Memory-Based Grammatical Relation Finding

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Tilburg,
op gezag van de rector magnificus,
prof. dr. F.A. van der Duyn Schouten,
in het openbaar te verdedigen ten overstaan van
een door het college voor promoties aangewezen commissie
in de aula van de Universiteit
op vrijdag 13 december 2002 om 10.15 uur

door

Sabine Nicole Buchholz

geboren op 1 december 1970
te Braunschweig, Duitsland

Promotores: Prof. dr. W.M.P. Daelemans

Prof. dr. H.C. Bunt

Copromotor: Dr. A.P.J. van den Bosch

ISBN 90-9016431-6

printed by PrintPartners Ipskamp, Enschede
NUGI 949

Acknowledgement

This research was done in the context of the "Induction of Linguistic Knowledge" research programme, which was funded by the Netherlands Organization for Scientific Research (NWO). I would like to thank Harry Bunt for convincing me to ask for an extension of my PhD contract and the Faculty of Arts for granting it. The USENIX Association and Stichting NLnet funded my stay in Cambridge through the Research Exchange Program (ReX). Thanks to Ted Briscoe for the idea of the research exchange and for supervising me in Cambridge.

I am very grateful to my supervisors Walter Daelemans and Antal van den Bosch for their encouragement, useful discussions and patience. Many thanks also to Yuval Krymolowski and my former colleagues Jorn Veenstra and Jakub Zavrel for fruitful discussions and collaboration and to my colleague Bertjan Busser for technical and moral support. "Mazel" and "sterkte" with yours! Many other colleagues helped with this thesis in various ways. Ko van der Sloot implemented TiMBL, which plays such a central role in this thesis. Erik Tjong Kim Sang gave feedback on the chunks and the parser. Ielka van der Sluis, Martin Reynaert, Piroska Lendvai and Roser Morante proofread the drafts. Special thanks to Ielka for helping with the book cover and for willing to be one of my paranymphs. Christine Erb, Anna Korhonen and Yan Zuo demonstrated the whole procedure that belongs to finishing a PhD. In addition, several of these colleagues also became true friends.

Among the non-colleagues, Andreas Döring is the person I talked with most about my research. I am ever grateful for the many fundamental discussions, the encouragement, and the fun. Many thanks to Yu-Fang Helena Wang for useful discussions and effective distraction. Markus Kuhn shared with me the good times in Cambridge and the hard times when we were both writing up. I hope that there are many more good times together ahead! I am very grateful to my parents Fritz and Susi Buchholz and my sisters Angelika and Verena for supporting me during all these years in many ways.

I also wish to thank the people who helped me during the past five years by enriching "life besides the PhD". Despite everything, I am grateful to Wouter de Ruijter for the good times. Thanks to Gerhard Kordmann for a far away holiday in the middle of writing up. Thanks to Viola Spek, and the people of Flying High, Strange Blue and the zwaardkring for the sports that provided a balance to sitting in the office all day. Thanks to Yann Girard, Jan Kooistra, Aldo de Moor and Anne Breitbarth for fun and suspense in Catan and other

distractions in Tilburg. Thanks to Caroline Gödde, Heidi Vogelsang, Frauke Ruhe, Micha Klein and Annika and Jürgen Herz for staying in touch during all these years, despite the distance. Special thanks to Caroline for willing to be the other paranymph.

Finally I would like to thank some more colleagues for making my time at the university a pleasant one: Iris Hendrickx, Anne Adriaensen, Hans Paijmans, Elias Thijsse, Erwin Marsi, Emiel Krahmer, and Rein Cozijn from Tilburg, and Naila Mimouni, Judita Preiss, Aline Villavicencio, Simone Teufel, Donnla NicGearailt, and Advaith Siddharthan from Cambridge.

Tilburg, 4th November 2002

Sabine Buchholz

Contents

Chapter 1

Introduction

Grammatical relations (GRs) are interesting from a theoretical as well as from a practical point of view because they constitute a link between syntax and semantics. In many cases the surface subject and direct object of a verb correspond to the first and second argument of the verb's semantic predicate. If they do not (e.g. in a passive sentence), the deep grammatical relations determine the argument positions. Temporal, locative and other adjuncts introduce additional restrictions on the state or event described by the main verb. The predicate-logic structures that are derivable from GR information can be used for applications such as Question Answering (see Chapter 7).

This thesis is about finding grammatical relations to verbs in English sentences by means of a supervised machine learning algorithm. This introductory chapter defines the task (Section 1.1), introduces the general parsing framework in which the relation finder is used (1.2), lists the central research questions (1.3) and gives an overview of this thesis (1.4).

1.1 The task

The topic of GRs is closely related to the discussion about the complement/adjunct (C/A) distinction. Many tests have been proposed for distinguishing complements (like direct objects) from adjuncts (like non-obligatory temporal expressions), see e.g. Jackendoff (1977, p.58), Pollard and Sag (1987, p.134), and Meyers, Macleod, and Grishman (1994). However while the tests work well in most cases, different tests might yield different results for some problematic cases. Jacobs (1994) argues (for German) that this is due to different concepts of what complements actually are. He shows that for any pair from a set of seven common concept definitions there is at least one example that would be classified as complement by the first definition but as adjunct by the second. Thus "complement" is only a cover term for a group of concepts, whose extensions do not coincide. In the main part of this thesis we will not explicitly distinguish complements from adjuncts. Our GRs are based on the annotations in the Wall Street Journal (WSJ) Corpus of the Penn Treebank (release

II). An example of a problematic case for a C/A distinction is the temporal expression "so long" in the sentence "I don't understand why it's taken so long".[1] Although it is obligatory, some theories might consider it an adjunct. Our learner will just assign it the label "closely related temporal adverb phrase", with the special tag *closely related* marking "constituents that occupy some middle ground between argument and adjunct of the verb phrase" (Bies et al., 1995), that is without committing itself to either analysis. It is only in Section 6.2.2 where we map our GR labels to the GRs of another system for comparison that we have to decide whether a label can best be mapped to a complement or an adjunct (according to the other system's definition).

1.1.1 Relations to verbs

In this thesis we deal with GRs to verbs only. There are several reasons for this:

- The verb and its direct dependents are central to the meaning of a sentence. Together they provide the main logical assertion.
- The distinction between complements and adjuncts of nouns and adjectives is even more controversial than for verbs.
- Verbs typically have more dependents than nouns and adjectives. Therefore some dependents will be quite distant from the verb. In addition the form of complements is different: verbs can take complements that are simple NPs whereas nouns and adjectives cannot. They typically require *of* PPs for the same relation. Nouns also allow NPs in the possessive. It is therefore not clear whether the same information should be used to find GRs of verbs, nouns and adjectives. It might thus be better to keep these tasks separate. This is clearly a point for further research.
- The Penn Treebank marks only clausal complements of nouns as complements and annotates only post-modifying adjuncts of nouns as separate constituents. For example the phrase "October 1987" in "the October 1987 global stock crash"[2] is no separate constituent and can therefore not be annotated with any function (although it clearly is a temporal specification of "crash"). This means that the task of finding GRs of nouns and adjectives that is definable on this material is not comparable to the task for verbs. Including it would harm overall interpretability of results.

