A Memory-Based Shallow Parser for Spoken Dutch

Sander Canisius and Antal van den Bosch

ILK / Computational Linguistics and AI, Tilburg University

Abstract

We describe the development of a Dutch memory-based shallow parser. The availability of large treebanks for Dutch, such as the one provided by the Spoken Dutch Corpus, allows memory-based learners to be trained on examples of shallow parsing taken from the treebank, and act as a shallow parser after training. An overview is given of a modular memory-based learning approach to shallow parsing, composed of a part-of-speech tagger–chunker and two grammatical relation finders, which has originally been developed for English. This approach is applied to the syntactically annotated part of the Spoken Dutch Corpus to construct a Dutch shallow parser. From the generalisation scores of the parser we conclude that existing memory-based parsing approaches can be applied to spoken Dutch successfully, but that there is room for improvement in the tagger–chunker.

1 Introduction

Applying memory-based learning to shallow parsing of natural language provides a useful alternative to manual grammar-based parser development. Being dependent on the availability of suitable training data, however, much research in this area has only targeted the English language. The increasing availability of corpora for other languages in recent years now opens up the possibility for testing the generality of memory-based shallow parsing techniques.

The Spoken Dutch Corpus (Dutch: “Corpus Gesproken Nederlands”, or CGN) includes a large treebank of contemporary spoken Dutch. It thereby provides the necessary resources to train a memory-based learning approach to perform shallow parsing of Dutch. In this paper, an existing memory-based shallow parsing method, which has been shown to be effective for English, is applied to the CGN data to construct a shallow parser for spoken Dutch.

The remainder of this paper is structured as follows. Section 2 describes the architecture of the parser and the learning task representation. Next, in Section 3, a general overview of memory-based learning, the machine learning approach used to construct the shallow parser, is given. Section 4 presents the CGN and reviews the relevant parts of its syntactic annotation. Finally, in Section 5, the performance of the system is evaluated, after which conclusions are drawn in Section 6.

2 Architecture and task representation

In the memory-based shallow parsing framework, the parsing task is split up into consecutive subtasks, each of which is learnt by separate classifiers. Applying these classifiers sequentially to the input data, more and more information becomes available, which can then be used to enrich the input for following classifier modules. Following Daelemans, Buchholz and Veenstra (1999), the parser de-

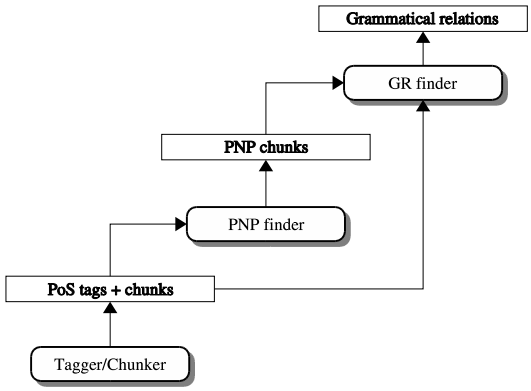

Figure 1: Architecture of the memory-based shallow parsing framework. The rounded rectangles represent the three parser modules; the normal rectangles and the arrows depict the information flow between the modules.

scribed in this paper distinguishes four subtasks: part of speech tagging, chunking, PNP finding, and grammatical relation finding. As can be seen in Figure 1, the four subtasks are performed by three modules; one module that performs both part of speech tagging and chunking at the same time, and two modules performing the remaining two relation-finding subtasks.

As a result of dividing the parsing process into smaller subtasks, the tasks to be performed are reduced to a level that allows memory-based classifiers to be trained for them straightforwardly. Joining the modules performing these subtasks in sequence leads to a system that performs the overall parsing task. The remainder of this section describes the learning tasks corresponding to each of the three parser modules.

2.1 Tagging–chunking

The first module in the parsing cascade combines the part of speech tagging and chunking steps into a single learning task, that is, words are assigned a fully disambiguated word class and at the same time joined in non-recursive, non-overlapping constituents referred to as chunks (Abney 1991). In contrast, the memory-based chunker described by Veenstra (1999) uses feature vectors that include features for the parts of speech of words, and therefore depends on a separate part of speech tagger to predict those parts of speech. The single-module approach adopted in this paper was introduced by Buchholz (2002), who shows that the two subtasks can be combined without loss of performance.

