Adcock, Anthony Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael Littman, and John McCann. 1996. Taggers for parsers. *Artificial Intelligence* 85(1–2).

Clark, Stephen and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING2004),* Geneva.

Copperman, Max and Frédérique Segond. 1996. Computational grammars and ambiguity: the bare bones of the situation. In Miriam Butt and Tracy Holloway King (editors), *The Proceedings of the LFG '96 Conference*. Rank Xerox, Grenoble. URL `http://csli-publications.stanford.edu/LFG/1/lfg1.html`.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. New York, NY: Academic Press.

Dalrymple, Mary, Ronald M. Kaplan, John T. Maxwell, III, and Annie Zaenen (editors). 1995. *Formal Issues in Lexical-Functional Grammar*. Stanford, CA: CSLI Publications.

Elworthy, David. 1995. Tagset design and inflected languages. In *From Texts to Tags: Issues in Multilingual Text Analysis*. ACL SIGDAT Workshop, Dublin.

Kaplan, Ronald M. and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan (editor), *The Mental Representation of Grammatical Relations*, pp. 173–281. Cambridge, MA: The MIT Press. Reprinted in Dalrymple et al. (1995, pp. 29–130).

Kaplan, Ronald M. and Tracy Holloway King. 2003. Low-level markup and large-scale LFG grammar processing. In Miriam Butt and Tracy Holloway King (editors), *On-line Proceedings of the LFG2003 Conference*. URL `http://csli-publications.stanford.edu/LFG/8/lfg3.html`.

King, Tracy H., Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*. Budapest. Held at the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03).

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994a. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994b. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19(2), pp. 313–330.

Maxwell, John T., III and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics* 19(4), pp. 571–590.

Maxwell, John T., III and Ronald M. Kaplan. 1996. An efficient parser for LFG. In Miriam Butt and Tracy Holloway King (editors), *On-line Proceedings of the LFG96 Conference*. URL `http://csli-publications.stanford.edu/LFG/1/lfg1.html`.

Oepen, Stephan, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING2002),* Taipei.

Prins, Robbert and Gertjan van Noord. 2001. Unsupervised POS-tagging improves parsing accuracy and parsing efficiency. In *Proceedings of 7th International Workshop on Parsing Technologies*. Beijing.

Prins, Robbert and Gertjan van Noord. 2003. Reinforcing parser preferences through tagging. *Traitement automatique des langues* To appear, special issue on Evolutions in Parsing.

Pulman, Steve. 1992. Using tagging to improve analysis efficiency. In Henry Thompson (editor), *SALT/ELSNET Workshop on Sub-Language Grammar and Lexicon Acquisition for Speech and Natural Language Processing*. Record of verbal presentation.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the ACL,* Philadelphia.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.

Toutanova, Kristina, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pp. 253–263.

van Rijsbergen, C. J. 1979. *Information Retrieval*. London: Butterworth-Hienemann, second edition.

Voutilainen, Atro. 1998. Does tagging help parsing? A case study on finite-state parsing. In *Finite-State Methods in Natural Language Processing*.

Wauschkuhn, Oliver. 1995. The influence of tagging on the results of partial parsing in German corpora. In *4th International Workshop in Parsing Technologies*. Prague.