Augmenting Automatic Grammatical Tagging

Hans Paulussen

FUNDP, rue de Bruxelles 61, B-5000 Namur, Belgium
hpaulus@cc.fundp.ac.be

Today, there are a large number of full-fledged Automatic Grammatical Taggers available, reaching a success rate of 95% or more. Nevertheless, the two main tagging approaches, probabilistic and rule-based, seem to be unable to overcome the final 5% errors. These errors are especially related to long-distance dependencies, and to semantic/pragmatic information which go beyond the AGT's knowledge sources. To improve their success rate, AGT's very often fall back to the use of ad hoc solutions, or they simply pass the unresolved tags to higher components: i.e. syntactic/semantic parsers. There are, however, still a number of optimizations possible, even within the limited constraints the AGT's are subjected to.

We propose a possible solution within a sub-linguistic context for the automatic lemmatizer-tagger DILEMMA. Our proposal is based on the fact that (i) "jargon" words in scientific texts tend to be used recurrently in the same text, and that (ii) they are less grammatically and semantically ambiguous than high-frequency words. If such a jargon word is not resolved, because of long-distance dependencies, we could scan the text for another occurrence of the same word, which has been unambiguously tagged. This solution was tested on a number of medical abstracts and has proved to be very promising.

Introduction

It is generally accepted that the first step in the computational analysis of written texts consists in assigning the right grammatical category to each word of the text. Practically, this means that every word is provided with a unique grammatical word label or *tag*.

There are two ways to grammatically tag a word in a sentence. Either the grammatical information is found morphologically in the word or the syntactic context of the word shows to which grammatical category the word belongs. An example of the functional definition of a category is given in the following sentence where the word *book* has different grammatical categories according to its position in the sentence: "*Book* this *book* in the ledger"[1]. As the number of words used in the context before and/or after the unresolved word, is usually very small, –2 to 3 words suffice in most cases– it is one of the key parameters in most of the present-day taggers. Although this local span has very powerful grammatical predictability, there is, however, a small number of cases where this local knowledge is not sufficient at all.

Although we will limit ourselves here to stand-alone taggers, word tagging or annotation is also found in many other programs, such as parsers. The difference between stand-alone taggers and parsers is that the latter do not store the tagging information permanently. As such, automatic grammatical tagging can be seen as a *permanently stored explicitation of the grammatical categories of a text.* Another difference between parsers and taggers is that the latter only use a *bottom-up* strategy, whereas the former use different strategies, next to bottom-up and *top-down*. But this is also clear from the fact that taggers have a low-level task to carry out, whereas parsers work on a higher syntactic level. In this way, taggers are very often used as pre-processors for parsers.

In what follows, we will first give a general comparison between the two main tagging approaches: the rule-based and probabilistic paradigm. Then we describe the most

---

1 Even if the contextual problem is more outspoken in English, where the functional categorial shifts of homographs are a general phenomenon, it is not limited to morphologically poor languages; similar disambiguation problems occur in e.g. Finnish and Arabic, which are morphologically highly complex.

important features of DILEMMA, a rule-based lemmatizer-tagger. This is followed by a description of the augmentation we carried out on DILEMMA for the sublinguistic field of medical abstracts. We conclude with a discussion on further augmentations and the evaluation of AGT's.

Rule-based and probabilistic approach

Automatic grammatical tagging is usually divided into two approaches, related to the two points of view found in natural language processing in general: the *rule-based* and the *probabilistic* approach[2]. The proponents of the rule-based approach try to put all linguistic knowledge into rules. This implies that they should have a very good knowledge of the language, and that their database of linguistic rules is built up on trial and error. Hence, they are often referred to as the *hypothesis-driven* approach. The probabilistic or stochastic model, on the other hand, is a *data-driven* approach, in the sense that statistic information about the probable tag sequences is acquired on the basis of an analysis of a sample corpus. This training corpus is usually tagged, (semi-) automatically or by hand (Cutting *et al.* 1992:133). On the basis of the tagged training corpus *collocational probabilities* are calculated: i.e. the probable category of each word type in the sample text (lexical probability) and the probability of one category following another (contextual probability). These calculations, usually based on a first or second order Markov model, are placed in a *probability transition matrix*, which is then used for tagging new texts.

