Generating Subtitles from Linguistically Annotated Text – ATraNoS WP4-11 & WP4-12 –

Erik F. Tjong Kim Sang
CNTS - Language Technology Group
University of Antwerp
erikt@uia.ua.ac.be

1 Introduction

This progress report describes the current (October 2003) status of the subtitle generator that is being developed in the ATraNoS project[1]. The generator provides summaries of individual sentences which will be supplied by a speech recognition system. The summarized sentences will be used as tv subtitles for hearing-impaired people. This report describes follow-up work on the report "A Baseline Subtitle Generator" (Tjong Kim Sang, 2003). The main target of the current work is to find out in what way linguistic annotation of the input sentences can aid the quality of the summarization process.

In section 2, we will start with an overview of the errors of the baseline system and show what syntactic information will be valuable for summarizing. After this we will describe the corpus we will use and the machine learning methods which will be used for extracting the desired information from the corpus. The third section contains an overview of the experiments we have performed in order to obtain syntactic information. In section 4 we compare the performances of the baseline subtitle generator with the one that has access to linguistic information. Section 5 contains concluding remarks and suggestions for future work.

2 Baseline Systems

In (Tjong Kim Sang, 2003), we describe two baseline systems for summarization: one based on machine learning and one that used hand-crafted deletion rules. Both systems used words and some syntactic information (word classes) although it must be said that the quality of the latter was not very high. The main problem for the learning approach was that the system that obtained an optimal accuracy in predicting deleted and replaced words did not

[1] http://atranos.esat.kuleuven.ac.be/

generate syntactically correct sentences. The basic rules of the rule-based summarizer failed often when either the quality or the quantity of the annotation of the data was insufficient. We believe that the availability of extra relevant linguistic annotation will diminish both problems.

Here is an incomplete overview of the errors made by the two baseline systems (square brackets denote deleted words):

Welkom bij [*het zeven uurjournaal*] (learning approach)

The system removes a noun phrase (NP, *het zeven uurjournaal*) from preposition phrase (PP, *by het zeven uurjournaal*) and leaves an incomplete phrase. It would probably have helped if the system had access to PP information but this was not available. The baseline system had access to word class information (part-of-speech tags) and information about noun phrases. An improved system should have access to information about other phrase types.

[*Yasser Arafat*] *roept* (learning approach)

In this case the system removes the subject of a sentence. Information about subjects and objects was not available to the baseline systems and this should be added.

roept [*op de aanslagen*] *te stoppen* (rule-based approach)

One of the deletion rules in the rule-based system states that all prepositional phrases can be removed. The phrase *op de aanslagen* is identified as a prepositional phrase because it consists of a preposition followed by a determiner and a noun. However, in this case *op* is not a preposition but a verb particle. It belongs with the verb *roept* en therefore it should not be deleted. This information is not available in the part-of-speech tagging scheme we used. It should be available as extra syntactic information.

[*Al*] *Qaeda* (rule-based approach)

Another deletion rule in the rule-based system specifies that in a sequence of names, all but the last can be removed. This rules works fine for person names that consists of two words but it will fail for names of other types which cannot be reduced. In order to take care of this problem the summarization systems need to have access to more elaborate name information. They need to know what type of name they deal with (person name or something else) and within the person names they need to know the difference between first names and surnames.

In (Tjong Kim Sang, 2003), we have shown that the quality of the available syntactic information is very important. Errors in for example word class information will have a negative influence on the summarization results. It speaks for itself that we prefer extra syntactic information that is of good quality.

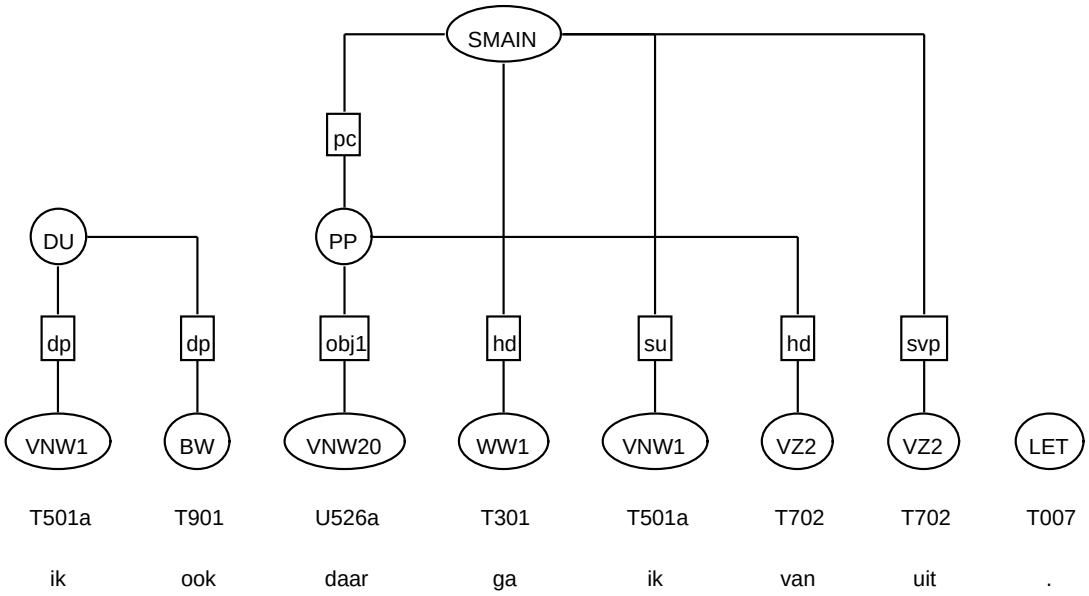

Figure 1: Example of a syntactically annotated tree in the Spoken Dutch Corpus.