1.1.2 Chunks and heads

In principle GRs to verbs do not only include subjects, objects, temporal adjuncts and the like but also the relation between an auxiliary or modal and the main verb, as in "will join".

[1] From file wsj_1041.mrg
[2] From file wsj_0700.mrg

Depending on the theory, "join" is the verbal complement of "will" (e.g. in Collins (1997)) or "will" has an auxiliary relation to "join" (e.g. in Carroll and Briscoe (2001)). In any case, relations of this kind are easier to find than other GRs. Following Abney (1991) (see also Section 2.4.1.1) we distinguish between *chunking* and *attaching*. During chunking, word sequences like "will join" or "has not been found" are grouped together into *chunks*. The GRs between words *within* the same chunk are then predictable just by looking at the part-of-speech (PoS) of the words. For example, for any sequence "modal+infinitive" there will be a relation such as `verbal_complement(infinitive,modal)` respectively `aux(modal,infinitive)`. There is extensive literature on automatic chunking (reviewed briefly in Section 2.4.1.1).

In this thesis we concentrate on GRs *between* chunks, on which less work has been done. These are therefore more challenging and also the more important GRs for applications such as Information Extraction or Question Answering. We define each chunk to have a unique headword. Thus finding GRs between chunks is equivalent to finding GRs between (head)words. In summary the task studied in this thesis is to find grammatical relations between (heads of) verb chunks and (heads of) other chunks in English sentences.

1.2 The parsing framework

Wanting to find GRs between verb chunks and other chunks implies that chunks have already been found, and labeled with their type (verb chunk etc.). It also suggests that the relation finder should use this chunk information when performing its task. This fits the general framework of History-Based Grammars (HBG), which is introduced in Black et al. (1992). Black et al. (1992) start by noting that humans successfully cope with the ambiguities of natural language sentences by examining the context. The central questions are then:

- What exactly is the context?
- How much information about the context of a word, phrase or sentence is necessary and sufficient to determine its meaning?

In a HBG model, the context is called the history, where "history is interpreted as any element of the output structure, or the parse tree, which has already been determined, including previous words, non-terminal categories, constituent structure, and any other linguistic information which is generated as part of the parse structure". This even includes the parse trees of all the sentences preceding the current sentence in a discourse (the discourse history). Any following parse decision can then, in principle, be influenced by any piece of information from the history.

Black et al. (1992) implement a generative model, so the part of the history that belongs to the current sentence contains all the nodes that have already been generated, and all

the rules that have been used, in the leftmost derivation of the parse tree up to the current node. However this implementation is not crucial for HBG, and other authors have applied the term also to non-generative models. For example Collins (1999, p.128) refers to the bottom-up parser of Ratnaparkhi (1997) as history-based.

In our case the history for the current sentence contains at least all the words in the sentence and the boundaries and types of chunks. As chunking frequently presupposes PoS tagging[3] we will also assume that the history contains the PoS of all the words. Although this does not exhaust the information that might potentially be useful for relation finding (e.g. one might want to consider information on coreference resolution, or Named Entities, or the discourse history) it already constitutes a rich, internally structured input without any fixed maximum length. Black et al. (1992) use decision trees in order to cope with this abundance of information. We will use an alternative machine learning algorithm: Memory-Based Learning (MBL).

1.2.1 Memory-Based Learning

The use of a standard machine learning method for a new task has the advantage that there are fast, robust, flexible implementations. For example the Question Answering application described in Chapter 7 requires a fast implementation that can run in a server mode. On the other hand using a general method also imposes certain challenging restrictions. Like many machine learners, a Memory-Based Learner is a propositional learner and performs classification. The propositional format means that each instance (the unit of learning) has to be represented as a fixed number of feature value pairs. As we saw above, the input to relation finding is internally structured and does not have a fixed maximum length. One of the central questions of this thesis is then how to represent the information from the history in the format that is required by the learner.

Classification means that each instance is assigned a symbolic class label (as opposed to a numeric class in regression). This requirement is easily fulfilled as GR labels, which are the desired output, are symbols. MBL can perform multi-class classification (as opposed to binary classification). This means that we can use a single classifier to find all types of GRs at the same time. Much recent work shows that combinations of classifiers often perform better than a single classifier (Dietterich, 1998; van Halteren, Zavrel, and Daelemans, 2001; Tjong Kim Sang, 2002). In this thesis we restrict ourselves to the one-classifier architecture, which is faster. We perform extensive post-experimental analyses that show why certain algorithmic settings, additional features or other feature representations work. These analyses would be much harder if we had to deal with several classifiers that jointly determine the output.

[3]See also Section 2.4.1.1 on chunking. Van den Bosch and Buchholz (2002) report on experiments to find chunks (and function tags) on the basis of the words only.

In its standard mode, a Memory-Based Learner performs local, mutually independent decisions.[4] This allows the relation finder to work on incomplete sentences and to extract only specified GRs of specified verbs if needed. These two properties are crucial for the Question Answering application in which the relation finder is applied to the "text snippets" that the search engine Google returns, which are rarely whole sentences. As the application is online, speed is important. Given a question like "Who invented the telephone?" and a sentence like "Twenty years after the News Letter was first printed, the telephone was invented by Alexander Graham Bell, in 1876." it would be unnecessary work to determine the GRs of "printed" or to find more than the (deep) subject and object of "invented".

In summary we try to perform the task of finding GRs to (heads of) verb chunks using a single Memory-Based classifier that performs local, independent decisions.

1.2.2 Cascaded Memory-Based Shallow Parsing

Although finding GRs to verb chunks presupposes a tagging and chunking step, we want to abstract from the properties, and errors, of any particular tagger or chunker for the main experiments of this thesis (Chapters 4 and 5). We will therefore directly extract the PoS tags and chunks from the treebank. This procedure is explained in detail in Section 3.1. For comparison with other systems that either do not assume a separate chunking step or define chunks differently, and for practical applications, however, we need to rely on an actual tagger and chunker (see Section 6.1.4).