In Table 1 a few examples of both types of taggers are compared. Although the information is very general, there are a few striking points[3]. Note that the largest dictionaries are used by the probabilistic taggers: i.e. CLAWS and the PARC tagger. Note

---

2 Another new approach is the *connectionist* model (Benello, Mackie & Anderson 1989, Elenius & Carlson 1989, Nakamura & Shikano 1989).

3 The table lists first the name(s) of the author(s) and the name of the tagger. Then comes the success rate as mentioned in the publications, followed by the size of the dictionary, the number of suffixes and the number of tags used. The next item, the number of rules used, is of course only relevant for a rule-based tagger. The last item, the context span, concerns the number of words preceding and/or following the word to be disambiguated.

also that the context span used by the PARTS tagger is a clause, and that CLAWS indicates that it can have a span up to 10 words, although this is very rarely used. The first tagger, TAGGIT (Greene & Rubin 1971), which was developed to tag the BROWN corpus, has always been considered the model of the rule-based taggers. However, rule-based taggers have changed considerably over the years, as can be seen when comparing TAGGIT with DILEMMA, for example, the latter scoring a much higher success rate and using a very small rules component.

author	Greene & Rubin (1971)	Cherry (1978)	Marshall (1983), Booth (1985)
name	TAGGIT	PARTS	CLAWS
success rate	77%	95%	97%
dictionary	3000	407	7000
suffix lists	450	51	700
tags	71	29	134 (Marshall)
rules	3300	101	Markov model
context span	5	clause	<= 10
author	Martin, Heymans & Platteau (1988)	Brill (1992)	Cutting, Kupiec, Pedersen & Sibun (1992)
name	DILEMMA	?	PARC tagger
success rate	95%	95%	96%
dictionary	3600	?	~50000
suffix lists	185	?	130
tags	12 categories 25 subcategories	> 150	87
rules	300	< 100	HMM
context span	3 words	3 (?)	2

Table 1

Each approach has its particular advantages and drawbacks. The rule-based taggers are by definition language dependant, so that an adaptation of the tagger for another language implies a new compilation of the database of linguistic rules. Another shortcoming is that it apparently takes too much time to build the rule component, which thus can become too complex to handle. We have experienced in DILEMMA, however, that the problem very often does not lie in the complexity of the rules but in the exchange of information

between those rules. Moreover, the number of rules used in DILEMMA are very easy to manipulate and to adapt.

Shortcomings in the stochastic taggers are mostly related with the characteristics of probabilistic tables, which require very large memory space, thus necessitating ingenious memory management. Also the finiteness of the training corpus may cause problems. If a word does not occur in the training corpus, its probability is zero. In order to get a representative corpus, the probability must be calculated from very large texts[4]. Stochastic taggers usually solve the problem of zero-frequency with *smoothing* techniques, so that a non-existent word will always get a probability, but a very small one, just above zero.

Although stochastic taggers are rather easy to construct[5], the initial task of tagging a training-corpus by hand (or semi-automatically) is unfortunately a very time-consuming activity. Stochastic taggers using a regular Markov model always need large sized training corpora[6]. This problem, however, has partially been solved, since the introduction of Hidden Markov Models (HMM)[7]. Whereas regular Markov models are based on tagged training corpora (i.e. supervised corpora), in the case of a HMM an untagged corpus is used (i.e. unsupervised corpora). Moreover, the training corpus can usually be kept rather small. The original but difficult task in the HMM tagger consists in finding the right parameters which indicate the most probable transition path. This is usually calculated by using the Baum-Welch algorithm, also called forward-backward algorithm (Baum 1972)[8]. A disadvantage is that in order to estimate the right parameters different smoothing and biasing techniques are necessary. Although Foster (1991:42) has tested that training on

---

4 Maltese & Mancini (1991:753) used a 100-million word corpus to estimate the probabilities of word triples.

5 The appeal of this data-driven approach comes from "the ease with which the necessary statistics can be automatically acquired and the fact that very little handcrafted knowledge need be built into the system" (Brill 1992:152).