3 Data and Methods

We will annotate data with machine learning methods trained on syntactically annotated corpora. This section contains a description of these learning algorithms and the data which we will use for training them

3.1 Corpora

In order to analyze text with machine learning we need annotated training examples. At this moment there are two syntactically annotated corpora for Dutch available: the Spoken Dutch Corpus[2] and the Alpino Treebank[3]

The Spoken Dutch Project (CGN) develops a 10-million-word corpus of spoken Dutch. All words in the corpus will receive a lemma and a part of speech tag and about 10% of the sentences will be annotated syntactically with parse trees. The current version of the corpus (release 6) contains about six million tokens of which about half a million tokens have received a full syntactic annotation.

An example parse tree can be found in Figure 1. The CGN material contains information which is relevant for solving three of the summarization problems mentioned in the previous section. First, the syntactic trees defines phrases of different types, like for example the

[2] http://lands.let.kun.nl/cgn/
[3] http://www.let.rug.nl/~vannoord/trees/

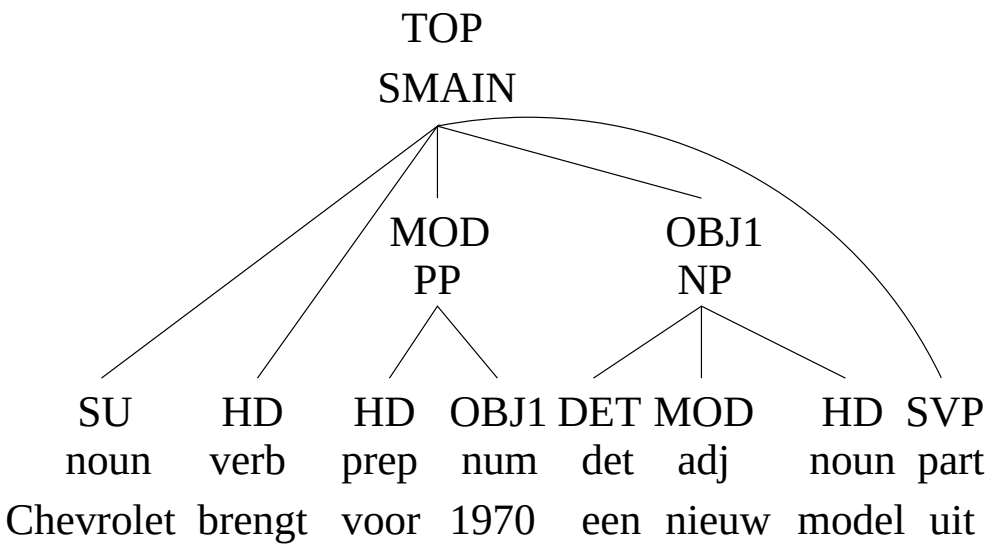

Figure 2: Example of a syntactically annotated tree in the Alpino Dependency Treebank.

prepositional phrase *daar van* in the example sentence. They also contain information about verbs and their subjects (*ik* in the example sentence). Finally, the trees tie isolated particles to the corresponding verbs, like *uit* to *ga* in the example sentence. The corpus contains information about proper names as well but no distinction is being made between different types of proper names.

The syntactic annotation contains two features which may complicate future usage. First, the trees contain crossing brackets. This means that phrases are discontinuous and this will make an automatic identification of phrases difficult. However, due to the nature of Dutch, crossing brackets in the trees are difficult to avoid. Problems in the phrase identification may also be caused by the fact that the depth of the trees is small. For example, one could have expected the subject of the example sentence *ik* to be marked up as a noun phrase and *ga uit* as a verb phrase. However, both levels are missing and the three words have been put immediately under the sentence node. It remains to be seen whether these choices can be generated by an algorithm.

The Alpino Dependency Treebank consists of sentences of the newspaper part of the Eindhoven corpus. The present release contains about 7000 sentences with about 140,000 tokens. The annotation format is quite similar to that in CGN although the part-of-speech features have received a less elaborate annotation. An example sentence of this corpus can be found in Figure 2. The annotated trees contain the same useful features as in CGN: explicit annotation of phrases, subjects and verb particles. However, they also lack useful annotation of phrases that only contain one word, like *Chevrolet* in the example sentence which has not been marked up as a noun phrase.

We have chosen the CGN data for generating syntactic models for Dutch because there is more CGN material available than is present in the Alpino treebank. The data has been divided in two sections: material that has received full syntactic annotation (461,409

tokens) and material that contains part-of-speech information (5,983,309 tokens). Each of these sections has been divided in three parts: one for training, one for evaluating the learning approaches during the parameter tuning process (development data) and one extra held-out test set. More information on these data sets can be found in the sections on part-of-speech tagging and chunking.

3.2 Learning algorithms

We will use memory-based learning (Daelemans et al., 2002b) for building syntactic models from the corpora. A memory-based learner stores all training data. Each test item is compared with the training data and the output class is chosen of the training item which is most similar to the test item. There are different ways to store the training data and compare the distances between items. We use a nearest-neighbor algorithm (IB1) with the gain ratio variant of information gain weighting of features combined with the overlap distance metric (see (Daelemans et al., 2002b) for background information). Unless specified otherwise we only look at the training item that is closest to the test item under investigation (that is, we do not perform smoothing).