To make this concrete we assume our GR finder to operate in the context of the cascaded Memory-Based Shallow Parser (MBSP) that was first suggested by Daelemans (1996) and first implemented by Buchholz, Veenstra, and Daelemans (1999). It consists of several modules that are applied in sequence: the Memory-Based Tagger (MBT), see Daelemans et al. (1996), a memory-based chunker (Veenstra and Van den Bosch, 2000), a memory-based PNP finder (although nothing hinges on all these modules being memory-based) and finally the memory-based relation finder. The PNP finder combines prepositional chunks and noun chunks to what we will call PNP chunks. For example it would combine the chunks "$[_{PP}$ `instead of ]` $[_{NP}$ `John ] , [`$_{NP}$ `Peter ] and [`$_{NP}$ `Mary ]`" to the PNP chunk "$\{_{PNP}$ $[_{PP}$ `instead of ]` $[_{NP}$ `John ] , [`$_{NP}$ `Peter ] and [`$_{NP}$ `Mary ] }`".

In this thesis we concentrate on the relation finder. Thus the task is to find GRs to verbs given the words, tags, simple chunks and PNP chunks.

[4] It is possible to use the output of previous decisions of the same learner as additional features, which makes decisions dependent on each other (this is implemented for example in the Memory-Based Tagger (Daelemans et al., 1996)). It is also possible to use beam search to find a decision sequence that is globally optimal (p.c. Jakub Zavrel). However this does not yield a big improvement, probably because the local decision is already based on much information, including the right context.

1.3 Research questions

Given the above task definition, we investigate the following two aspects:

- **What information is useful for performing the task?** In linguistics a similar question is often answered with the help of examples. If there is one example of the task for which some information can be shown to be useful then this information is deemed useful for the task. By contrast our approach is more quantitative. Certain pieces of information will be relevant for most examples (i.e. instances) so their influence will be great. Others are only relevant in rare cases. Although our error analysis might still uncover these cases, influence of the information on overall performance will be negligible. Such a quantitative analysis might prove insightful for linguists as well. It is certainly relevant for people who attempt a similar task with a different (machine learning or other) technique as even the smartest method needs the crucial information.
- **How can this information best be used by a Memory-Based Learner?** As the definition of "best" depends on the context of the application, there are actually three subquestions:
  - Which information and representation yields the best performance?
  - Which information and representation speeds up or slows down the process (speed-performance trade-off)?
  - Which information and representation increases or decreases memory requirements of the process (memory-performance trade-off)?

  The answers to these questions are interesting for anyone who wants to use a memory-based relation finder for some application, especially if the type of application (e.g. online) imposes constraints on speed and/or memory. In selected cases we will not only look at overall results but also break down performance by individual relations. This analysis can help to design tailored classifiers for applications that use only a subset of the GRs.

1.4 Overview

This thesis is structured as follows. Chapter 2 sketches the theoretical background of grammatical relations: how they are defined, what phenomena are described by them and how they are annotated in treebanks. It also reviews work that is related to determining GRs. The review shows that diverse types of information are relevant to the task and that this information can be represented in different ways. It thereby provides a point of reference for our own experiments on this topic. Sections 2.4.3 and 2.4.4 specifically review other work on GR extraction.

Chapter 3 describes the original Penn Treebank data and how we extracted chunks, heads and GRs from it. This conversion is complex because this information is not explicitly contained in the treebank. The second part of the chapter explains the general set-up for the experiments in the following two chapters, which form the core of this thesis.

Chapter 4 introduces the MBL algorithms that are used in this thesis and explains the various parameters. The second part of the chapter applies these algorithms with different parameter settings to the GR data. The most interesting improvement over the default setting is achieved through the Modified Value Difference Metric (MVDM) which models task specific similarity between feature values. Our analyses show that this allows the algorithm to implicitly learn hierarchies of PoS, syntactic and semantic similarity between words and a non-linear measure of "distance" in a sentence.

Chapter 5 systematically tries to improve performance and/or speed and memory requirements by deleting superfluous features and adding useful new ones and by trying different representations of the same information. The most interesting new information involves sequences of PoS or chunks. Representing information from sequences is especially challenging in a propositional format. We present two possibilities: using MVDM on sequences regarded as atomic values and using numeric features. We also show how information from words that are not semantic but syntactic heads can help the learner. Even knowledge about the absence of such words in a chunk can be relevant.

The first sections of Chapter 6 treat several practical issues. They show that training the learner with a more fine-grained class definition than is actually needed does not harm results on a coarser-grained evaluation. Training on material that is slightly different from test material, however, decreases performance. This is shown by experiments with two different corpora and with manually annotated versus automatically tagged and chunked text. In the last section of Chapter 6, the cascaded Memory-Based Shallow Parser is compared to other systems that also extract grammatical relations.

In Chapter 7, MBSP is integrated into a Question Answering prototype. This application demonstrates that the performance that can be achieved by the parser is sufficient for Question Answering when large numbers of documents are available. This is typically the case on the World Wide Web.

Chapter 8 summarizes the answers to our research questions and suggests future research.

Chapter 2

Theoretical background and related work

This chapter gives an overview of work from various subfields of (computational) linguistics that are concerned with GRs. It serves several purposes:

- It provides the theoretical background for Chapter 3 (Data) by describing the fundamentals of phrase structure and dependency structure syntax.
- It introduces many of the phenomena described through GRs, the problematic cases for these descriptions and the terminology used for both. Many of these concepts will return in qualitative analyses in subsequent chapters.
- It reviews related work in parsing and subcategorization extraction (amongst others the two approaches with which we compare our system in Section 6.2). Some approaches use techniques that are also employed in our Memory-Based Shallow Parser. Others illustrate alternative approaches.

We briefly describe each method. In overview tables, we focus on the question what tests, heuristics or pieces of information are used to determine 1) whether there is a GR between two units (words, chunks or constituents), and 2) what type of GR it is. Question 1) is the *attachment problem*. Question 2) is connected to the problem of the complement/adjunct distinction.

The chapter is divided into six sections, each of which is devoted to a subfield of (computational) linguistics:

Grammar theories (Section 2.1) introduce formalisms for describing the grammars of natural languages. They differ for example with respect to whether they treat GRs as primitives of the theory or as derived from some other representation.

Grammars (Section 2.2) typically try to give a comprehensive overview of all the phenomena in one language. As GRs are part of the classic inventory of grammar descriptions, their properties are also described in grammars in detail.

Treebanks (Section 2.3) are corpora in which each sentence has been manually annotated with its syntactic structure. The most central concept is that of attachment, which is typically expressed through the tree structure. Depending on the annotation scheme, different types of GRs are distinguished and marked either implicitly or explicitly.