The memory-based learning task for the tagger–chunker module uses the chunking-as-tagging method described by Ramshaw and Marcus (1995): for each word in the sentence, an instance is generated, which is then assigned a chunk tag

denoting whether the word is inside a chunk (`I`), inside a chunk, but not in the same chunk as the word directly preceding it (`B`), or outside any chunk (`O`). An additional chunk type is appended to this tag expressing the type of chunk, if any, the word is in. For example, `I-NP` means the word is inside a noun phrase (NP).

The English memory-based chunkers use a definition of chunks that guarantees that the head word of a chunk is always its last word. Therefore, predicting a chunk automatically predicts the corresponding head word, as well. In contrast, the definition of chunks used in this paper is derived from the CGN annotation, which explicitly marks head words. These head words are often, but not by definition, the last word of a chunk. For this reason, the IOB-style chunk tags have been extended to predict not only the chunk for a word, but also whether the word is the head of its chunk. To deal with the situation when no word in a chunk is assigned the head role, or when multiple heads are predicted for a single chunk, the rightmost word that has been assigned a head tag, or if there is no such word, the rightmost word overall, is taken to be the head word.

Given the tagging scheme for chunks as described above, and part of speech tags taken from the training corpus as such, the target classes for the combined tagging–chunking task are a concatenation of a part of speech tag and a chunk tag. As an example, consider the following sentence.

(1) ik stond net op de kruising van de Postweg en de Groenewoudseweg.
I stood just at the cross-roads of the Postweg and the Groenewoudseweg.
'I just stood at the cross-roads of Postweg and Groenewoudseweg.'

The correct assignment of tags for this sentence is the following.

```
ik/VNW1-I-NP-HD stond/WW1-I-SMAIN-HD net/BW-I-ADVP-HD
op/VZ1-I-PP-HD de/LID-I-NP kruising/N1-I-NP-HD
van/VZ1-I-PP-HD de/LID-I-NP Postweg/N5-I-NP-HD en/VG1-O
de/LID-I-NP Groenewoudseweg/N5-I-NP-HD ./O-LET
```

This corresponds to the following chunk structure.

```
[NP ik/VNW1/HD ] [SMAIN stond/WW1/HD ] [ADVP net/BW/HD ]
[PP op/VZ1/HD ] [NP de/LID kruising/N1/HD ]
[PP van/VZ1/HD ] [NP de/LID Postweg/N5/HD ] en/VG1
[NP de/LID Groenewoudseweg/N5/HD ] ./LET
```

Instances for each word in a sentence are generated using the Memory-based tagger (MBT) (Daelemans, Zavrel, Berck and Gillis 1996). MBT generates its feature vectors by applying a windowing technique on the words in a sentence. Two types of features are used to describe the words in a window. One of them is the word form itself. The other is either the tag of the word or an ambiguous tag, describing all possible tags for the word. If a word precedes the focus word in the sentence, then MBT has already predicted a tag for this word (correctly or not); hence, this tag can be used in the instance description. In contrast, if a word follows the focus word in the sentence, then the correct tag for this word has yet to be predicted. As an alternative to the fully disambiguated tag for the word, then,

a symbol encoding all possible tags for this word according to the training data, is used.

MBT uses separate sub-taggers for known words and unknown words (i.e., words not seen in the training material). The known-word sub-tagger uses the features mentioned above. The unknown-word sub-tagger cannot make use of the word form of the focus word, because it does not match any known word in the training material. Instead, it uses word form features: prefix letters, suffix letters, and markers indicating whether the word is capitalised, contains a dash, or ends in numeric characters. A more detailed overview of the issues involved in constructing memory-based classifiers by MBT is given in Daelemans et al. (1996).

2.2 PNP finding

During the chunking step, prepositional phrases are not recognised as such. As sentences are only parsed into non-recursive, non-overlapping constituents, prepositional phrases are split up into a PP chunk and one or more chunks corresponding to its complements. However, finding prepositional phrases in sentences may be very useful, since they can contain information valuable to higher-level applications, for example information about times and locations.