6 Cfr. Maltese & Mancini (1991:755) use a reference vocabulary of 260000 inflected forms. To improve vocabulary access performance the 20000 most frequent words — the base vocabulary — are permanently stored in central memory.

7 The *Hidden Markov Model* (HMM) has become increasingly popular in low-level speech recognition and synthesis (Giustiniani & Pierucci 1991, Wilcox & Bush 1991), and has later been introduced in taggers as well (Jelinek 1985, Kupiec 1989, Maltese & Mancini 1991, Cutting *et al.* 1992).

8 A comparison of different probabilistic tagging algorithms is given in Merialdo (1991).

an untagged corpus is a very weak method, new techniques seem to have lead to considerable improvements (Cutting *et al.* 1992).

Whatever the advantages and drawbacks of either approach, the stochastic taggers seem to score usually slightly higher than the rule-based taggers. Nevertheless, as the proverb goes: a miss is as good as a mile, and both approaches seem to be powerless when trying to resolve the final 5% of untagged words. The main reason is that to solve these errors, the tagger has to go beyond the local span which forms the limit of the AGT's knowledge sources. There are, however, still a number of optimizations possible, as we will show in the DILEMMA example here below.

DILEMMA-1

In this section we would like to explain the most important features in which the program DILEMMA differs from other rule-based taggers[9]. DILEMMA is unique in its *dictionary structure*, the use of a *categorial graph* and the use of *direction pointers*.

Unlike most tagger dictionaries, DILEMMA does not list next to each entry all the possible categories that entry can have. Instead, categorial information is stored preferentially. Each word is given a preferential default category (DC in Table 2) which can shift to other categories along a categorial graph (Figure 1), guided by the condition-action rules of the rules component in the DILEMMA-program. A dictionary is then structured as follows (see Table 2): each entry word is followed by its *lemmatized form* and a preferential *default category*. Then comes a *pointer* indicating in which direction the category can shift according to the categorial graph shown in Figure 1. Where necessary, one or more *specifiers* (or sub-categories) are mentioned. The use of default categories and direction pointers changes the dictionary into an economical and *dynamic* set of lexemes.

9 For further details on the DILEMMA architecture, we refer to Martin, Heymans & Platteau (1988). Note also that DILEMMA, unlike most other taggers, is not only a tagger but also a lemmatizer.

word form	lemma	DC	pointer	specifiers
king	king	noun	l	
kiss	kiss	verb	r	
kit	kit	noun	l	
kitchen	kitchen	noun	n	
knee	knee	noun	l	
kneel	kneel	verb	n	
knelt	kneel	verb	n	pastpapa
knew	know	verb	n	past

Table 2

The possible shifts from one category to another are indicated in the categorial graph[10]. For example, the word "kiss" mentioned in the dictionary sample (Table 2), has the category `verb` as DC, and can shift `right` to the `noun` category.

There are two sets of categories. The first set contains three *open* categories: `noun`, `verb` and `adjective`. These categories have an infinite number of members. The second set contains all the other categories. These categories are *closed*. The graph shows clearly that all closed categories can shift into the open categories, and that the open categories can never shift into the closed set. In this way, the categorial shifts are linguistically motivated.

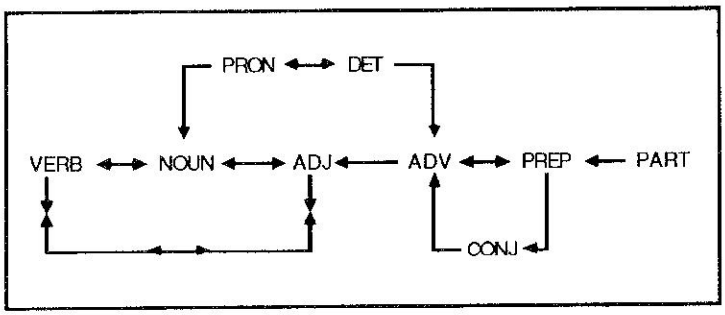

Figure 1

There are three types of direction pointers: n(eutral), l(eft) and r(ight), showing the direction in which the category can shift. The neutral pointer is used when no further shift is possible. In this way the neutral pointer is a *flag* indicating the categories of which DILEMMA is sure they have been categorised correctly.