Parsers (Section 2.4) automatically assign a syntactic structure to (new) sentences. As present-day parsers are frequently trained and tested on treebanks, they are largely dependent on the treebank's annotation scheme. Much knowledge and many tests that treebank annotators use to determine the structure of a sentence cannot be automated. Instead it has to be specified explicitly which parts of the information that is available should be used by the algorithm to make attachment and GR type decisions.

Subcategorization dictionaries (Section 2.5) list the subcategorization frame(s) for lexical items. The items are at least the verbs, and sometimes also adjectives or nouns. The C/A distinction is crucial for subcategorization dictionaries as a subcategorization frame is a list of the complements that the lexical item can take. Thus the inventory of complement GRs determines how fine-grained the description of a frame can possibly be. Adjuncts are only treated insofar as this is necessary to distinguish them from complements. Subcategorization dictionaries typically have guidelines for lexicographers for making this distinction.

Automatic subcategorization acquisition (Section 2.6) tries to achieve automatically what is done manually for most subcategorization dictionaries. Thus most of what was said about the dictionaries carries over to these automatic methods: the C/A distinction is crucial, and the inventory of GRs largely determines the number of frames. As with parsers, one has to specify explicitly on what information decisions are based.

2.1 Grammar theories

This section describes a limited selection of grammar theories. A lot of other work that could have been included here has been left out, e.g. Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), Tree Adjoining Grammar (TAG) (Joshi, 1987) and most notably the work of Noam Chomsky. However, many of their ideas and formal solutions return in other theories.

2.1.1 $\overline{\mathrm{X}}$ syntax

$\overline{\mathrm{X}}$ theory (Jackendoff, 1977) is a refinement of the general system of phrase structure (PS) grammars. It uniquely defines a head for each constituent, restricts the shape of grammar rules, and allows cross-category generalizations. With a few exceptions all grammar rules are of the form $\mathrm{X}^n \rightarrow (\mathrm{C}_1) \ldots (\mathrm{C}_j)\ \mathrm{X}^{n-1}\ (\mathrm{C}_{j+1}) \ldots (\mathrm{C}_k)$, where X can be a lexical category like N(oun), V(erb), A(djective) etc. X^n is the nth projection of X (the nth bar-level) and X^{n-1} is the head child of the rule. According to the same scheme, X^{n-1} itself must contain a head child X^{n-2} and so on down to X^0, i.e. X, which is a lexical item and the lexical head of all the X^is above it. The maximal projection of X is called a phrasal category. It is commonly denoted by XP. X^1, X^2 and so on are also written as $\overline{\mathrm{X}}$, $\overline{\overline{\mathrm{X}}}$ or X′, X″. All the C_is in the above rule must be phrasal categories or specified grammatical formatives (like auxiliaries or complementizers). In principle, the C_is are optional (which is indicated by the parentheses around them).

Coordination forms an exception to the general $\overline{\mathrm{X}}$ scheme. A coordinated phrase has the same category and bar level as the conjuncts, and it is unclear what the head is. Syntactic categories are defined through their values for distinctive syntactic features. For example the four major parts-of-speech can be distinguished through two binary features. Jackendoff (1977, p.32) defines verbs and nouns to be *Subj+*, and verbs and prepositions to be *Obj+* whereas Chomsky (1970) groups verbs and adjectives (*V+*) and nouns and adjectives (*N+*) together. In any case, these features allow generalizations across categories by defining certain constructions to be possible for all categories that are e.g. *N+*.

The C_is to the left of the head are called its specifiers, the C_is to the right its complements (or, together, the complement). Note that this is a different use of the term complement than in the rest of this thesis. Jackendoff (1977) distinguishes three bar levels and accordingly three types of complements:[1]

- Functional arguments. These are subcategorized. Except for the subject, they attach under X′. Semantically, they are arguments of the head's predicate. Examples are direct, indirect and predicative NP object, predicative AdjP, subcategorized AdvP or QP (quantifier phrase), particle, clausal and PP object.

- Restrictive modifiers attach under X″. They contribute to the main assertion of the sentence by adding extra truth conditions. Under V″, they include AdvPs and PPs of manner, means, time, instrument, purpose or accompaniment, measure phrases, QPs, and adverbial and comparative clauses. The attachment of arguments to X′ and of restrictive modifiers to X″ means that under the standard $\overline{\mathrm{X}}$ schema, the former always precede the latter. This in turn means that cases like "John gave the beans quickly to Bill" and "John told me yesterday that he met Mary" have to be

[1] Chomsky (1970) assumes only two bar levels, with complements under $\overline{\mathrm{X}}$ and specifiers under $\overline{\overline{\mathrm{X}}}$ (rules: $\overline{\mathrm{X}} \rightarrow$ X Comp; $\overline{\overline{\mathrm{X}}} \rightarrow$ [Spec,$\overline{\mathrm{X}}$] $\overline{\mathrm{X}}$). Adjuncts must then be attached by so-called Chomsky-adjunction through rules like $\overline{\mathrm{X}} \rightarrow \overline{\mathrm{X}}$ Adjunct, in which the bar level of the head category stays the same.

considered extraposition, and represented with coindexed traces ($\_\__i$) and fillers (S^1_i), e.g. like
(S (NP John) (V^2 (V^1 (V^0 told) (NP me) $\_\__i$) (NP yesterday) (S^1 that ...)$_i$)).

- Non-restrictive modifiers attach under X‴. Semantically, they add an auxiliary assertion, one of whose arguments is usually the main assertion. They include sentence adverbials, sentential appositives, parentheticals, and other subordinate clauses under V‴(i.e. S).

Jackendoff (1977, p. 58) also lists some criteria for distinguishing the three types of complements. Information that is testable against the surface syntax of a single sentence includes the *do so* construction, clefting, commas, position in the sentence and order relative to other complements. Other criteria for complements of verbs refer to obligatoriness, focus and sentence negation.

In $\overline{\text{X}}$ theory, grammatical functions are defined through configurations in the tree. The direct object, for example, is the first NP after the head under X′.[2] This approach means that grammatical functions like "temporal adjunct" are not definable syntactically, only distinctions between different categories and attachment levels are possible. A distinction like that between direct and predicative object can only be expressed in the subcategorization frame of the verb.