An important subclass of prepositional phrases is formed by those having noun phrases as complements. The PNP finding task consists of reconstructing this type of prepositional phrases out of separate PP and NP chunks. These reconstructed prepositional phrases will be denoted as PNP chunks. As an example, applying the PNP finder to the same sentence as used in the previous section would lead to the following output.

```
[NP ik ] [SMAIN stond ] [ADVP net ] {PNP [PP op ] [NP de
kruising ] } {PNP [PP van ] [NP de Postweg ] en [NP de
Groenewoudseweg ] } .
```

Although the structure of PNP chunks is not identified when a sentence is divided into base chunks, the information to reconstruct them can be derived from the grammatical relations between the head preposition and the noun phrases making up its complement. The technique used by the PNP finder is therefore very similar to the one used by the grammatical relation finder, which is described in the next subsection. However, while the grammatical relation finder identifies relations between verbal chunks and other chunks, the PNP finder aims at those between PP chunks and NP chunks instead. Having found these relations, PNP chunks are constructed from a PP chunk and the NP chunks grammatically related to it.

Instances for the PNP finding task are centred around pairs of chunks. In PNP finding instances, such a pair corresponds to a PP chunk and an NP chunk, which is henceforth referred to as the focus chunk. For a given sentence, instances are generated for each pair of a PP chunk and an NP chunk following it within a certain distance. To encode the variable-length chunks by fixed-size feature vectors, the chunks are represented by their head words only. The heads are the most prominent words of chunks and therefore, grammatical relations could equally well be

Table 1: The first two PNP finder instances generated for the sentence "ik stond net op de kruising van de Postweg de Groenewoudseweg."

dist.	betw.	prep.	context-1			focus		context+1			
1	0	op	op	VZ1	PP	kruising	N1	van	VZ1	PP	+
3	1	op	van	VZ1	PP	Postweg	N5	en	VG1	-	-

considered to hold between head words of chunks, rather than between chunks as a whole. Apart from the word form, the features to describe the chunks are the part of speech of the head word and the chunk type.

In addition to the PP and focus chunks, instances include features to encode the local context of the focus chunk. This local context consists of a number of chunks directly preceding and a number of chunks directly following the focus chunk. Again, these chunks are represented by their head words, part of speech tags and chunk types. In short, the complete feature vector of PNP finding instances consists of the following features:

- The distance between the preposition and the focus chunk, counted in number of chunks or words outside chunks.
- The number of PP chunks between the preposition and the focus chunk.
- The preposition, that is, the head word of the PP chunk.
- The focus chunk, which is described by two features.
  - The head word of the focus chunk.
  - The part of speech of the head word.
- The context chunks, each described by three features.
  - The head word of the context chunk.
  - The part of speech of the head word.
  - The type of the context chunk.

The classification categories for the PNP finding task are kept simple. Instances are classified to decide whether a given preposition and focus chunk are to be joined in a PNP chunk. This is the case if the two are grammatically related. In contrast with the grammatical relation finding task, for PNP finding, the type of the grammatical relation is unimportant and consequently need not be predicted. PNP finding instances can be classified as one of either "+" or "-", thereby only indicating whether, not how, the preposition and focus chunk are related.

For the example sentence, three instances would be generated for the combination of "op" with each one of "kruising", "Postweg", and "Groenewoudseweg", and two more instances for "van" combined with "Postweg", and with "Groenewoudseweg". Examples of the first two instances that are generated, are given in Table 1.

2.3 Grammatical relation finding

The final step in the memory-based parsing cascade is grammatical relation finding. Having found the chunks of a sentence in the two preceding parsing steps, in this step, the parser determines their syntactic functions, for example subject or object of a verb, by identifying grammatical relations between pairs of chunks. Following Buchholz (2002), grammatical relation finding in this paper is restricted to relations to verbal chunks. Although this restriction excludes potentially useful relations, the central meaning of a sentence can often be retrieved quite well by only considering relations to verbs. Moreover, Buchholz (2002) speculates that finding relations to nouns or adjectives might require information different from that useful for finding relations to verbs and would therefore be better performed in separate tasks.

As was already mentioned in the section on PNP finding, PNP finding and grammatical relation finding are very similar and therefore their learning tasks are very similar too. With grammatical relation finding, however, the relations to be identified are not those between PP chunks and NP chunks, but consist of relations between verbal chunks and any other chunks. Furthermore, while the PNP finder only needs to determine whether two chunks are related, the grammatical relation finder should predict the type of the relation as well.

Machine learning instances for the grammatical relation finding task are centred around the head words of a verbal chunk and a focus chunk, which may be of any type. Additionally, instances contain features to encode the local context of the focus chunk. To find the grammatical relations in a sentence, instances are generated for each verbal chunk and focus chunk within a certain distance to the left or to the right. The classifications of these instances indicate whether and, if so, how the focus chunk is related to the verbal chunk.