10 When a word cannot shift from one category to another according to the pointers in the categorial graph, it has to be entered twice in the dictionary, and thus gets two default categories. The authors of DILEMMA claim that "it is never a coincidence that a word has two DC's. For, if a word has two DC's, this means that its syntactic and/or semantic behaviour is in some way anomalous." (Martin, Heymans & Platteau 1988:19) That is why in the graph an adjective cannot shift to an adverb, as English adjectives are normally formed by adding `-ly` to the stem. This is the case for the following examples: *alone*, *cheap*, *deep*, *early*, *loud*, etc.

DILEMMA scored very well on a number of general language text samples. A sample of error analysis on 6 texts taken from the LOB Corpus shows e.g. that for general language texts, DILEMMA's success rate does not go below 90% and does not exceed 97% either, on the average leading to a score of ± 93.50%[11] (see Table 3).

text number	total of wordtokens	total of errors made	error percentage	relative success rate
1	123	6	4.87	95.13
2	183	7	3.82	96.18
3	196	7	3.57	96.43
4	143	14	9.79	90.21
5	239	17	7.11	92.89
6	246	19	7.72	92.28
1-6	1130	70	6.14	93.86

Table 3

However, when testing DILEMMA on a number of medical abstracts, the scoring reference point of 93.50% was not reached at all. "Best results" were more likely to lie within the 85%-90% area, the average being about 86% (Paulussen & Martin 1992:146). In order to correct these results we developed DILEMMA-2.

DILEMMA-2

DILEMMA-2 was an improvement on DILEMMA-1 for the specific purpose of lemmatizing-tagging medical abstracts. This adaptation was developed in two stages. The first stage concerned especially the introduction of a sublanguage specific suffix list and a sublanguage specific gaps filler default[12]. The results of the first stage, summarized in Table 4, show that the modifications in DILEMMA-2 have improved the success rate considerably, even passing the scoring reference point of DILEMMA-1 (93.5% vs. 96%).

11 Martin, Heymans & Platteau (1988) used another, larger sample of texts and scored a success rate of 95%.

12 A description of this first step is found in Paulussen & Martin (1992).

	text 1	text 2	text 3	text 4	text 5	text 6	TOTAL	
total amount of words	186	86	150	176	242	233	1073	
wrong assignments	12	3	6	5	22	5	53	
gaps	21	7	14	15	32	11	100	
total amount of errors	33	10	20	20	54	16	153 (14.25%)	success rate DILEMMA-1 85.75 %
total amount of corrections	25	7	16	14	37	9	108	
remaining errors	8	3	4	6	17	7	45 (4.19%)	success rate DILEMMA-2 95.81 %

Table 4

The second stage consisted in a better memory management for detecting jargon words which were not yet resolved. We found out that a number of recurring words were not always grammatically disambiguated: i.e. in the same text a word was in one place resolved and in another place left untouched. There were two striking points about this phenomenon.

First, the problem is related more to technical texts than to general texts[13]. This was especially the case for the medical abstracts we have analysed for DILEMMA-2. Secondly, most of these unresolved words were low frequency words in general texts, but very frequent in the technical texts; we could call them *text-specific* or *jargon* words. An important point here is that jargon words, being used in a restricted context, are very often monosemic, so that copying information of resolved words to unresolved ones is less of a problem in sublinguistic texts than in general texts (see Deville 1989:92)[14]. A typical example is the list of words in Table 5 coming from a text of 459 words (and punctuation marks) containing 17 unresolved words (4%). A striking point here is that (i) "jargon" words tend to be used recurrently in the same text, and that (ii) they are less grammatically and semantically ambiguous than high-frequency words.

13 Similar observations on the semantic level are found in Janssen (1990).

14 Sublinguistic features are not limited to the lexicon, as one often is inclined to think (see Gopnik 1972, Kittredge & Lehrberger 1982, and Kittredge & Mel'cuk 1983).