The software environment which we have used for our experiments is the memory-based tagger MBT developed in Tilburg, The Netherlands (Daelemans et al., 2002a). It is a shell built on a general memory-based learner. The extra features of MBT, like standard inclusion of orthographic features and separate treatment of frequent, infrequent and unknown words, allow the user the quickly develop, test and use a good memory-based classifier. The system has one disadvantage: it can only process information of a single source, for example only words from only part-of-speech tags but not both at the same time. This means that theoretically its optimal performance is inferior to that of a general memory-based learner which can combine information from different sources. However, in our earlier work with named-entity recognition we have found it hard to use a general learner to improve on results obtained with MBT.

MBT allows the user to specify different parameters. Generally we have kept the default parameters of the general underlying learner and changed only the MBT-specific parameters:

- a number of words (parameter w) before and after the focus word (both for known and unknown words),
- a number of output classes to the left (d) and a number of sets of possible output classes to the right (a) of the focus word (both for known and unknown words),
- a number prefix characters (p) and a number of suffix characters (s) of the focus word (unknown words only),
- the presence of the current word (W) and its set of output classes (f, known words only), and
- the presence of capitalized characters (c), hyphens (h) and numeric characters (n) in the focus word (unknown words only).

The single characters between brackets in this list are the names of the parameters[4]. In total the parameters contain ten integer values and five binary values. We have always included the the set of output classes of the focus word for known words. For the remaining fourteen features we have searched for the optimal set with bidirectional hill climbing (Caruana and Freitag, 1994) starting from a set of zero-values. This means that we have progressed through the 14-dimensional feature space by performing an experiment for each feature set that differs in one position (+1 or –1) from the point under investigation and choosing the best performing set as the next point to examine. This procedure was halted when the performance of the current point was not improved by one of its neighbors.

In order to check if our results are reasonable, we have also applied a second tagger to the data sets: the Hidden Markov Model tagger TnT (Brants, 2000). This tagger has achieved state-of-the-art results for part-of-speech tagging. TnT is both fast and accurate but it does not allow elaborate parameter tuning. We are using it as a comparison method only. Even if this tagger would perform well, it is uncertain whether its license allows future application by the commercial project partners.

4 Experiments

In this section we will describe the experiments which we have performed for building syntactic analyzers for Dutch text. We will discuss four components: a part-of-speech tagger, a lemma finder, a text chunker and a relation finder. We will also mention the development of three other modules which will be useful for summarization of spoken Dutch text: a person name finder, a capitalization module and a module for punctuation insertion. We will discuss the data used, the experiments performed, and the results that were achieved.

4.1 Part-of-speech tagging

The first task we have examined is part of speech tagging: assigning word classes to words. We have used the CGN material as training and test data. CGN contains a rich tags set in which most of the part-of-speech tags have different attributes. Here is an example sentence from the corpus:

> word/WW(pv,tgw,ev) je/VNW(pers,pron,nomin,red,2v,ev) hier/VNW(aanw,adv-pron,obl,vol,3o,getal) nou/BW() wakker/ADJ(vrij,basis,zonder) van/VZ(fin) ?/LET()

An overview of the different tags that appear in the data set can be found in Table 1. We have removed sentences in which the speaker was registered as comment, background information or unknown. The remaining data has been divided in three parts: development

[4] An extra dummy parameter F is used for unknown words for keeping left and right context words apart, as in wFw.

tags		subtags	name
350,116	6%	35	ADJ
579,973	10%	2	BW
611,134	10%	1	LET
365,993	6%	10	LID
768,683	13%	20	N
175,035	3%	8	SPEC
369,581	6%	3	TSW
76,918	1%	15	TW
338,277	6%	3	VG
901,923	15%	187	VNW
524,455	9%	4	VZ
921,221	15%	27	WW

Table 1: Major category part-of-speech tags in the cleaned version of release 6 of the CGN corpus (611,136 sentences, 5,983,309 tokens). The number of subtags shows how many different sets of attributes appeared with each tag.

data (files of which the name ended with 01, 59,869 tokens), test data (files of which the name ended with 69, 60,000 tokens) and training data (all other files, 5,863,440 tokens). The development data and test data sets have been chosen in such a way that they contain about 1% of all available material.

Because of the large number of tokens in the training data, running a single memory-based learning experiment requires many hours. This makes it difficult to perform even small-scale feature selection. Therefore we have searched for the best set of features for a subset of the training data only. In our first experiments we started with the first 0.2 million tokens of training data from zero features. We used MBT in combination with bidirectional hill-climbing and searched for the set of features that obtained the highest accuracy. The best set of features found was used as a starting point for a feature search with 0.5 million tokens. Thus we found another best set which was used as a starting point for a series of experiments with one million tokens. Finally, the best features found were applied to the complete training data.

This approach did not work very well. We observed a constant improvement of the best accuracy: 89.96%, 91,59%, 92.79% and 94.61%. However, most of the performance increase was caused by the fact that each new set of experiments had access to more training data. For example, the feature selection approach with 0.5 million words of training data started with an accuracy of 91.46 for the best feature set. This means that the extra training data has added 1.50% to the accuracy and the feature selection method only 0.13. When we look at the performance of TnT on the complete training data, it becomes clear that the overall performance (94.61%) should have been better: TnT obtains 96.85%.

