The Detection of Inconsistency in Manually Tagged Text

Hans van Halteren
Dept. of Language and Speech
University of Nijmegen
P.O. Box 9103
6500 HD Nijmegen
The Netherlands
hvh@let.kun.nl

Abstract

This paper proposes a method to detect the presence of inconsistency in a given manually tagged corpus. The method consist of generating an automatic tagger on the basis of the corpus and then comparing the tagger's output with the original tagging. It is tested using the written texts from the BNC sampler and a WPDV-based tagger generator, and shown to be both an efficient method to derive a qualitative evaluation of consistency and a useful first step towards correction.

1 Introduction

Wordclass tagged corpora are a very popular resource for both language engineers and linguists. If these corpora are used for inspiration and exemplification, size may be more important than quality and a fully automatically tagged corpus can suffice. For other uses, quality is of much higher importance, and here there will generally be a preference for manually corrected corpora, even though they may be smaller. However, manual correction means human involvement, and that again means a much higher potential for inconsistency (cf. e.g. Marcus et al. (1993); Baker (1997)).

Before we go and base our NLP systems or linguistic theories on the wordclass tags found in a tagged corpus, then, it would certainly be a good idea to evaluate whether those tags have indeed been assigned appropriately, and, if not, possibly correct the situation. This means that we have to inspect (part of) the corpus and decide whether the tags are consistent with the tagging manual or, if the tagging manual is not clear on the subject, whether the tags have at least been applied consistently throughout the corpus. In this paper we show, by way of an experiment, how this task can be made more efficient with the help of software already in general use in wordclass tagging circles, viz. tagger generators.

The tagged corpus on which we perform our experiment consists of all the written texts of the BNC sampler CD. Its size (about 1Mw) is average for manually corrected corpora, the tagset is well-developed (C7) and the tagging process has involved the use of an equally well-developed automatic tagger (CLAWS4) and subsequent correction by a team of experienced annotators (cf. Garside and Smith (1997)). We can assume that the consistency may not be as high as that of the LOB corpus, which by now has reached an admirable level of consistency, but certainly higher than notoriously inconsistent corpora like the Wall Street Journal (cf. van Halteren et al. (To appear)).

In the following sections, we first examine the concept of consistency (section 2), then describe the tagger generator used in the experiment (section 3), evaluate the output of the experiment (sections 4 and 5), and conclude by summarising the main findings (section 6).

2 Consistency and its Evaluation

It is generally agreed that one of the desired properties of any tagging is consistency, and that we therefore want to have some means of evaluating it. An important step towards such means is an examination of what this property of "consistency" is supposed to entail, beginning with a general definition of the concept:

> When we say that somebody is *consistent*, we mean that if the same situation is encountered more than once, that person will take the same action each time.

With this general definition in place, we can take a closer look at some aspects of the concept which are important for the specific activity we are interested in, viz. the tagging of text.

First of all, we have to distinguish between *internal* consistency and consistency with regard to a *defined standard* (aka *conformance*). With wordclass tagging, there is invariably some kind of defined standard, e.g. in the form of a tagging manual. In fact, the importance of the standard is often taken to be such that deviations from it are not just called inconsistencies, but that the stronger term "errors" is used.[1] It is this type of consistency which is measured in most evaluations of the tagged material and which is referred to with "correctness" or "accuracy" percentages. However, wordclass tagging is also assumed to correspond to a general descriptive linguistic tradition (whether "theory neutral" or not), which makes it very unlikely that any tagging manual can ever really be complete. The resulting friction between the (hopefully) clear but necessarily incomplete tagging manual and each tagger's personal conception of the underlying linguistic tradition cannot but lead to individual decisions. In these cases it is impossible to evaluate the consistency with regard to the standard, as the standard is partly incomplete (the manual) and partly not well-defined (the linguistic tradition). Instead, we will have to evaluate the internal consistency, i.e. the degree to which the individual decisions have been taken consistently.