0	458	cysts	*
1	116	eccrine	
2	144	eccrine	
3	324	eccrine	
4	420	gland	
5	402	gland	
6	215	gland	
7	8	gland	
8	320	nodular	
9	97	nodular	
10	275	nodular	
11	230	nodular	
12	183	sole	
13	9	tumour	
14	83	tumours	
15	105	tumours	*
16	312	tumours	

Table 5

As the solution to these unresolved words is often related to jargon words, of which some are disambiguated, we wanted to solve the disambiguation problem by using a *temporary memory* of disambiguated words, where the unresolved words could be scanned for a possible match. This idea was not new to DILEMMA, which already did use a temporary memory; yet, its scope was limited to the sentence. The whole concept of the DILEMMA program is restricted to sentence analysis. Our idea, now, was to use the complete text as a *knowledge database* for solving the errors left over.

In this way, we decided to filter through only disambiguated *nouns* and *adjectives*[15] which have a *neutralized direction pointer*: i.e. a pointer which no longer can shift to left or right. This last point is very important, as we can only base ourselves on the solutions of which DILEMMA is sure that they are completely solved.

We have tested the program on 10 medical abstracts, but although the jargon words were all detected, the success rate was not so eye-catching when compared to the first implementation of DILEMMA-2, especially as the impact of the corrections was only perceived on a very small set of final errors. In the case of the six texts of Table 4, the remaining errors dropped from 45 to 38, thus raising the success rate with almost 1% (96.45%).

15 We knew from previous analyses of medical abstracts that most of the unresolved words are potential *nouns* or *adjectives*.

Discussion

The development of DILEMMA-2 has resulted in a better success rate of the DILEMMA-progam, even if the corrections are felt in a minor scale and within a sublinguistic domain. Nevertheless we think that the DILEMMA-program, like many other taggers of both approaches, has reached a plateau which is very difficult to overcome.

Depending on the application, one could improve in a sublinguistic or domain specific field, without necessarily recurring to ad hoc solutions. For general purpose taggers, on the other hand, solutions can only be found in higher order semantic/pragmatic resources which surpass the local scope knowledge which forms the core of all present-day taggers.

Instead of adding semantic/pragmatic knowledge and thus develop a parser, it would be better to weigh the tagging output in such a way that manual checking becomes redundant. It is better to output a 60% tagged text where every tagged word has been assigned with 100% certainty than a 90% tagged text where every tag has to be completely checked again by hand. The neutralized direction pointer in DILEMMA is one of such certainty flags. This implies that we and the program itself know what type of error it makes. Foster (1991:64) gives a good example of how to take account of these error checking capacities. Most of the errors are related to local scope dependencies[16]: the confusion between nouns and adjectives, past participles and preterites are typical examples.

The automatic detection of tagging errors would better clarify the success rate of taggers, which at present too often is mystified by lack of detailed information[17]. In this way, one would be able to develop objective criteria for the evaluation of taggers, showing in which situation which approach should be preferred. Maybe the time has come to make a thorough evaluation of all existing taggers.

---

16 Macklovitch (1992) gives a detailed description of the typical tagging errors and proposes some interesting solutions. Contrary to what he says, the errors are not limited to stochastic taggers. Another description of tagging errors is given by DeRose (1989).

17 Van Halteren (forthcoming) describes a possible strategy for the evaluation of taggers, taking into account the major factors which influence the success rate of taggers: e.g. size and detail of tagset, completeness of the lexicon, the treatment of multi-token words.

Bibliography

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

Benello, J., A.W. Mackie & J.A. Anderson (1989), "Syntactic category disambiguation with neural networks", in *Computer Speech and Language*, no. 3, 203-217.

Booth, B.M. (1985), "Revising CLAWS", in *ICAME News 9*, 29-35.

Brill, Eric (1992), "A simple rule-based part of speech tagger", in *Proceedings of the Third Conference on Applied Natural Language Processing* (ACL), Trento, 152-155.

Cherry, L.L. (1978), *PARTS — A system for assigning word classes to English text*, Computing science technical report No. 81, Bell Laboratories.