More features than those mentioned above can be added to the instance description to improve classification performance. Buchholz (2002) specifically intends to identify features that are useful for memory-based grammatical relation finding. A notable result is the highly informative value that features coding sequences of part of speech tags or chunks, appear to have. In this paper, however, the simpler instance format used by Daelemans et al. (1999) is used. The features of this format are presented below.

- The distance between the verb and the focus chunk, counted in number of chunks or words outside chunks. A negative distance indicates that the focus chunk is to the left of the verb, a positive distance means the focus chunk is to the right of the verb.
- The number of verbal chunks between the verb and the focus chunk.
- The verb, that is, the head word of the verbal chunk.
- The part of speech of the verb.
- The focus chunk, which is described by four features.

Table 2: The first two grammatical relation finder instances generated for the sentence "ik stond net op de kruising van de Postweg de Groenewoudseweg."

d.	b.	verb		context-1			focus				context+1			
-1	0	stond	WW1	-	-	-	ik	VNW1	NP	-	stond	WW1	SMAIN	SU
2	0	stond	WW1	net	BW	ADVP	kruising	N1	PNP	op	G'seweg	N5	PNP	-

  - The prepositional head word, if the focus chunk is a PNP chunk, or "-" otherwise. This is the head word of the PP chunk within the PNP.
  - The head word of the focus chunk. If the focus chunk is a PNP chunk, this means the head word of the NP chunk within it.
  - The part of speech of the head word.
  - The type of the focus chunk.

- The context chunks, each described by three features.

  - The head word of the context chunk.
  - The part of speech of the head word.
  - The type of the context chunk.

The classification categories to be assigned to these instances correspond to the types of relations to be predicted and "-", to indicate that the given focus chunk is not related to the given verb. In the context of the research described in this paper, only subject and direct or indirect object are predicted. However, the task representation presented above does not restrict the number and types of relations that can be predicted. Table 2 shows two example instances in the format described in this section.

3 Memory-based learning

Memory-based learning denotes a class of machine learning algorithms that classify unseen instances by matching them to training instances stored in memory. IB1 (Aha, Kibler and Albert 1991) is a memory-based learner that uses a variant of the classic k-Nearest Neighbour algorithm (Cover and Hart 1967). It differs from normal k-NN in that it allows non-numeric features. For this paper, the IB1 implementation as provided by the TiMBL (Daelemans, Zavrel, Van der Sloot and Van den Bosch 2003) software package, has been used. This implementation extends standard IB1 with feature relevance weights (Daelemans and Van den Bosch 1992), distance-weighted nearest neighbour voting (Dudani 1976), and the MVDM distance metric (Cost and Salzberg 1993).

Instances are described by fixed-length vectors of feature-value pairs and a label denoting the target class the instance belongs to. At training time, IB1 simply stores all instances in its instance base. When a test instance is to be classified, this

instance base is searched for the instances that are most similar to the test instance. Similarity, in this respect, is defined as a distance between two instances. TiMBL uses the function in Equation (1) to calculate the distance between instances X and Y, both described by a feature vector of length n.

$$\Delta(X, Y) = \sum_{i=1}^{n} w_i \delta_i(x_i, y_i) \quad (1)$$

In this equation, distance is defined as the weighted sum of the distances per feature. w_i is a weight factor for feature i, and δ_i denotes a function for feature i that computes the distance per feature. Based on this distance function, the class for a test instance is determined by applying a (possibly distance-weighted) voting over the classes of the k nearest neighbours.

4 The Spoken Dutch Corpus

The CGN corpus (Oostdijk, Goedertier, Van Eynde, Boves, Martens, Moortgat and Baayen 2002) is the first corpus to provide a large-scale treebank for spoken Dutch[1]. CGN has been developed in the context of a project aiming to compose a corpus consisting of 1,000 hours of spoken standard Dutch. The data for the corpus are collected both in The Netherlands and in Belgium and contain a wide variety of different types of speech, such as spontaneous conversations, television broadcasts, read speech, and several others.

As the corpus is intended to be a useful resource for various different research interests, among which is speech and language technology, it includes a rich set of transcriptions and annotations. Of special interest to the research presented in this paper is the syntactically annotated part of the CGN. The CGN treebank provides a full dependency annotation structured very similarly to the NEGRA corpus (Skut, Krenn, Brants and Uszkoreit 1997). Syntactic annotations are encoded in the form of dependency trees. In these dependency trees, three kinds of information are stored: part of speech tags, phrasal categories and dependency relations.