The problem with the latter kind of evaluation is that, in wordclass tagging, the concept "same situation" can be taken at different levels of granularity. When taken only in the strictest sense, it would mean that the exact same word is occurring in the exact same context.[2] It is this sense which is used when, during a tagging project, inter-annotator consistency is measured. Several taggers are given the same text and their taggings are compared. This is useful for training purposes and improvement of the tagging manual and is also a good quality control mechanism if quality is seen in relation to the manual (and possibly the more exactly defined parts of the linguistic tradition). It is not, however, very useful in the evaluation of consistency between different parts of the corpus. Barring exceptional situations, such as news items which are repeated in several broadcasts, it is extremely unlikely that there are multiple occurrences of the same word combined with the same context. This is unfortunate, as such occurrences would be extremely easy to find, and hence compare, automatically.

Internal consistency is much more likely to be expressed in terms of the same "type" of word occurring in the same "type" of context. The question, then, is if and how we can determine which types of word in which types of context are tagged differently from occurrence to occurrence. The position taken in this paper is that, just as for the tagging process itself, the best choice is a combined effort by man and machine. For the time being, only man has sufficient knowledge of the actual aims of wordclass tagging and the generalisation skills to determine which situations are indeed "the same". On the other hand, the number of situations to be examined for inconsistency is much too large for exhaustive treatment, so that some kind of sampling is necessary. Seeing that random sampling tends to reveal only the most frequent inconsistencies (see below), we will have to use the machine to select situations with a high potential for inconsistency.

Now we may not have any algorithms ready at hand which detect inconsistency, but there are quite a number of algorithms which do the opposite: machine learning algorithms are designed to try to detect consistent behaviour in order to replicate it. In the context of wordclass tagging, machine learning algorithms come in the form of tagger generators, which automatically create tagging programs on the basis of a tagged training set. If we had the ideal tagger generator and a perfectly consistent training set, the generated tagger should be able to replicate the tagging in the training set completely. This means that errors made by a generated tagger must either be due to inconsistencies in

[1]Below, we will follow this choice of terminology and use the term "error" for tags which are inconsistent with regard to the standard, leaving the term "inconsistency" for those cases where (the description of) the standard provides no information on a "correct" tag and individual choices vary.

[2]Here, we take the context to be that which a human annotator would use to make decisions. This ought to be at least the whole sentence, but might well include the surrounding paragraph or more.

the training set or to insufficiency of the learning algorithm.[3] With both causes, the situations in which errors are made can be assumed to have a high potential for inconsistency: in the first case, they are related directly to inconsistencies; in the second, they are at least non-trivial and hence possibly more error-prone for humans as well. It would therefore seem to be a good idea to focus the human evaluator's attention on those tokens for which an automatic tagger's output and the original tagging disagree.

3 Tagger Generation

For the experiment in which we test this idea, we use a new tagger generator, which is based on the Weighted Probability Distribution Voting algorithm (WPDV; cf. van Halteren (To appear)). A tagger generated by this system goes through the following steps:

1. Normally, the first step would be *tokenisation*. In our experiment, however, we use the tokenisation as present in the original tagging of the corpus, as this makes comparison much easier. This means, however, that the intelligence embedded in the tokeniser is disabled. The most important example for the data at hand is that capitalised words in headings or at the start of sentences are not decapitalised but treated as is.

   There is one area where we have to deviate from the BNC tokenization. In the sampler material, multi-token units, such as "in front of", are present as a group of tokens which together receive one tag. As we want to detect inconsistency in this grouping as well, we translate such multi-token units to sequences of separate tokens, each tagged with a ditto tag. However, as no special treatment is present for such sequences in the tagger generator, they can be expected to be responsible for a good number of errors in the tagger output.

2. Next, the *lexical lookup* component attaches to each token a list of tags which were observed with that token in the training set. Note that, as mentioned above, the token "The", e.g. at the start of a sentence, is different from the token "the".