The $\overline{\text{X}}$ scheme does not depend on a specific definition of heads. In Chomsky (1986), the old VP is reanalyzed as IP consisting of an Infl(ection) head which subcategorizes for a VP and the subordinate clause as CP, with a Comp(lementizer) head that subcategorizes for an IP. Abney (1987) analyzes the old NP as DP, with a Det(erminer) head that subcategorizes for an NP and the AdjP as a DegP, with a Deg(ree) head that subcategorizes for an AdjP. The analysis of PPs as having a P head that subcategorizes for a DP/NP already fits this new scheme. The new analyses distinguish between *lexical heads*, which belong to an open-class PoS (nouns, verbs, adjectives) and *functional heads*, which are closed-class (verb inflectional suffixes and modal auxiliaries, complementizers, determiners, adjective inflectional suffixes and degree particles like *too (big), as (big), so (big)*, prepositions). Other names for this distinction are semantic and syntactic heads respectively. We will use these latter names in the remainder of this thesis, as "lexical head" is used to refer to *any head* that is a word, in contrast to the head of a rule which might be a constituent.

2.1.2 Dependency syntax

Dependency syntax (Tesnière, 1959; Mel'čuk, 1988) stands in opposition to phrase structure syntax. Figures 2.1 and 2.2 show a phrase structure tree and a dependency structure for the same sentence (adapted from Mel'čuk (1988)). Whereas PS is concerned with constituents (the higher nodes in the tree), the categories of constituents (node labels),

[2] Jackendoff (1977) assumes an obligatory *of*-insertion rule for direct objects of NPs and ADJPs.

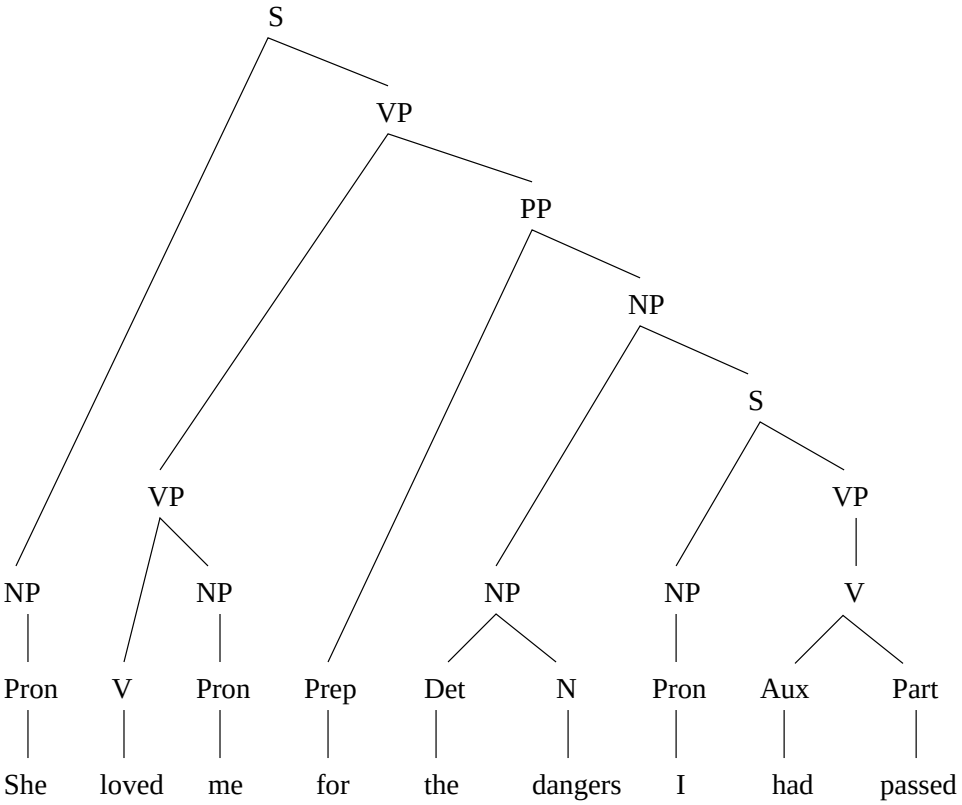

Figure 2.1: A phrase structure tree for a sentence. This is an example only, the constituent labels do not mean to express any particular theory.

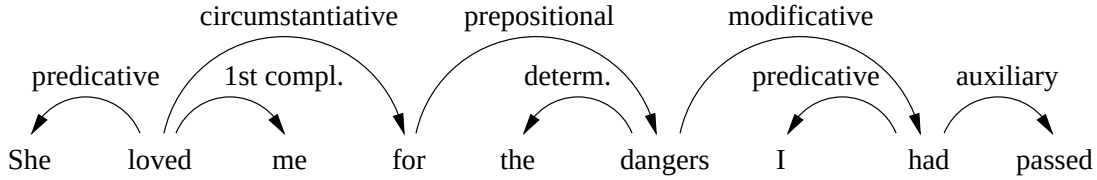

Figure 2.2: A dependency structure for the same sentence. This is an example only, the dependency labels do not mean to express any particular theory (1st compl. = 1st completive, determ. = determinative).

and how constituents can be combined into larger constituents, dependency syntax only accepts relations (i.e. dependencies) between words as primitives. The arrows, or arcs, in the figure represent relations. Following Mel'čuk (1988), they point from the head to the modifier.[3] Instead of *head* and *modifier*, *governor* and *dependent* are also used. We can say that the head *governs* the modifier, and the modifier *depends* on the head.

In addition to (surface) syntactic dependencies, like those depicted in our example, Mel'čuk

[3]Note that other authors (Eisner, 1996a; Krymolowski and Dagan, 2002) use arrows from the modifier to the head. Although this is confusing, it does not affect the underlying theory or algorithm.

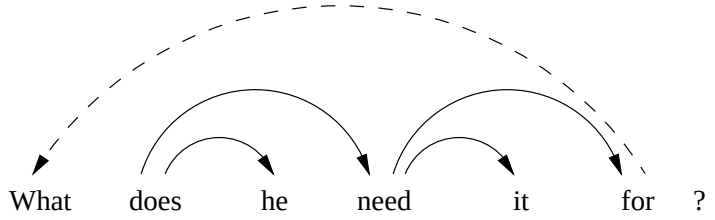

Figure 2.3: A non-projective sentence: the link between "what" and "for" (dashed) covers the root "does". Labels are not shown.

(1988) also acknowledges morphological and semantic dependencies between words, and anaphoric and communicative links. Morphological and semantic dependencies often coincide with syntactic dependencies, but they do not have to. Agreement for example is a morphological dependency. The adjective in a German NP reflects the gender of the noun, the definiteness of the determiner and the case that the verb imposes:

(1) Der klein*e* Mann lacht. (masculine, nominative, definite)
Ein klein*er* Mann lacht. (masculine, nominative, indefinite)
Ich sehe den klein*en* Mann. (masculine, accusative, definite)

Syntactically however, the adjective only depends on the noun. In a control construction, like "He promises her to come", there is a semantic subject relation between "he" and "come" without a (direct) syntactic relation. Similarly, there is a semantic object relation from "passed" to "dangers" in the example sentence in Figure 2.2 while there is no direct syntactic relation.[4] This is a case of a *non-local dependency* (also called *long-distance* or *unbounded dependency*). However, in general, "morphological dependencies are used to indicate syntactic dependencies. Syntactic dependencies, in turn, generally indicate semantic dependencies" (Mel'čuk, 1988, p.118). In contrast to morphological and semantic dependencies, syntactic dependencies always link all the words in a sentence into one structure.