The terminal nodes of a dependency tree correspond to the individual words in a sentence and are assigned a part of speech label. The complete tag set comprises 316 different part of speech categories (Van Eynde 2001). For the syntactic annotation, it is reduced to a set consisting of 72 different labels relevant for the annotation.

The non-terminal nodes in the annotation structure represent higher-level syntactic constituents, dependency domains in CGN terminology. They are assigned one of 25 phrasal category labels (Moortgat, Schuurman and Van der Wouden 2001). Generally, a domain has exactly one daughter that is referred to as the head of that domain. The other daughters complement or modify the head.

The final element of the CGN annotation is the dependency relation, which is stored on the edges of the annotation graph. A dependency relation between a daughter and a mother domain either identifies that daughter as the head of its

[1] Using the same annotation scheme, Alpino, a treebank for written Dutch has been produced in parallel with CGN (Van der Beek, Bouma, Malouf and Van Noord 2001).

Table 3: Instance statistics from the CGN treebank for each of the parser modules. Syntactic elements correspond to either chunks or relations depending on the parsing task.

Module	Instances	Syntactic elements
Tagger–Chunker	514,101	298,806 chunks
PNP Finder	41,139	23,186 chunks
Grammatical relation finder	194,790	64,266 relations

mother domain or expresses its syntactic function to that head. The CGN annotation uses 34 dependency labels.

The experiments reported upon in this paper have been based on release 6 of the CGN, which provides a treebank containing approximately half a million words. Chunks and grammatical relations have been extracted from this treebank by an algorithm based on that described by Buchholz (2002), and adapted for the CGN by Canisius (2004). Table 3 lists some statistics on the number of instances generated for each of the parser modules, and the number of chunks, or relations encoded by these instances.

5 Results

This section describes the results obtained with the Dutch memory-based shallow parser. First, some baseline scores for the three parsing subtasks are presented. These scores are an indication of the difficulty of the tasks, and can therefore serve as reference scores with which to compare the results reported on in this section. For tagging–chunking the baseline score (averaged over ten 10% heldout test set partitionings of the full data set) has been obtained by predicting the class which is most frequently encountered with a given word in the training data. The baseline experiment for both PNP finding and grammatical relation finding predicts the most frequent class given the distance between the focus chunk and the target verb or preposition. The resulting scores are listed in Table 4 in terms of accuracy (the percentage of correct classifications in test data), and for all tasks besides tagging, the precision (the ratio of correctly identified and labelled chunks or relations over all identified chunks or relations in test data), recall (the ratio of correctly identified and labelled chunks or relations over all chunks or relations actually occurring in test data), and $F_{\beta=1}$ (the harmonic mean of precision and recall) (Van Rijsbergen 1979). Note that, although in Table 4 and all following tables separate scores are listed for part of speech tagging and for chunking, a single module is responsible for both tagging and chunking.

In the remainder of this section, the experiments conducted with the shallow parser are discussed in two parts. First, the experimental optimisation of the memory-based learners is described. Then, the performance of the optimised memory-based shallow parser is presented.

Table 4: Baseline scores for the parsing subtasks.

	Accuracy	Precision	Recall	$F_{\beta=1}$
Tagging	89.26	-	-	-
Chunking	76.07	64.62	62.14	63.36
PNP finding	94.72	96.94	92.49	94.66
Grammatical relation finding	74.20	63.88	57.84	60.71

Optimisation of the modules To realise the full potential of the memory-based shallow parsing techniques, the parser modules have been optimised experimentally, before evaluating their performance. In a modular system, such as the one described in this paper, optimisation can take place on two levels. The first one is the level of the individual modules, that is, the modules can be optimised so that they perform optimally when tested in isolation. The second level is that of the global framework; on this level, the modules are optimised to perform well on input data that are generated by other modules, and that may therefore be noisy.

Although it is most likely that some extra performance can be gained by considering optimisation on the global level (Van den Bosch 1997), for this paper, the optimisation has been restricted to the level of the individual modules only. Module-level optimisation includes such strategies as algorithmic parameter setting optimisation, and feature selection. Either of those would be infeasible, if required to be performed optimally (e.g. by exhaustive search among all recombinations of feature selections and parameter settings). Here, only a limited feature selection has been performed, in which the size of the local context is varied, and a heuristic search for optimal parameter settings is performed.