Cutting, Doug, J. Kupiec, J. Pedersen & P. Sibun (1992), "A practical part-of-speech tagger", in *Proceedings of the Third Conference on Applied Natural Language Processing* (ACL), Trento, 133-140.

DeRose, Steven J. (1989), *Stochastic methods for resolution of grammatical category ambiguity in inflected and uninflected languages*. Doctoral dissertation. Providence, Rhode Island: Brown University, Department of Cognitive and Linguistic Sciences.

Deville, Guy (1989), *Modelization of task-oriented utterances in a man-machine dialogue system*, Doctoral dissertation, University of Antwerp.

Elenius, K. & R. Carlson, "Assigning parts-of-speech to words from their orthography using a connectionist model", in *Proceedings of the European Conference on Speech Communication and Technology*, Vol. 1, 534-537.

Foster, George F. (1991), *Statistical lexical disambiguation*, Master's thesis, McGill University, School of Computer Science, Montreal, Canada.

Giustiniani, M. & P. Pierucci (1991) "Phonetic ergodic HMM for speech synthesis", in *EUROSPEECH 91, Proceedings of the 2nd European Conference on Speech Communication and Technology*, Genova, Italy, Vol. 1, 349-352.

Gopnik, Myrna (1972), *Linguistic structures in scientific texts*, The Hague: Mouton.

Greene, Barbara B. & Gerald M. Rubin (1971), *Automatic grammatical tagging of English*, Unpublished manuscript, Providence, Rhode Island, Department of Linguistics, Brown University.

Halteren, Hans van (forthcoming), "Comparison of tagging strategies, a prelude to democratic tagging", to appear in Hockey, Susan & Nancy Ide, *Selected papers from the ALLC-ACH92 Conference*, Oxford 1992.

Janssen, Sylvia G. (1990), "Semi-automatisch desambigueren met behulp van een machine-leesbaar lexicon", in *TABU, Bulletin voor Taalwetenschap*, 20 (2), Groningen, 126-136.

Jelinek, F. (1985), "Markov source modelling of text generation", in Skwirzinski, J.K. (ed.), *Impact of processing techniques of communication*, Dordrecht: Nijhoff.

Kittredge, R. & J. Lehrberger (eds.) (1982), *Sublanguage: studies of language in restricted semantic domains*, Berlin: Walter de Gruyter.

Kittredge, Richard & Igor Mel'cuk (1983), "Towards a computable model of meaning-text relations within a natural sublanguage", in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Vol. 2, 657-659.

Macklovitch, Elliott (1992), "Where the tagger falters", in *TMI-92*, Fourth International Conference on Theoretical and Methodological Issues in Machine Translation, Montreal, 113-126.

Maltese, Giulio & Federico Mancini (1991) "A technique to automatically assign parts-of-speech to words taking into account word-ending information through a probabilistic model", in *EUROSPEECH 91, Proceedings of the 2nd European Conference on Speech Communication and Technology*, Genova, Italy, Vol. 2, 753-756.

Marshall, I. (1983), "Choice of grammatical word-class without global syntactic analysis: Tagging words in the LOB Corpus", in *Computers and the Humanities* 17, 139-150.

Martin, W., R. Heymans & F. Platteau (1988), "DILEMMA, an automatic lemmatizer", in W. Martin (ed.): *COLINGUA 1, Antwerp Papers in Linguistics 56*, Antwerp, 5-62.

Merialdo, B. (1991), "Tagging text with a probabilistic model", in *Proceedings of ICASSP-91*, Toronto, 809-812.

Nakamura, Masami & Kiyohiro Shikano (1989), "A study of English word category prediction based on neural networks", in *ICASSP 89: International Conference on Acoustics, Speech and Signal Processing*, 731-734.

Paulussen, Hans & Willy Martin (1992), "DILEMMA-2: a lemmatizer-tagger for medical abstracts", in *Proceedings of the Third Conference on Applied Natural Language Processing* (ACL), Trento, 141-146.

Wilcox, Lynn D. & Marcia A. Bush (1991) "HMM-based wordspotting for voice editing and indexing", in *EUROSPEECH 91, Proceedings of the 2nd European Conference on Speech Communication and Technology*, Genova, Italy, Vol. 1, 25-28.