A syntactic dependency structure is a connected directed labeled graph. It has exactly one non-governed node (top node or root) and all the other nodes have exactly one governor. From this latter requirement and the connectivity it follows that syntactic dependencies cannot form cycles. Mel'čuk (1988) does not exclude crossing links. However, he notes that a sentence is called *projective* if 1) no arc covers the top node, and 2) arcs do not cross. He also notes that most sentences are projective. Figures 2.3 and 2.4 show two sentences that are not projective. Cases in which the best dependency analysis is not obvious include notorious linguistic problems like coordination, especially with gapping, and multi-word expressions like compound prepositions and idioms. Although Mel'čuk (1988) does not give an inventory of syntactic relations, example 2.2 shows that there is no complement/adjunct dichotomy. Rather, different types of adjuncts are identified by different relation labels. The general claim is that syntactic relations are recoverable from the combined information

[4] The indirect relation *dangers* → *had* → *passed* points in the other direction.

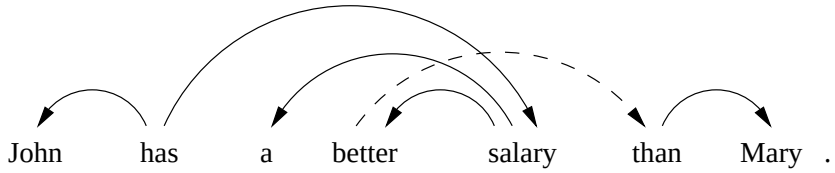

Figure 2.4: A non-projective sentence: the link between "better" and "than" (dashed) crosses several other links. Labels are not shown.

of function words (e.g. governed prepositions and conjunctions, auxiliary verbs), word form arrangements (word order), prosody (in speech), and inflections (e.g. for agreement).

2.1.2.1 Converting between dependency and phrase structures

To turn a phrase structure tree into a dependency structure (which is roughly what we do when converting the original treebank data into our grammatical relation format, see Section 3.1) one first has to identify the lexical head of each constituent. This can be done by following the projection path. Next, one has to make the implicit GRs explicit. So for each application of a rule like $X''' \rightarrow NP\ X''$ one notes a subject relation between the head of the NP and the head of X''. For each application of a rule like $X' \rightarrow X^0\ NP\ \ldots$ one notes a direct object relation between the head of the NP and X^0, and so on. Alternatively, one can use a description of the configuration directly as label, e.g. NP_V'''_V'' instead of subject (of a verb) and V^0_V'_NP instead of object (of a verb). Note that one then needs a special device to make the distinction between direct and indirect objects. Note also that given PS trees as described in Jackendoff (1977), one cannot distinguish predicative from non-predicative objects and modifiers can only be distinguished into "PP restrictive modifier", "PP non-restrictive modifier", "ADVP restrictive modifier", etc. and not into semantic classes like "temporal modifier" or "locative modifier". Trace-filler constructions have to be resolved before the dependencies are computed.

To turn a dependency structure into a phrase structure, we first have to define constituents. If the sentence is projective, a sequence of a head and its direct dependent(s) and those dependents' dependents and so on forms one constituent. We need a convention, based on the GRs and/or the parts-of-speech of X, to decide whether a structure like $Y \leftarrow X \rightarrow Z$ corresponds to the bracketing (Y X Z) or ((Y X) Z) or (Y (X Z)). Next we have to determine the category of the constituent by looking at the PoS of the head, which yields the X, and by checking what types of complements or adjuncts are attached, which yields the bar level. If the sentence is not projective, we have to introduce traces for the crossing links or introduce discontinuous PS trees (PS trees with crossing branches), see Bunt and Horck (1996).

2.1.3 Lexical-Functional Grammar

Lexical-functional grammar (LFG) (Kaplan and Bresnan, 1982) uses two levels of representation. In constituent structure (c-structure) context-free rules are used to assign a phrase structure tree to a sentence. Rules and lexical entries carry functional descriptions which together define the functional structure (f-structure) of a sentence. The f-structure alone is the input to the semantic component which derives predicate-argument formulas.

F-structure is a directed acyclic graph and consists of features (e.g. TENSE), symbols as feature values (e.g. PAST), and semantic forms as feature values (e.g. 'give$\langle$(↑ SUBJ), (↑ OBJ), (↑ OBJ2)$\rangle$'). Some features denote grammatical functions (SUBJ, OBJ, OBJ2). Grammatical functions are thus primitives of the theory and not defined through phrase structure. Examples of grammatical functions are SUBJ(ect), OBJ(ect), OBJ2 (second object), oblique objects introduced by a preposition like TO OBJ or BY OBJ, COMP (*closed complement* like subcategorized *that*-clause, *wh*-clause, etc.), XCOMP (*open complement* like subcategorized infinitive and predicative object, see below), and the rather generic ADJ(unct). Semantic forms consist of a predicate and its arguments, e.g. give$\langle x, y, z\rangle$.

In Kaplan and Bresnan (1982) grammatical functions denote surface functions. Thus a lexical redundancy rule for passive works by changing SUBJ to BY OBJ and OBJ to SUBJ to derive e.g. a lexical entry for the passive participle *handed* from the finite form *hand*. However the old SUBJ's and new OBJ's value is still associated with the semantic form's predicate's first argument. As it is with these argument positions that thematic roles like Agent or Patient are associated, the intuition that active and passive constructions of the same verb roughly mean the same is still captured. Due to this distinction between grammatical functions and semantic argument positions, also expletives like "it" and "there" or parts of idiomatic expressions can have grammatical functions without showing up in semantic form. Grammatical functions are also used to define predicative verbs and *control verbs*. Both subcategorize for an *open* complement XCOMP, which has no overt subject.

(2) The girl persuaded the baby to go.

persuaded V (↑ XCOMP SUBJ) =(↑ OBJ)
(↑ PRED) = 'persuade$\langle$(↑ SUBJ), (↑ OBJ), (↑ XCOMP)$\rangle$'

(3) The girl promised the baby to go.

promised V (↑ XCOMP SUBJ) =(↑ SUBJ)
(↑ PRED) = 'promise$\langle$(↑ SUBJ), (↑ OBJ), (↑ XCOMP)$\rangle$'

For example the (simplified) lexical entry for the verb *persuaded* in (2) captures the intuition that the understood subject of the infinitival complement (XCOMP) is identical to the object of the matrix verb (object control, see the example sentence), whereas for the verb *promised* the understood subject is the matrix verb's subject (subject control). These two types of verbs are called *equi verbs*. In *raising verbs* there is also control of the understood subject but the *controller* does not appear in the semantic form of the matrix

verb. See (4) for an example sentence and lexical entry (all examples from Kaplan and Bresnan (1982)).