The parameter setting optimisation procedure is best described as a heuristic hill-climbing search through the space of parameter settings. In brief, each of the parameters is, in turn, tested with all possible values, while keeping the other parameters fixed. The parameter value leading to the best performance is selected and left unchanged for the rest of the procedure. The parameters to be varied and their possible values are the following.

- Feature weighting
  - No weighting
  - Gain Ratio (Quinlan 1993)
  - Information Gain (Quinlan 1986)
  - Chi-squared (White and Liu 1994)
  - Shared Variance (White and Liu 1994)
- Distance metric
  - Overlap (Aha et al. 1991)

Table 5: Performance of the optimised parser modules when tested in isolation.

	Accuracy	Precision	Recall	$F_{\beta=1}$
Tagger	95.96	-	-	-
Chunker	87.99	83.91	85.91	84.89
PNP finder	97.68	97.17	97.82	97.50
Grammatical relation finder	92.92	90.56	85.94	88.19

  - Modified value difference (Stanfill and Waltz 1986, Cost and Salzberg 1993)
  - Numeric difference

- The number of nearest neighbours
  - Could be any integer value, however, only a limited number of them is actually tested.

- Nearest neighbour voting rule
  - Majority voting
  - Inverse distance (Dudani 1976)
  - Inverse linear (Dudani 1976)
  - Exponential decay (Shepard 1987)

This parameter optimisation has been performed for all three learning tasks. Training and test data covered the entire treebank provided by release 6 of the CGN. All experiments used 10-fold cross-validation, so that the choices in the optimisation procedure were statistically sound. The parameter settings for the three modules that were found lead to the scores listed in Table 5. It can be seen that all three memory-based parser modules outperform the baseline scores for the corresponding learning tasks by wide margins.

Performance of the parsing cascade The scores in Table 5 correspond to modules for which the input instances were constructed from gold-standard corpus data, that is, the PNP finder and the grammatical relation finder have been run on perfect part of speech and chunk information; likewise, the information about PNP chunks fed to the grammatical relation finder was directly extracted from corpus data. While such a setup is acceptable for the heuristic optimisation procedure, the actual performance of the entire shallow parser can only be measured when the three modules are applied in sequence to the input data.

Table 6 shows the scores obtained when constructing the input for the modules from the predicted outputs of the preceding modules. In addition to the performance scores on the entire corpus, the table also lists scores split out per component of the corpus. Each such component corresponds to a specific category of

Table 6: Performance of the parser modules as part of the memory-based shallow parsing cascade. The first row shows the scores on the entire corpus, the other rows show the scores on the various subcomponents of the corpus: a) spontaneous conversations , b) interviews, f) interviews/discussions/debates (broadcast), g) discussions/debates/meetings (non-broadcast), h) lessons recorded in a classroom, i) live commentaries (broadcast), j) news reports/reportages (broadcast), k) news (broadcast), l) commentaries/columns/reviews (broadcast), m) ceremonious speeches/sermons, n) lectures/seminars, o) read speech

	Tagger	Chunker			PNP finder			GR finder		
	Acc	Prec	Rec	$F_{\beta=1}$	Prec	Rec	$F_{\beta=1}$	Prec	Rec	$F_{\beta=1}$
all	95.96	83.91	85.91	84.94	85.98	89.79	87.84	83.76	79.11	81.37
a	96.07	83.42	85.79	84.59	85.11	88.90	86.97	85.38	82.27	83.79
b	95.79	83.66	85.64	84.64	85.84	90.20	87.97	84.08	79.04	81.48
f	95.07	84.30	86.03	85.15	85.97	90.68	88.26	81.92	77.51	79.66
g	94.71	82.32	84.01	83.16	85.40	88.19	86.77	76.92	68.39	72.40
h	95.20	83.72	84.53	84.13	89.66	91.07	90.36	83.14	76.38	79.62
i	94.74	79.93	84.15	81.98	86.68	90.24	88.43	80.31	74.30	77.19
j	94.58	82.83	84.32	83.57	84.77	90.14	87.37	77.85	62.03	69.05
k	94.96	84.81	85.80	85.30	86.46	91.70	89.00	76.92	62.32	68.85
l	93.12	82.26	83.17	82.71	84.10	83.31	83.71	75.50	67.62	71.34
m	92.36	78.24	80.53	79.37	73.44	84.21	78.46	72.46	64.94	68.49
n	93.28	81.88	83.97	82.91	82.78	86.62	84.65	80.73	74.08	77.26
o	93.74	83.95	84.48	84.21	86.41	90.17	88.25	77.46	67.27	72.00

speech data, for example, spontaneous conversations, television broadcasts, etc. The overall scores were obtained using ten-fold cross-validation; those for the individual components with leave-one-out cross-validation, so that even for small components the training set always contains a relatively large amount of similar data.