In order to improve the performance of the memory-based learner we have checked a number of different promising start locations in the feature space. We have evaluated the performance with using one, two and three tags to the left and right of both known and unknown

method	known	unknown	accuracy	comments
baseline	-	-	87.20%	
TnT	-	-	95.08%	
MBT	fW	Fsss	88.33%	context 0
MBT	dfWa	Fsss	93.33%	known, context 1
MBT	ddfWaa	Fsss	93.22%	known, context 2
MBT	dddfWaaa	Fsss	93.12%	known, context 3
MBT	fW	dFsssa	88.59%	unknown, context 1
MBT	fW	ddFsssaa	88.56%	unknown, context 2
MBT	fW	dddFsssaaa	88.53%	unknown, context 3
MBT	dfWa	dFsssa	93.59%	best start pair
MBT	dfWaw	chssswdFw	94.05%	feature selection

Table 2: Accuracies obtained for part-of-speech tagging on the development data with the first million words of training data.

method	unique	ambiguous	unknown	all
baseline	98.59%	89.82%	22.46%	90.10%
TnT	98.59%	96.82%	81.10%	96.85%
MBT	98.62%	96.55%	71.24%	96.52%

Table 3: Accuracies obtained for part-of-speech tagging on the development data with the complete set of training data. 11% of the tokens in the test had a unique tag in the training data, 88% contained more than one tag and 1% did not occur in the training data. Overall results that are more than 0.12 apart are significantly different.

words for a training data set of one million tokens. A context of one tag proved to be the best, both for known as for unknown words. After this we have started a feature selection process from this feature set and applied the best feature set found to the complete data set. The results can be found in Tables 2 and 3.

The baseline method outputs the most frequent tag of each word. Ties between two tags are resolved by selecting the most frequent of the two in the complete training data. Unknown words receive the most frequent tag of the least frequent words in the training data. Of the seven context sizes we have checked, a context of one proved to be best for both known and unknown words. We have used this as a starting point for a feature selection process which improved the accuracy on the development data from 93.51% to 94.05%. We have applied the best feature set to the complete training set. The results can be found in Table 3. Both TnT and MBT perform better than the baseline. TnT performs significantly better than MBT but the absolute difference is small.

Apart from the experiments described in Table 2, we have also evaluated some of the parameters of the basic memory-based learner. Examining a larger neighborhood (parameter k) resulted in better performances with one million words of training data (accuracy 94.42%) but this requires large feature sets which are impractical to use with all training data. We have evaluated an alternative memory-based algorithm which is faster in classification

method	training	known	unknown	accuracy
baseline	all	-	-	98.22%
TnT	5,400k words	-	-	98.88%
MBT	200k words	wdf	sF	94.81%
MBT	500k words	wf	ssFaa	95.53%
MBT	1,000k words	wfa	ssssF	96.24%
MBT	all	wfa	ssssF	98.51%

Table 4: Lemmatization results obtained on the development data.

(IGTREE). However this algorithm performed worse than IB1IG (accuracy 93.75%).

4.2 Lemmatization

Lemmatization involves finding the base form of a word. This task is usually applied to dictionaries in which case a word can have more than one base form. We have applied the task to running text and we assume that in that case words can have only one base form. The base form depends on the characters in the word. Defining the task as a character transformation task is quite complex and therefore we have defined it as a word tagging task.

It is quite unlikely that a classifier will learn any generalizations in such a word-to-word transformation task. Therefore we have defined the task as a word-to-pattern transformation task. The patterns define how the word needs to be changed in order to obtain the base form. Example: *toegekend* receives pattern `-toege-d+toe+Len` which means that the base form can be obtained by removing *toege* from the start of the word and *d* from the end, and then adding *toe* to the front of the word and appending the final character followed by *en*. This produces the base form *toekennen*. We have defined 74 basic patterns. These were combined with a list of 27 prepositions to create a total of about 17,000 patterns. We have used the same segments of the CGN data as in the part-of-speech tagging experiment as training, development and test data.

We have searched for the best MBT features in the same way as in our initial part-of-speech tagging experiments: starting with small training data sets and increasing these step-by-step. The results can be found in Table 4[5]. The final performance is quite high (98.51%). However, the small difference with the baseline and the score obtained by TnT[6] suggest that there is room for improvement. The features used by the learner (only final characters, no initial characters) indicates that it will have difficulties with processing unknown words with an initial verb particle.

[5]Probably further work on lemmatization will boost performance but since the system performs well on the task we need for summarization, finding base forms for frequent verbs, we have not devoted extra time to it.

[6]TnT crashed when trained with the omplete data set (temporary lexicon entry (tags) overflow) and therefore we have applied it to part of the training data only.

phrases	tokens		length	name
24,701	24,701	5%	1.0	clause boundary (CLB)
5,674	13,066	3%	2.3	multi-word unit (MWU)
123,109	169,605	37%	1.4	noun phrase (NP)
28,491	29,245	6%	1.0	prepositional phrase (PP)
77,979	77,979	17%	1.0	verb phrase (VP)
-	146,813	32%	-	none

Table 5: The five phrase types in the syntactically annotated part of release 6 of the CGN corpus (51,416 sentences, 461,409 tokens).

4.3 Text chunking

Text chunking involves dividing sentences in base phrases containing syntactically related words. For example, the sentence *De bonden eisen meer duidelijkheid over Ford Genk .* can be divided as follows:

> [NP De bonden NP] [VP eisen VP] [NP meer duidelijkheid NP] [PP over PP] [MWU Ford Genk MWU] .