(4) The girl expected the baby to go.

expected V (↑ XCOMP SUBJ) =(↑ OBJ)
(↑ PRED) = 'expect⟨(↑ SUBJ), (↑ XCOMP)⟩'

2.1.4 Head-Driven Phrase Structure Grammar

Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1987; Pollard and Sag, 1994) adopts a PS analysis of sentences in which the head is of central importance. Following $\overline{\text{X}}$, the head determines most of the syntactic properties of its projections. We saw that in $\overline{\text{X}}$ theory, category symbols were taken as abbreviations for a combination of syntactic feature values. HPSG carries this approach to its logical conclusion by encoding *all* information in features. *Signs*, which correspond to words or constituents, are feature structures. Even the tree structure is encoded in features: a phrasal sign has a DAUGHTERS feature which contains the signs for all its children. This models *immediate dominance* relations between signs. *Linear precedence* ("which constituent is realized before or after which other constituent") is taken to be derived from the information contained in the feature structure by independent principles (see below).

Features are hierarchically organized. A sign has features for PHONOLOGY, SYNTAX and SEMANTICS. Semantic features contain information about quantifiers, gender, number, kinds of pronouns, etc. Syntactic features are divided into local and non-local features[5] (binding features; for non-local dependencies). Local features are lexicality (a sign is either lexical or phrasal), SUBCAT, which has a list value, and various head features. Head features are either binary (auxiliary, inverted, predicative) or have symbolic values (major syntactic category, case, verbal form (finite, infinitival, present participle, etc.), nominal form (expletive or "normal"), and preposition of PP). The particular value of features for a given phrase or sentence is derived by unification between the lexical signs, rules and universal and language-specific principles. For example the *locality principle* states that only the information under a sign's SYNTAX and SEMANTICS feature but not the information under DAUGHTERS is accessible to any other sign that wants to combine with it. This in turn means that information stemming from a sign's non-head children can only influence this sign's syntactic behaviour if this information is unified into the SYNTAX or SEMANTICS of the head child (from where it is unified into the parent).

The distinction between complements and adjuncts is directly expressed in the feature structure by having different types of non-head-daughters: complement-daughters, adjunct-daughters and filler daughters (for non-local dependencies). Pollard and Sag (1987, p.134) list criteria for distinguishing complements from adjuncts which refer to optionality, syntactic category, semantic type (e.g. manner, durative), linear order and iterability. The

[5]Similar to GPSG's head and foot features (Gazdar et al., 1985).

complements (including the subject in HPSG) are listed under the SUBCAT feature of the head. Unlike $\overline{\text{X}}$ theory, HPSG allows complements and adjuncts to be sisters, and even to be interspersed. Constraints on the unmarked relative order of signs are expressed by referring to a sign's lexicality, category, discourse function (like FOCUS) and the relative *obliqueness* of constituents. Obliqueness of complements is determined by their order on the head's SUBCAT list. This order also defines grammatical functions. Thus the least oblique complement is the subject, the second-least oblique complement is the direct object and so on. Adjuncts and heads are more oblique than complements. As in the unmarked case, less oblique constituents have to precede more oblique ones, the typical order of "direct object < indirect object < adjuncts" is captured. Other linear precedence constraints handle the order of focussed constituents.

2.1.5 Summary

In this section we described the alternative syntactic representations of PS and dependency structure and how they can be converted into one another. GRs are defined through tree configurations in $\overline{\text{X}}$ theory, and through obliqueness/order on the SUBCAT list in HPSG whereas they are primitives of the theory in dependency grammar and LFG. The C/A distinction is expressed through attachment to different bar levels in $\overline{\text{X}}$ theory, through occurrence in the semantic predicate in LFG and through the SUBCAT list in HPSG, while it is not explicitly represented in dependency grammar. We explained the idea of syntactic and semantic heads, how GRs have been used to encode predicative objects, passive and control, and HPSG's locality principle which restricts the features that can theoretically be relevant for the combination of a head and a dependent. Phenomena like expletives, idiomatic expressions, multi-word prepositions, coordination, extraposition or non-local dependencies pose special challenges for theories.

2.2 Grammars

There are many grammars of English, some very general, others tailored to the needs of e.g. native and non-native language learners. We only describe the grammar of Quirk et al. (1985) here. It is a very comprehensive grammar, and also the one referenced in the annotation guidelines for the Penn Treebank, which forms the basis of our experimental data.

2.2.1 Quirk

The grammar of Quirk et al. (1985) describes many English linguistic phenomena in great detail. Especially interesting for our purposes is the comprehensive treatment of adjuncts (traditionally called *adverbials*), a topic that receives much less attention in most syntactic

theories than complements. Quirk et al. (1985) distinguish seven semantic types of adjuncts and many subtypes:

- space: position, direction, distance
- time: position, duration, frequency, relationship *(still, already)*
- process: manner, means, instrument, agentive (the *by*-phrase in passives)
- respect (as in "With respect to the date, many people are expressing dissatisfaction.")
- contingency: cause, reason, purpose, result, condition, concession
- modality: emphasis *(certainly)*, approximation *(probably)*, restriction *(only)*
- degree: amplification and diminution *((not) very much)*, measure *(sufficiently)*

All semantic types of adjuncts can be realized by nearly all types of syntactic phrases: NPs, AdvPs, PPs, verbless, finite and non-finite clauses. Syntactically the (sub)types differ in

- their *wh*-element in relative clauses or questions: *where (to/from), (since/till) when, how (far/long/often/much)*
- their possible or preferred/unmarked position in the sentence as a whole: e.g. degree adjuncts cannot occur sentence-initially, time adjuncts often occur sentence-initially, space adjuncts often sentence-finally, adjuncts of modality and degree can occur sentence-medial, modality occurs rarely sentence-finally. These preferences of semantic types interact with the syntactic realization, e.g. whereas space adjuncts in general often occur sentence-finally, PPs indicating position in space (*where*) are also often found sentence-initially. Another factor is the *heaviness* which roughly corresponds to the length of a constituent. Longer constituents usually contain more information and often this information is new in the discourse. As adjuncts have more freedom of movement, their heaviness influences their position more than it influences the position of complements: e.g. adjuncts found sentence-medial are typically single word AdvPs.
- their order relative to other adjuncts, especially in sentence-final position (in sentence-initial or medial position, more than one adjunct is avoided). The unmarked order is respect $<$ process $<$ space $<$ time $<$ contingency. Again, heaviness and the requirements of the information focus can change this default. There are also preferences between subtypes: e.g. for time: duration $<$ frequency $<$ position.