3. For those cases where lexical lookup provides no or insufficient information, we fall back on *lexical similarity lookup*. This means that potential tags are generated by a WPDV model, using the length of the token, its pattern of character types (e.g. "1980s" would be "one or more digits followed by one or more lower case characters") and its last three actual characters. The output consists of all tags which, according to this model, are at least 0.025 times as probable as the most probable tag for the token.

4. For tokens which were observed 10 times or more in the training set, only the output of the lexical lookup is used. For all other tokens, the output of the lexical similarity lookup is added. The resulting list of tags is used in two ways. Throughout the tagging process, the full list is used as a filter on the potential tags for a token, i.e. even if the context provides overwhelming evidence that a specific tag should be used, the tag is ruled out if it does not occur in the list. Additionally, the most probable tags in the list (up to three) are used to define an *ambiguity class* (cf. Cutting et al. (1992)) for the token, which is used in the context-dependent components. The lexical probabilities of the tags are used only to determine the selection for presence (and relative position) in the ambiguity class. They are not used in the context-dependent components.

5. In the main *context-dependent components*, two WPDV models then determine the most probable tag for each token on the basis of the (disambiguated) tags of two preceding tokens and the ambiguity classes of the focus and two following tokens. The difference between the two models is that one follows the normal order of the tokens, i.e. tags from left to right, while the other uses reverse order, i.e. tags from right to left.

6. The *final selection* of the tag for each token is determined by a WPDV model using the suggestions of the two context-dependent

[3]The latter obviously in relation to the size of the training set.

models for the focus and two tokens on either side of it.

There are two reasons for the selection of this particular tagger generator. First, an evaluation with the same training and test set used by van Halteren et al. (To appear) has shown this tagging strategy to compare favourably with other state-of-the-art tagger generators: 97.82% agreement with the test set versus 97.55% for TnT (Brants, 1999), 97.52% for MXPOST (Ratnaparkhi, 1996), 97.06% for MBT (Daelemans et al., 1996) and 96.37% for the Brill tagger (Brill, 1992).[4]

Furthermore, the use of WPDV allows leave-one-out[5] application for all components[6] so that the tagger can, without any additional effort, be used in two different modes: a) with the test set equal to the training set and b) with the test set disjoint from the training set. In the first mode, the tagger will have a very large amount of specific knowledge in each situation. We should expect errors under these circumstances to show "hard" inconsistencies, such as the same word receiving different tags in the company of the same tags in the direct context. In the second mode, the tagger is operating "normally", as if tagging unseen data. Here, we should expect "soft" inconsistencies, more to do with types of words and types of contexts than with exact words and contexts. We should also expect more errors due to tagger generator learning disabilities here, and the resulting higher error rate will force us to select a smaller fraction of the errors for detailed examination.

[4]These percentages have been measured on a 115Kw test set. This means that the 99% confidence intervals are 97.71–97.93%, 97.43–97.67%, 97.40–97.64%, 96.93–97.19% and 96.23–96.51% respectively.

[5]The normal way to test a tagger is by splitting the available corpus into separate training and test sets, and then train on the training set and test on the test set. In this way the test is fair, as the test data has not not been seen during training. The standard strategy is to split the corpus into 10 parts, and to repeat the train-test process 10 times, using each 10% part once as test data. This is called *10-fold cross-validation.* For some machine learning systems, however, it is possible to (virtually) remove the information about each individual instance from the model(s) when that specific instance has to be classified. This technique, called leave-one-out testing, in effect allows total cross-validation, e.g. for the case at hand one-million-fold.

[6]Even lexical lookup uses the WPDV system, so that we can use leave-one-out here as well.

4 Tagger-Corpus Disagreement

When a tagger is generated from the written text samples found on the BNC sampler CD, and used to re-tag those samples in the two modes described, we find an agreement rate of 99.45% when running without special measures (i.e. test equal to train) and of 96.93% when running with leave-one-out. In the first case, there are 6326 errors, in the second 35563. As we will want to compare the relative efficiency of using one run or the other, we want to examine similar numbers of errors in each case. Therefore, we take every 10th sentence for the first set (615 errors) and every 50th sentence for the second set (660 errors). Furthermore, we choose the two sets in such a way that the second set is a subset of the first one, so that we can evaluate the relative recall of the different runs. For the selected sentences, we examine all tokens where disagreement occurs.[7] In addition, in order to simulate random sampling, we take every 1000th sentence of the original corpus. For these sentences, we examine every single token (1210 tokens in total) for errors or inconsistencies in the corpus tagging, but without any reference to automatic tagger output.