This sentence contains two noun phrases (NP), one verb phrase (VP), one prepositional phrase (PP) and one multi-word unit (MWU). The punctuation sign is not part of a phrase. No embedded phrases are allowed and therefore the prepositional phrase only contains a preposition rather than a preposition and a noun phrase. We evaluate the output of text chunkers by counting the number of completely correct phrases and dividing them by the number of phrases found by the system (precision) and by the number of phrases in the text (recall), and computing the harmonic mean of these two ($F_{\beta=1}$).

Building a text chunker from the syntactically annotated CGN material is not a simple task. In section 3.1, we have shown that the annotation scheme is incomplete. Furthermore, the presence of discontinuous phrases, made visible by crossing brackets in the syntactic trees, complicates the task of the chunker. Indeed our first naive approach to this task: predict the annotation level immediately above the part-of-speech tags, did not work well. At best we obtained $F_{\beta=1}$=74 while we have obtained $F_{\beta=1}$=92 for English data (Tjong Kim Sang, 2002b). The precision of that chunker (74) was so low that it was unlikely that its output would be of use to the summarizer.

In order to obtain useful annotation, we have simplified the task. Instead of attempting to predict the 25 different syntactic tags which we found in CGN, we have restricted ourself to three most frequent phrase types: noun phrases, verb phrases and prepositional phrases. Some person names have been marked up as multi-word units and therefore we have included these as well. It was relatively easy to extract words that marked the start of a relative clause and therefore we included a phrase type for these words as well.

method	data	known	unknown	precision	recall	$F_{\beta=1}$
baseline	words	-	-	82.74%	86.11%	84.39
TnT	words	-	-	87.72%	89.59%	88.64
MBT	words	wdfWaw	csswddFw	89.68%	90.35%	90.01
baseline	POS	-	-	82.74%	86.11%	84.39
TnT	POS	-	-	90.34%	91.30%	90.82
MBT	POS	wdfWw	wF	92.62%	93.03%	92.83

Table 6: The performance of MBT and TnT on the text chunking task. $F_{\beta=1}$ scores that are more that 0.30 apart are significantly different. The best performance is obtained with part-of-speech tags as input. MBT performs well for verb phrases ($F_{\beta=1}$=96) but multi-word units proved to be difficult ($F_{\beta=1}$=46).

Because the syntactic annotation in CGN was incomplete for our task, we had to produce some additional information. Nearly all pronouns have been put in noun phrases of their own except for pronouns that marked the start of a relative clause. Nouns not already included in noun phrases were put in a noun phrase of their own. All verbs and verb particles were stored in verb phrases. Each separate tree was regarded as a separate sentence. We collected 51,416 sentences of syntactic annotated material which contained 461,409 tokens (see Table 5). This material was divided in three sections: development data (file names ending with 2, 45,533 tokens), test data (file names ending with 9, 46,853 tokens) and training data (all other files, 369,023 tokens).

We have performed two feature selection experiments with this data. In the first, the learner was presented with words and in the second with part-of-speech tags. Both the part-of-speech tags of the training and development data were derived by the TnT tagger[7]. The results of these experiments can be found in Table 6. MBT performs better with part-of-speech tags as input than with words as input. For both input variants it outperformed TnT in this task.

4.4 Relation finding

The CGN material contains useful relation information like subjects, head verbs, objects, modifiers and complements. Automatically identifying these relations will be a hard task. We have attempted to simplify the task by looking only for two types of phrases: head verbs and subjects of main clauses and relative clauses. The example sentence of the previous section will be annotated as:

[SU De bonden SU] [HD eisen HD] meer duidelijkheid over Ford Genk .

We worked with the same data sets as in the text chunking task. Only the subjects and heads of phrases of the types SMAIN, SSUB, SV1 and PPRES have been used. The phrases

[7] At the time when this experiment was conducted MBT did not yet generate good part-of-speech tags.

phrases	tokens		length	name
52,672	52,734	11%	1.0	clause head (HD)
50,400	65,172	14%	1.3	clause subject (SU)
-	343,503	75%	-	none

Table 7: The two phrase types used in the relation finding task with release 6 of the CGN corpus (51,416 sentences, 461,409 tokens).

method	data	known	unknown	precision	recall	$F_{\beta=1}$
baseline	words	-	-	87.77%	76.08%	81.51
TnT	words	-	-	87.64%	85.88%	86.75
MBT	words	wddfWaa	chwddddFaw	91.07%	89.69%	90.38
baseline	POS	-	-	89.16%	88.29%	88.72
TnT	POS	-	-	88.78%	88.20%	88.49
MBT	POS	wddfWaa	pdFa	92.95%	92.51%	92.73
baseline	chunks	-	-	68.03%	50.78%	58.15
TnT	chunks	-	-	67.77%	81.08%	73.83
MBT	chunks	wfw	nF	77.78%	83.08%	80.34

Table 8: The performance of MBT and TnT on the relation finding task. The best performance is obtained with part-of-speech tags as input. MBT does well in finding clause heads ($F_{\beta=1}$=97) but retrieving clause subjects is more difficult ($F_{\beta=1}$=88).

have been extracted from the corpus in a straight-forward way; there were no exceptional cases. We have run three sets of feature selection experiments with different input types: one with words, one with part-of-speech tags and one with text chunks. The part-of-speech tags have been generated by TnT and the chunks by the best MBT text chunker described in the previous section. The results of the experiments can be found in Table 8. MBT outperformed TnT for all three input types. The best results were obtained with part-of-speech tags as input.