The scores reveal that the performance of the PNP finder and grammatical relation finder deteriorates considerably when their input is constructed from predicted rather than perfect tagging–chunking information. Clearly, errors committed by the tagger–chunker (especially the chunk part of the joint prediction) have a large effect on the performance of these two modules. Surprisingly, the scores on spontaneous conversations are among the highest for tagging, chunking and grammatical relation finding, whereas on components that would be expected to be less noisy, the scores are sometimes quite low. For example, the chunking score on ceremonious speeches is the lowest of all, and the grammatical relation finding score on read speech is also considerably lower than on spontaneous conversations.

In an attempt to improve the performance of the tagger–chunker, another experiment has been conducted to evaluate whether extending the parsing cascade with an additional module that removes disfluencies from the input data before the tagging–chunking step, could have a positive effect on the performance of the entire cascade. Lendvai, Van den Bosch and Krahmer (2003) show that disfluencies, which they define as chunks that do not fit under the syntactic tree of a sentence, can be predicted fairly reliably using only information that can be extracted from

Table 7: Performance of the parser modules as part of the memory-based shallow parsing cascade with input data from which disfluencies have been removed.

	Accuracy	Precision	Recall	$F_{\beta=1}$
Tagger	96.26	-	-	-
Chunker	89.00	85.50	86.61	86.05
PNP finder	90.18	88.04	92.39	90.17
Grammatical relation finder	90.78	85.66	80.48	82.99

raw text data. As a first step, the potential of disfluency filtering as part of the parsing cascade has been evaluated using *perfect* filtering of disfluencies, rather than applying a disfluency predictor for this purpose. This way, the scores obtained can serve as an estimate of the upper-bound of the performance to be expected when a real disfluency filtering module is inserted.

The results of this experiment are presented in Table 7. The increase of the performance scores with respect to the unfiltered system are significant for all modules (estimated by one-tailed paired t-tests with $p < 0.05$). However, it should also be noted that the performance gains are not too high either. This finding suggests that memory-based shallow parsing is already robust to a large extent with respect to the presence of disfluencies, and possibly noise in general, in the input data. Experiments with a real disfluency predictor should be conducted to decide whether the observed increase remains significant when the filtering of disfluencies is not perfect, and therefore whether there is any added value in disfluency filtering before memory-based chunking.

6 Conclusions

The availability of Dutch treebanks, such as the one produced within the CGN corpus project, enables memory-based learning techniques to be adopted to train a Dutch shallow parser. In this paper a memory-based shallow parsing method originally developed for written English is applied to the CGN data. In the first part of the paper, a summary of the various learning tasks forming the memory-based shallow parsing framework, has been given. These learning tasks did not require important changes in order to apply them to the CGN data, even though the CGN corpus adopts a stronger dependency-parsing orientation than the Penn Treebank (Marcus, Santorini and Marcinkiewicz 1993). This is due to the fact that the coding methods developed by Buchholz (Buchholz 2002) to produce instances for the tagger–chunker and the two relation finding modules, are designed to operate on any treebank that can be converted to a predefined intermediate format.

The results obtained by the Dutch shallow parser, on the one hand, suggest that the memory-based shallow parsing framework is both language-independent and robust to the kinds of noise typical of spoken language. On the other hand, however, an $F_{\beta=1}$ of 84.89 for the chunking component of the tagger–chunker

also leaves room for improvement. Given the fact that, in the parsing cascade, the performance of the tagger–chunker has a considerable effect on that of the other two modules, effort is probably best spent on trying to improve the tagging–chunking learning task representation.

A promising technique in this respect has been put forth by the results of the disfluency filtering experiment. It has been shown that there is some improvement to be made by the special handling of disfluencies in the input data. Moreover, in addition to simply removing them from the input data, other ways to deal with disfluencies might prove even more useful. Future research in the area of memory-based shallow parsing of spoken data should therefore pay special attention to the role of disfluencies.

Acknowledgements

This work is funded by NWO. Part of this study was conducted at the Universiteit Maastricht. The authors wish to thank Jaap van den Herik and Eric Postma for discussion and support and Sabine Buchholz, Walter Daelemans en Jorn Veenstra for their earlier work on memory-based shallow parsing.