Every disagreement (or observed error in the third group) is classified as to whether tagger and/or original corpus are right or wrong. Such a right-or-wrong decision is only taken if the tagging manual (or, as a backup, the linguistic tradition) is clear on the subject.[8] If such clarity does not exist, the full original corpus is inspected to determine if one of the possible tags is chosen in a substantial majority of instances of the same situation, in which case that tag is assumed to be the correct one. The resulting classification makes use of the following four classes:

T Tagger error. The original corpus is correct, the tagger is wrong.

B Benchmark error. The tagger is correct, the original corpus is wrong.

[7]We ignore all other tokens. This means that, if there are tokens which receive the same erroneous tag in both original corpus and tagger output, these will not be examined, and the error will not be detected.

[8]As we are taking the point of view of the average user, we use only the tagging manual that is found on the BNC Sampler CD. No reference is made to other manuals in the CLAWS tradition, such as Johansson (1986).

Table 1: Assignment of blame for corpus-tagger disagreement (see text for key).

		T	B	X	I
full run	615	416	121	5	73
leave-one-out	660	503	84	6	67
random sample	1210	-	6	-	18

X Extreme error. Both the original corpus and the tagger are wrong.

I Inconsistency. The manual does not indicate a single correct choice and the practice in the corpus varies.

The number of times these classes are found in each of the three examinations are listed in Table 1.

Both examinations based on disagreement between automatic tagger and corpus provide a high number of inconsistency-linked situations, certainly much higher than that provided by random sample examination. Which of the two tagger runs is more useful depends on what we intend to do with the results.

The most likely aim is the identification of all erroneous tags and inconsistencies in the original corpus. In this case, we are mostly interested in recall and the leave-one-out run is preferable. Assuming that the distribution of classes remains the same throughout the corpus, examination of all 35563 disagreements found with the leave-one-out run would yield 4850 (90/660 of 35563) corpus errors and a further 3610 (67/660 of 35563) tokens which are currently tagged inconsistently and which therefore may also have to be adjusted. With the full run, we would only have to check 6326 disagreements, but this inspection would yield only 1296 errors and 751 inconsistent tags (1 in 3.7 and 1 in 4.8). We see comparable figures when we examine the part of the corpus which has been checked for both runs:[9] only 25 of the 90 corpus errors which are detected because they are flagged by the leave-one-out run are also flagged by the full run (1 in 3.6) and 15 of the 67 inconsistencies (1 in 4.5).[10] However, even the higher recall of the leave-one-out run is insufficient to find all erroneous tags and inconsistencies. In the random sample, we spotted only 6 corpus errors, but of those 6 only 2 are flagged by either tagger run, and of the 18 spotted inconsistencies, 9 escape unflagged.[11]

However, the unflagged errors and inconsistencies all show similarities in context with errors and inconsistencies which have been flagged. Therefore, we can adjust our proposal and switch to a two-phase inconsistency determination:

1. use tagger disagreement to determine contexts where inconsistency occurs
2. examine all instances of those contexts in the full corpus

With the revised strategy, recall is only interesting with regard to the number of context classes which are identified, and precision is more important, as it helps increase the efficiency of the process. Furthermore, precision is also the more important property if we do not intend to identify and correct every individual error in the corpus, but only want to get a general impression of tagging quality. From Table 1, it would seem that the full run has a higher precision, as it contains 20.5% (126/615) errors and 11.9% (73/615) inconsistencies, versus 13.6% and 10.2% for the leave-one-out run. In the next section, we will examine whether it also has sufficient recall as to the context classes we want to identify.