4.5 Person name finding

Identifying person names in text is important for summarization because in the target text, news wire text, first names can often be removed. In this task we have used the data set of the Dutch named entity recognition task of CoNLL-2002 (Tjong Kim Sang, 2002a). We have only used the person names tags of this data and removed the tags for locations, organizations and miscellaneous entities. A complication with this data set is that all names have been tagged as PERSON while we need to keep apart first names and surnames. We have manually added the required information to the training data. The first 90% of the data (182,796 tokens) was used as training data and the final 10% (20,135 tokens) as test data.

It did not seem sensible to perform person name finding on part-of-speech tags or text

phrases	tokens		length	name
2,752	2,823	1%	1.0	first names (FIRST)
4,285	4,751	2%	1.1	surnames (SUR)
-	195,357	97%	-	none

Table 9: The two phrase types used in the person name finding task in the data constructed from the CoNLL-2002 shared task for Dutch named-entity extraction (16,093 sentences, 202,931 tokens).

method	data	known	unknown	precision	recall	$F_{\beta=1}$
baseline	words	-	-	79.25%	36.92%	50.37
TnT	words	-	-	68.48%	69.23%	68.85
MBT	words	wddd fWa	cwddFaww	76.03%	69.01%	72.35
baseline	words+list	-	-	69.85%	80.44%	74.77
TnT	words+list	-	-	48.74%	89.23%	63.04
MBT	words+list	wdddddfaaww	nF	83.75%	88.35%	85.99

Table 10: The performance of MBT and TnT on the person name finding task. The best performance is obtained with words and a list of person names as training data. MBT performed better on recognizing first names ($F_{\beta=1}$=89) than on identifying surnames ($F_{\beta=1}$=83).

chunks and therefore we have only performed a feature selection process with words as input. The results were not very good ($F_{\beta=1}$=72). This is probably caused by the small number of positive training examples: Table 9 shows that we had only about 7,000 examples of person names. In all the previous tasks we had tens of thousands or even hundreds of thousands examples of the phrases we were looking for.

We have attempted to improve performance by creating extra training data. We applied the person name finder to an unannotated corpus (material from the Belgian magazine Knack) and selected sentences which contained phrases annotated as names which were not already present in the training corpus. We manually checked and corrected the sentences and added these to the training data. We stopped after adding about 40,000 tokens with about 4,000 positive phrases. By then performance had increased to $F_{\beta=1}$=74. It seems that a lot more data was needed in order to obtain reasonable results.

Our next improvement attempt consisted of adding a list of names to the training data. Again we stumbled upon the problem that there was not distinction between first names and surnames in the list. We have classified the names by guessing that a word is a first name if it appears more often on the first position of a multi-word name than on any non-initial position. This approach converted a list with 35,225 person names into one with 22,343 first names and 31,882 surnames. This list was added to the training data. As a result performance improved to $F_{\beta=1}$=86. Probably it is possible to improve performance even further but the current performance is good enough for aiding the summarization process.

method	data	known	unknown	precision	recall	$F_{\beta=1}$
baseline	words	-	-	69.20%	8.29%	14.80
TnT	words	-	-	55.83%	15.17%	23.86
MBT	words	wddfWaw	hpssssdddFaw	52.96%	32.39%	40.20

Table 11: The performance of MBT and TnT on the punctuation task for CGN data. Training data consisted of half a million words. MBT obtained $F_{\beta=1}$ rates between 45 and 50 for periods and empty sentence endings but the performance for question marks and ellipses was quite bad ($F_{\beta=1}$ less than 6).

4.6 Capitalization and punctuation

At this moment we are working with text data. In order to be able to process speech data in the future, we have looked at two additional prepossessing tasks which can aid an automatic analysis of speech data. The first is capitalization: the task of finding out in whether a word contains capital characters or not. This function is useful for name recognition as well. By changing the initial character of a written sentence to lower case whenever appropriate, one can reduce the number of errors made by a named-entity system.

The method we have applied for the capitalization task is an adapted version of the method by Michael Collins mentioned in (Curran and Clark, 2003): find out how often a word appears with capital characters and in lower case by examining a large corpus. This method works quite well but it has problems with phrases like *New York*. Therefore we have collected bigram information as well: a set of words which usually appear in lower case except when they are preceded by or followed by another specific capitalized word. We use this method for standardizing capitalization versions in the summarization experiments that will be discussed in the next chapter. The capitalization method has not been evaluated yet.

Another task which we need to perform is punctuation insertion: finding out where sentences start and end, what punctuation sign to insert at those places, and where to insert commas and other sentence-internal punctuation signs. It is relatively easy to obtain training material for this task since it can be generated from any well-formatted text. For a start, we have attempted to reproduce the CGN punctuation information. We have selected about half a million words of CGN material as training data and about 50,000 tokens as test material. All punctuation signs and sentence boundaries were removed from the data. The task was to find back the sentence boundaries (8%) and insert a period (86%), question mark (8%), ellipsis (5%) or nothing (1%). The performance of the learner was measured with precision, recall and $F_{\beta=1}$ rates.

We have performed one feature selection experiment with MBT. The results can be found in Table 11. MBT is right in suggesting the position of a sentence break in about half of the cases but it finds back less than a third of these places. Punctuation insertion proves to be a difficult task. It seems that additional syntactic annotation could be helpful but in a normal setup one would expect the punctuation process to precede the syntactic analysis. Some extra work is required on this task.

tokens		name
18,084	20%	delete
3,967	4%	replace
70,824	76%	copy

Table 12: The three main token actions that are performed in the summarization process in the selected part of the VRT news corpus (7,603 sentences, 83,238 tokens). The replacement action specifies the replacing tokens as well (1,315 different tokens).