References

Abney, S.(1991), Parsing by chunks, *Principle-Based Parsing*, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 257–278.

Aha, D. W., Kibler, D. and Albert, M.(1991), Instance-based learning algorithms, *Machine Learning* **6**, 37–66.

Buchholz, S.(2002), *Memory-Based Grammatical Relation Finding*, PhD thesis, Tilburg University.

Canisius, S.(2004), *Memory-based shallow parsing of spoken dutch*, Master's thesis, Universiteit Maastricht.

Cost, S. and Salzberg, S.(1993), A weighted nearest neighbour algorithm for learning with symbolic features, *Machine Learning* **10**, 57–78.

Cover, T. M. and Hart, P. E.(1967), Nearest neighbor pattern classification, *Institute of Electrical and Electronics Engineers Transactions on Information Theory* **13**, 21–27.

Daelemans, W. and Van den Bosch, A.(1992), Generalisation performance of backpropagation learning on a syllabification task, *in* M. F. J. Drossaers and A. Nijholt (eds), *Proceedings of TWLT3: Connectionism and Natural Language Processing*, Twente University, Enschede, pp. 27–37.

Daelemans, W., Buchholz, S. and Veenstra, J.(1999), Memory-based shallow parsing, *Proceedings of CoNLL*, Bergen, Norway, pp. 53–60.

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

Daelemans, W., Zavrel, J., Van der Sloot, K. and Van den Bosch, A.(2003), TiMBL: Tilburg memory based learner, version 5.0, reference guide,

ILK Technical Report 03-08, Tilburg University. available from http://ilk.uvt.nl/downloads/pub/papers/ilk.0308.ps.

Dudani, S.(1976), The distance-weighted k-nearest neighbor rule, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-6, pp. 325–327.

Lendvai, P., Van den Bosch, A. and Krahmer, E.(2003), Memory-based Disfluency Chunking, *DISS 2003 : Disfluency in Spontaneous Speech Workshop*, pp. 63–66.

Marcus, M., Santorini, S. and Marcinkiewicz, M.(1993), Building a Large Annotated Corpus of English: the Penn Treebank, *Computational Linguistics* **19**(2), 313–330.

Moortgat, M., Schuurman, I. and Van der Wouden, T.(2001), CGN Syntactische Annotatie, *Technical report*, http://lands.let.kun.nl/cgn/publicat.htm.

Oostdijk, N., Goedertier, W., Van Eynde, F., Boves, L., Martens, J.-P., Moortgat, M. and Baayen, H.(2002), Experiences from the Spoken Dutch Corpus Project, *LREC 2002 : Third Int. Conference on Language Resources and Evaluation*, Vol. I, pp. 340–347.

Quinlan, J.(1986), Induction of Decision Trees, *Machine Learning* **1**, 81–206.

Quinlan, J.(1993), C4.5*: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

Ramshaw, L. and Marcus, M.(1995), Text chunking using transformation-based learning, *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora*, Cambridge, MA, pp. 82–94.

Shepard, R.(1987), Toward a universal law of generalization for psychological science, *Science* **237**, 1317–1228.

Skut, W., Krenn, B., Brants, T. and Uszkoreit, H.(1997), An annotation scheme for free word order languages, *Proceedings of the Fifth Conference on Applied Natural Language Processing ANLP-97*, Washington, DC, pp. 88–95.

Stanfill, C. and Waltz, D.(1986), Toward memory-based reasoning, *Communications of the* ACM **29**(12), 1213–1228.

Van den Bosch, A.(1997), *Learning to pronounce written words: A study in inductive language learning*, PhD thesis, Universiteit Maastricht.

Van der Beek, L., Bouma, G., Malouf, R. and Van Noord, G.(2001), The alpino dependency treebank, *Computational Linguistics in the Netherlands, CLIN 2001*, Rodopi.

Van Eynde, F.(2001), Part of Speech Tagging en Lemmatisering, *Technical report*, Centrum voor Computerlinguistiek, K.U.Leuven, http://lands.let.kun.nl/cgn/publicat.htm.

Van Rijsbergen, C.(1979), *Information Retrieval*, Buttersworth, London, UK.

Veenstra, J.(1999), Memory-based text chunking, *Proceedings of ACAI'99*, Chania, Greece.

White, A. and Liu, W.(1994), Bias in information-based measures in decision tree induction, *Machine Learning* **15**(3), 321–329.