5 Inconsistency Context Classes

Apart from classifying who is to blame for each disagreement, we have also classified all disagreements for the type of situation they represent, i.e. their *inconsistency context class*. This classification has been done manually, and it is here that the abovementioned need for human knowledge and generalisation skills becomes very clear. As an example, where "before" in "just before the film began" is tagged II (preposition) instead of CS (subordinating conjunction), we judge that it is a case of generic preposition-conjunction confusion, and

[9]Remember that the 1/50 part of the corpus checked for the leave-one-out run is a subset of the 1/10 part checked for the full run.

[10]There is only one inconsistency flagged by the full run which is missed by the leave-one-out run. There are no errors for which this is the case.

[11]These numbers are too small for a statistically sensible extrapolation to the whole corpus.

that there is no need for subclassification based on the actual word in question or on the context. However, where the same thing happens with "as" in "such a stiff fabric as damast" we decide that this disagreement belongs to a more specific class (confusion for the word "as"), since it is the comparison aspect of "as" which leads to conjunction being preferable to preposition. The creation of classes like preposition-conjunction confusion could fairly easily be done automatically, as they correspond to specific tag (or tag group) confusions and could be based on confusion lists and numbers of times the confusion is found. However, finer distinctions like "as"-confusion, or like confusion for words ending in "-ing" when in noun-modifying position, can best be decided on manually.

The final result of the classification for the examined disagreements is a list of 51 classes, which is shown in Table 2, together with the number of corresponding disagreements in the different evaluations.[12]

For most of the classes, we find corpus errors, sometimes very unexpected ones, e.g. 4 of the 5 "single letter" errors are instances of the personal pronoun "I" which are erroneously tagged as proper noun. For some classes, only tagger errors are found, but even these may be traced back directly to corpus errors elsewhere, e.g. the 2 "letter combination" errors are both tagger errors but are clearly caused by 16 misuses of the ZZ2 tag[13] in the corpus, and the consistent mistagging by the tagger of "in front of" as preposition-noun-preposition instead of a multi-token preposition is (at least partly) due to a single such mistagging in the corpus. Only rarely, e.g. with the confusion between present tense verb and infinitive, does it appear that the blame can be put entirely on the inability of the tagger generator to learn to make the necessary distinction.

This means that practically all classes are useful for the strategy proposed above, and the tagger runs hence have to flag instances of as many classes as possible. Examination of the table shows that both the full run and the leave-one-out run provide 49 of the 51 classes. This would indicate that either run can be used, as similar numbers of inspected tokens yield similar numbers of classes. However, we would advise using a combination of the two as this is likely to provide a more varied sample. Whatever sample of flagged tokens is used, after determining the inconsistency classes, it will be necessary to use specific searches on the whole corpus to determine which words (and/or which contexts) belong to those classes.

As an example, let us look at the "preposition vs -ing participle" class. The two tagger runs only show disagreements with "including", "excluding" and "following". However, a full search shows that "barring", "concerning", "considering" and "regarding" are also tokens which are sometimes tagged as preposition and sometimes as participle. At least "concerning" and "barring" appear to have some corpus errors connected to them. The situation is especially bad for "barring", where two of the three examples are suspect: in "laws barring the manufacture of cocaine" the tag II is chosen and in "barring a disaster, the payout will be the same" the tag VVG (ing-participle).[14]

6 Conclusion

The proposed method, generating a wordclass tagger from the tagged corpus and comparing its output with the original corpus, turns out to be an efficient means of identifying inconsistency in the corpus tagging. In both modes of operation, without special measures and with leave-one-out, a substantial percentage of disagreements are linked to inconsistency.

If one intends to eradicate all errors and inconsistencies, the method will have to be combined with other types of sampling, as not all instances are themselves flagged as disagreements. However, these other types of sampling can be based on a classification of contexts underlying inconsistency. Determination of the classes involved can be done by random sampling, but is much more efficient when done on the basis of the tagging disagreements.