5 Using Syntactic Annotation

With the syntactic annotation tools described in the previous chapter, we have obtained the measures to enrich the parallel Atranos subtitling corpus. In this chapter we will describe our experiments with this syntactically annotated material. We will work with the VRT corpus. The current corpus contains 431,190 tokens. Not all this material is useful for extracting summarization information. Many subtitles are exact copies of their corresponding autocue sentence. In some sentence pairs there have been so many changes that it will be hard to find correspondences between the sentences.

We have selected the most interesting sentence pairs from the corpus. We have defined a sentence pair as interesting when the two sentences are different but share at least half of the words. The alignment between sentences has been manually checked but the correspondences between words have been estimated by an algorithm and have not been checked. The selection process produced 7,603 sentences from the January, February and March sections of 2002. The small December 2001 section has been kept aside as future test material. The words in the selected sentences have been converted to default case and all punctuation signs have been removed. In total 92,875 tokens remained. All sentences in the corpus received part-of-speech tags, lemmas, chunk tags, relation tags and person name tags. We have kept the first 83,238 tokens as training material and used the remaining 9,637 tokens (about 10%) as test material.

The summarization process has been defined as a tagging process. The summarizer can perform three actions for each token: copy it, delete it or replace it (see Table 12). Inserted tokens have been ignored. Here is an example sentence from the corpus:

de politici vinden de euro natuurlijk/DELETE een goeie/goede zaak

Two tokens need to be changed. The adverb *natuurlijk* needs to be deleted and the adjective *goeie* needs to be replaced by *goede*. We evaluate the summarization process by counting the total number of predicted deletions and replacements that have been predicted correctly and dividing them by the total number of deletions and predictions that were predicted (precision) and those that are present in the corpus (recall). We combine these measures by computing the harmonic mean of the two ($F_{\beta=1}$).

We started with performing two MBT feature selection experiments: one with only words

method	data	known	unknown	comp.r.	precision	recall	$F_{\beta=1}$
baseline	words	-	-	96%	53.30%	13.61%	21.69
TnT	words	-	-	96%	43.49%	8.99%	14.90
MBT	words	wwfWaa	hnFaa	90%	44.80%	24.26%	31.47
baseline	POS	-	-	100%	64.94%	2.22%	4.29
TnT	POS	-	-	100%	55.88%	1.69%	3.27
MBT	POS	wwwwfWaaaaw	Fww	85%	33.59%	25.06%	28.70
TiMBL	all	w_{-3}–w_{+1},l_0,p_{-1}–p_{+1},n_0–n_{+1}		86%	41.24%	30.29%	34.93
Rules	all			51%	23.24%	49.09%	31.55

Table 13: The performance of MBT and TnT on the summarization task. The parameters used by the TiMBL process are words (w), lemmas (l), part-of-speech tags (p) and person name information (n). A subscript following a character indicates its position with respect to the current token. The target overall word compression rate is 81%. $F_{\beta=1}$ rate differences that are larger than 1.83 are significant.

as input and one with only part-of-speech tags. The results can be found in Table 13. MBT with words as input performed better than with part-of-speech tags as input. The overall performance is not very high. Especially the low precision is worrying: even with the best features, more than half of the predicted deletions and insertions of MBT are wrong according to the data. We have performed experiments which optimized precision rather than $F_{\beta=1}$. These improved precision to just over 50% but as a size effect the recall dropped under 10%.

After the MBT experiments we performed a feature selection experiment with the general memory-based learning which is the base of this tagger: TiMBL. For each token, the system had access to a window of three previous and three following tokens, lemmas, part-of-speech tags, chunk tags, relation tags and person name information. After a feature selection process which started from zero features, the system selected five word features, three part-of-speech features, two person name features and a lemma feature. The relation and the chunk features were not used. With the eleven features, TiMBL obtained an $F_{\beta=1}$ rate of almost 35, which is about 3.5 points better than MBT (see Table 13).

We have compared the performance of the learning systems with a set of hand-crafted deletion rules. The rules use information about the words, their lemmas, part-of-speech tags, chunk tags and person name tags but not the relation information. The rule set contains deletion rules for adjectives, adverbs, first names, prepositional phrases, phrases between commas, phrases between brackets, relative clauses, numbers in certain positions and time phrases. Currently there is no limit on the application of the rules which means that they delete every word of which they assume that it can be deleted. Therefore the compression rate of the rule set is higher than that of the learning systems.

The rules do not obtain a higher $F_{\beta=1}$ rate than TiMBL. We can conclude that, measured over $F_{\beta=1}$ rates, the syntactic annotation has resulted in a significant improvement (from 31.47 to 35.93, with an improved word compression rate). However, it remains to be seen whether the current performance of the summarizer is good enough to be used in a practical

environment.

6 Concluding remarks and future work

We have presented four syntactic analysis tasks for Dutch (part-of-speech tagging, lemmatizing, text chunking and relation finding) and three additional analysis tasks (person name finding, capitalization and punctuation insertion) which are useful for converting sentences to summarized subtitles. Application of the syntactic information resulted in a significant improvement of the learning summarizer. However, the current performance of the summarizers is probably not good enough for them to be used in practice.

Although there is room for improvement in each of the syntactic modules, it seems that there is not much extra to be gained in summarization performance by improving the first three of them. An additional summarization performance boost can probably obtained by more elaborate relation finding and an extra module for identifying verb particles. The initial versions of capitalization and punctuation modules need some further development as well. An important feature which needs to be added to both summarizers is an opportunity for the user to select different compression rates. We expect that this option will improve the quality of the summarizations as well.

Automatic evaluation of the summarization processes remains a problem. We suggest to combine three measures: $F_{\beta=1}$ rates for deletions and replacements, character compression rate (both overall totals as percentage of successful sentences) and a basic grammatical analysis which checks for grammatical errors which are observed frequently in summarized sentences. The implementation of this evaluation scheme should be one of the main tasks in the next project year.

References

Brants, T. (2000). *TnT – A Statistical Part-Of-Speech tagger.* Saarland University.

Caruana, R. and Freitag, D. (1994). Greedy attribute selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36. New Brunswick, NJ, USA, Morgan Kaufman.

Curran, J. R. and Clark, S. (2003). Language independent ner using a maximum entropy tagger. In *Proceedings of CoNLL-2003.*

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2002a). *MBT: Memory-Based Tagger, version 1.0, Reference Guide.* ILK Technical Report ILK-0209, University of Tilburg, The Netherlands. http://ilk.kub.nl/.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2002b). *TiMBL: Tilburg Memory Based Learner, version 4.3, Reference Guide.* ILK Technical Report ILK-02-10. http://ilk.kub.nl/.

Moortgat, M., Schuurman, I., and van der Wouden, T. (2002). *CGN Syntactische Annotatie.* Progress report Spoken Dutch Project (in Dutch).

Tjong Kim Sang, E. F. (2002a). Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

Tjong Kim Sang, E. F. (2002b). Memory-based shallow parsing. *Journal of Machine Learning Research*, 2(Mar):559–594.

Tjong Kim Sang, E. F. (2003). *A Baseline Subtitle Generator, – ATraNoS WP4-09 & WP4-10 –.* Internal ATraNoS progress report.

Van der Beek, L., Bouma, G., Daciuk, J., Gaustad, T., Malouf, R., Van Noord, G., Prins, R., and Villada, B. (2002). *Algorithms for Linguistic Processing.* NWO PIONIER Progress report.

A Examples

This section contains some example output texts of the current versions of the summarizers. The example text has been taken from the start of the VRT 19:00 news broadcast of Sunday December 16, 2001. It is neither part of the training data nor of the development data of the memory-based summarizer. Here is the output of the TiMBL summarizer for this text with deletions enclosed in square brackets and replacements and movements underlined:

- welkom bij het zevenuurjournaal(zeven uurjournaal) .
- in Afghanistan veroveren anti-Talibantroepen de tunnels van Tora Bora , maar van [Osama] Bin Laden is geen spoor .
- [Yasser] Arafat roept de Palestijnse extremisten op [de] aanslagen tegen Israël te stoppen .
- [en] de Belgen grijpen naast de medailles op de Europese zwemkampioenschappen in Antwerpen .

- in Afghanistan hebben anti-Talibanstrijders de tunnels en grotten van Tora de VS(Bora) veroverd , het laatste bolwerk van terroristenleider [Osama] bin Laden .
- vanmiddag kwam één van de commandanten van de [alliantie] [terug] van het front , met goed nieuws en slecht nieuws .
- het goede was : Tora Bora is gevallen en Al Qaeda is verslagen .

TiMBL generates a perfect summary for the first sentence: two words are combined into a compound. The sentence is the first of every news broadcast and thus the summary could be found in the training data. The next four deletions are correct: two first names, an article preceding a plural noun and a sentence-initial conjunction. Replacing the unknown word *Bora* by *de VS* is obviously incorrect. The deleted first name and the adverb are both correct but removing the noun *alliantie* results in an ungrammatical sentence.

Here is the output of the rule-based summarizer:

- welkom [bij het zeven uurjournaal] .
- [in Afghanistan] Anti-Talibantroepen veroveren de tunnels [van Tora Bora] [[, maar [van [Osama] bin Laden] is geen spoor]] .
- [Yasser] Arafat roept de [Palestijnse] extremisten [op de aanslagen] [tegen Israël] te stoppen .
- En de Belgen grijpen [naast de medailles] [op de [Europese] zwemkampioenschappen] [in Antwerpen] .
- [in Afghanistan] Anti-Talibanstrijders hebben de tunnels [en grotten] [van Tora Bora] veroverd [, het laatste bolwerk [van terroristenleider] [Osama] bin Laden] .
- [vanmiddag] één [van de commandanten] [van de Alliantie] kwam [terug] [van het front] [, [met goed nieuws] en [slecht] nieuws] .
- Het goede was : Tora Bora is gevallen en [Al] Qaeda is verslagen .

The first two sentences are fine although not much is left of the original sentences. In the second sentence the subject of the sentence is moved to the sentence-initial position, a feature of the rule-based approach that is not present in the learning approach yet. The third sentence contains a familiar error: *op* is classified as a preposition rather than a verb particle. Probably this is a problem which can be solved but it requires detailed information about the different Dutch verbs. Sentence four contains another problem with an obligatory verb particle: *naast*. The next sentence is to short: the multiword unit *één van de* should probably not have been split and *terug* should not have been removed (part-of-speech tagging error?). The final sentence has a problem with the word *Al* which is a frequent adverb in Dutch. It is mistagged and removed, which should not have happened.

The rule-based summarizer is available for testing at http://cnts.uia.ac.be/cgi-bin/atranos