[12]In those cases where a disagreement could be assigned to more than one class, the most specific class has been selected, e.g. a potential location-indicating noun (NNL) in noun-modifying position is classed as special noun type rather than generic noun modifier.

[13]ZZ2 is meant for plural forms of letters, such as "a's", but is also found for tokens like "AA".

[14]In the third example, at least, the wordplay "an unusual example of a gift barring Greeks" we find the correct tag, VVG.

Furthermore, if one decides that (some of the) additional sampling is too labour-intensive,[15] inspecting and, where necessary, correcting only the flagged tokens already provides a substantial consistency improvement. Which type of run to use for this probably depends on the available manpower. The leave-one-out run provides the best recall of errors and inconsistencies, but flags about five times more tokens than the full run.

With both choices of run type, the reduction of items to be checked is dependent on the quality of the generated tagger. For the corpus and tagger generator used in this paper, the number of flagged tokens is relatively low, and certainly low enough to be manually re-checked completely. For other tagged corpora or tagger generators, the relative number may well be higher, but we expect the method to be cost-effective as long as the annotation is limited to wordclass tagging. Something which has yet to be investigated is whether the use of the same tagger generator as has been employed during the original tagging of the corpus might interfere with the inconsistency detection. While this is uncertain, it seems wise to alway use a different type of tagger generator, which should not be a problem, given the wide choice of available systems.

For other corpus annotation tasks, such as word sense tagging or syntactic annotation, the quality of machine learning systems tends to be much lower. If the automatic re-annotation method is to be used here, we strongly suggest the use of several machine learning systems. Preferably these are then combined, e.g. as described by van Halteren et al. (To appear). If the combination system is still too inaccurate for a full inspection of all flagged items, the best items to check will be those where all (or at least a substantial majority of) the systems agree, but disagree with corpus annotation. After all, a wrong prediction by one or two systems can easily be blamed on a learning disability on the part of the systems, but the same wrong prediction by a majority of the systems is a strong indication that it is probably the corpus annotation that is mistaken.

[15]E.g. there are 22900 instances of tokens which can be either preposition or conjunction.

References

J.P. Baker. 1997. Consistency and accuracy in correcting automatically tagged data. In Garside, Leech, and McEnery (eds), *Corpus annotation*, pages 243–250. Addison Wesley Longman, London.

Thorsten Brants. 1999. *Tagging and Parsing with Cascaded Markov Models – Automation of Corpus Annotation.* Saarbrücken Dissertations in Computational Linguistics and Language Technology. German Research Center for Artificial Intelligence and Saarland University, Saarbrücken, Germany.

E. Brill. 1992. A simple rule-based part-of-speech tagger. In *Proc. of the Third ACL Conference on Applied NLP, Trento.*

D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. 1992. A practical part-of-speech tagger. In *Proc. of the Third ACL Conference on Applied NLP, Trento.*

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In Ejerhed and Dagan (eds), *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

R. Garside and N. Smith. 1997. A hybrid grammatical tagger: CLAWS4. In Garside, Leech, and McEnery (eds), *Corpus annotation*, pages 102–121. Addison Wesley Longman, London.

H. van Halteren, J. Zavrel, and W. Daelemans. To appear. Improving accuracy in NLP through combination of machine learning systems. *Computational Linguistics.*

H. van Halteren. To appear. Weighted Probability Distribution Voting, an introduction. In *Computational Linguistics in the Netherlands, 1999.*

S. Johansson. 1986. *The tagged LOB Corpus: User's Manual.* Norwegian Computing Centre for the Humanities, Bergen, Norway.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proc. of the Conference on Empirical Methods in Natural Language Processing, May 17-18, 1996, University of Pennsylvania.*

Table 2: Classes of inconsistency contexts (errors made in both corpus and tagger are shown as B and T rather than X, so that the sum of the blame types can be higher than the total count).

Context type		Full		Leave-one-out		Random	
special noun types	direction (ND)	2	(2I)	5	(5I)	-	
	title (NNA and NNB)	2	(1I 1T)	3	(1I 2T)	-	
	location (NNL)	13	(1B 12I)	20	(20I)	2	(2I)
	time (NNT)	2	(2T)	3	(2I 1T)	1	(1I)
	measure (NNU)	3	(2B 1T)	6	(3B 2I 2T)	-	
	day or month (NPD and NPM)	2	(2T)	-		-	
	capitalised word	38	(8B 5I 26T)	23	(10B 5I 12T)	4	(1B 3I)
	nominalised adjective	31	(8B 1I 21T)	19	(7B 12T)	2	(2I)
	nominalised -ing form	18	(5B 4I 9T)	17	(4B 13T)	4	(1B 3I)
noun modifiers or complements	-ing form (JJ vs VVG)	22	(3B 5I 14T)	23	(8B 5I 11T)	1	(1I)
	-ed form (JJ vs VVN)	32	(5B 9I 18T)	35	(7B 5I 23T)	1	(1I)
	-ed form of noun	1	(1T)	3	(1B 1I 1T)	-	
	-ist form (JJ vs NN)	1	(1I)	3	(3I)	-	
	capitalised word	36	(5B 5I 27T)	21	(5B 5I 12T)	3	(3I)
	other	25	(5B 4I 18T)	23	(4B 3I 16T)	1	(1B)
quantity-related	number of noun	5	(2B 1I 3T)	6	(1B 4I 1T)	-	
	quantification	16	(16T)	5	(5T)	-	
	modifier of number	12	(2B 6I 4T)	5	(2B 3T)	-	
verb tense	-ed form (past vs part)	39	(2B 1I 36T)	33	(7B 3I 23T)	-	
	base form (pres vs infin)	39	(39T)	21	(21T)	-	
	other	9	(9T)	8	(1B 7T)	-	
adverbs	adjectives used as (JJ vs R)	19	(1B 18T)	7	(2B 1I 4T)	-	
	function of (RG vs RR vs RP)	6	(6T)	4	(1B 3T)	-	
prepositions	vs conjunction	24	(5B 2I 16T)	13	(8B 5T)	-	
	vs verb particle	18	(1B 1I 16T)	21	(1B 20T)	-	
	vs locative adverb	2	(2T)	3	(3T)	-	
	vs verb participle	2	(1I 1T)	1	(1T)	-	
difficult words	as	10	(1B 9T)	12	(3B 9T)	-	
	his and her	4	(4T)	1	(1T)	-	
	once	3	(1B 2T)	-		-	
	one	7	(1B 1I 5T)	1	(1T)	1	(1B)
	's	2	(2T)	4	(2B 2T)	1	(1B)
	so	2	(2B 1T)	3	(1B 2T)	-	
	that	6	(2B 4T)	2	(1B 1T)	-	
	there	3	(1B 2T)	1	(1B)	-	
	to	8	(1B 7T)	10	(1B 9T)	-	
	when and where	6	(1B 5T)	8	(4B 4T)	-	
not English words	capitalised foreign word	-		9	(8I 1T)	-	
	foreign word	9	(1B 3I 5T)	10	(4I 6T)	-	
	formula vs digit-letter	7	(5B 3T)	9	(7B 2T)	-	
	single letter (ZZ1)	3	(1B 2T)	5	(4B 1T)	-	
	letter combination (ZZ2)	-		2	(2T)	-	
multi-token units	unrecognised	30	(2B 2I 26T)	69	(2B 67T)	-	
	falsely recognised	17	(8B 9T)	4	(2B 2T)	2	(2I)
impossible tag for token	capitalised words	6	(1B 5T)	7	(2B 5T)	-	
	other	30	(4B 26T)	25	(8B 17T)	1	(1B)
miscellaneous	untaggable words (FU)	1	(1T)	11	(11T)	-	
	strange spelling	8	(1B 8T)	14	(5B 11T)	-	
	capitalised words	18	(18T)	14	(1B 13T)	-	
	noun-verb confusion	48	(2B 46T)	51	(7B 44T)	-	
	other	13	(13T)	12	(3B 9T)	-	