In summary, semantic types and subtypes, syntactic category and heaviness influence the possible positions of adjuncts, both with respect to the verb and the complements and with respect to other adjuncts. Conversely this means that the relative surface position of an adjunct together with its syntactic category should help in determining its semantic type, i.e. its function.

```
( (SBARQ (WHNP-1 (WP Who))
       (SQ (VBD was)
           (NP-SBJ-2 (-NONE- *T*-1))
           (VP (VBN believed)
               (S (NP-SBJ-3 (-NONE- *-2))
                  (VP (TO to)
                      (VP (VB have)
                          (VP (VBN been)
                              (VP (VBN shot)
                                  (NP (-NONE- *-3)))))))))
       (. ?) ))
```

Figure 2.5: The sentence "Who was believed to have been shot?" in Penn Treebank II annotation. The asterisks mark empty elements (traces, with PoS `-NONE-`). They are coindexed with their fillers. Function tags after the syntactic category indicate grammatical function. Here, only subjects carry overt tags (`-SBJ`). This annotation allows to extract the underlying predicate-argument structure of the sentence: `believe('someone', shoot('someone', who))`

2.3 Treebanks

The annotation scheme of treebanks is normally loosely based on some grammar theory. Of the two treebanks described in this section, the Penn Treebank implements a relatively pure PS approach whereas the NEGRA treebank uses a combination of PS and dependency annotation.

2.3.1 Penn Treebank

The Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993; Marcus et al., 1994; Bies et al., 1995) is the largest English treebank and probably the one most widely used in computational linguistics (e.g. by most of the work in Section 2.4). It is also the basis for the experiments reported in this thesis. We will therefore describe its annotation scheme in more detail in a later section (3.1.1). Here we only give a general overview. The Penn Treebank Project started in 1989. Between then and 1992, 4.5 million words of American English were automatically PoS tagged and then manually corrected. About two thirds of the material were also automatically parsed and then hand corrected. The first release uses a basically context-free PS annotation for parse trees, where node labels are mostly standard syntactic categories like NP, PP, VP, S, SBAR etc. (cf. Appendix A.2). In 1995 a new version was released that used a much richer annotation including coindexed null elements (traces) to indicate non-local dependencies, and function tags on the node labels to indicate the grammatical function of a constituent. Figure 2.5 shows an example.

The parsed texts are from the 1989 Wall Street Journal (WSJ) corpus, and from the Air

Travel Information System (ATIS) corpus (Hemphill, Godfrey, and Doddington, 1990). This second release is the basis for the experiments reported in Chapter 4 and following. A third release came out later using basically the same annotation scheme as the second but including also a parsed version of the Brown Corpus (Kucera and Francis, 1967). This data is used for additional experiments in Chapter 6.

The PoS tag set is the same in all three releases. It is based on the Brown Corpus tag set but the Penn Treebank project collapsed many Brown tags (Marcus, Santorini, and Marcinkiewicz, 1993). The reasoning was that statistical methods, which were used for the first automatic annotation and envisaged as potential "end users" of the treebank, are sensitive to the *sparse data problem*. This problem comes into play if certain statistical events (e.g. the occurrence of a certain trigram of PoS tags) occur very infrequently or not at all in the training data so that their probability cannot be estimated properly. The sparseness of the data is related to the size of the corpus and the size of the tag set. Thus given a fixed corpus size, the sparse data problem can be reduced by decreasing the number of tags. Consequently, the final Penn Treebank tag set has only 36 PoS tags for words[6] and 9 tags for punctuation and currency symbols (\$,£). These are listed in Appendix A.1. Most of the reduction was achieved by collapsing tags that are recoverable from lexical or syntactic information. For example, the Brown tag set had separate tags for the (potential) auxiliaries *be, do* and *have*, as these behave syntactically quite different from main verbs. In the Penn tag set, these words have the same tags as main verbs. However, the distinction is easily recoverable by looking at the lexical items.[7] Other tags that are conflated are prepositions and subordinating conjunctions (together IN) and nominative and accusative pronouns (together PRP) as these distinctions are recoverable from the parse tree by checking whether IN is under PP or under SBAR, and whether PRP is under S or under VP or PP.[8] It should be noted however that most parsers, including MBSP, use the original (conflated) Penn tags.

The syntactic annotation is guided by the same considerations as the PoS tagging. There is e.g. only one syntactic category label (SBAR) for *that*- or *wh*-clauses and only one (S) for finite and non-finite (infinitival or participial) clauses although the two types behave syntactically quite differently. Again the argument is that these distinctions are recoverable by inspecting the lexical material in the clause[9] and again all parsers basically use the simple treebank categories.

In general only maximal projections (NP, VP, ...) are annotated, i.e. intermediate bar levels (N′, V′) are left unexpressed (with the exception of SBAR). In the first release,

[6] In addition to these 36 simple tags, a word can also get a disjunction of tags if the correct single tag cannot be determined. This sometimes happens with JJ|NN (adjective or noun as prenominal modifier), JJ|VBG (adjective or gerund/present participle), JJ|VBN (adjective or past participle), NN|VBG (noun or gerund), and RB|RP (adverb or particle).

[7] It is not clear why the punctuation and currency tags are not conflated then, too.

[8] In practice it is more complicated than this. In small clauses like the italic part in "I see *him swimming*" there is an accusative pronoun under S according to the treebank annotation (Bies et al., 1995, p.257).

[9] The absence of a *that* complementizer is marked by a special null element.

the distinction between complements and adjuncts of verbs was expressed by attaching complements under the VP as sisters of the verb and by adjoining adjuncts at the VP level. In the second release, both complements and adjuncts are attached under VP. The special tag `-CLR` (*closely related*) "marks constituents that occupy some middle ground between argument and adjunct of the verb phrase" (Bies et al., 1995). These include what other theories might analyze as subcategorized PPs or AdvPs. A handful of adjunct functions (like temporal or locative) is differentiated (see Appendix A.3 for a list of function tags). Heads are not explicitly marked.

2.3.2 NEGRA

The NEGRA project constructed a treebank for German (Skut et al., 1997a; Skut et al., 1997b). Although this thesis deals with English only and therefore does not use NEGRA material, this treebank is described here because it uses an interesting alternative to the Penn syntactic annotation. This comparison can serve to highlight some of the underlying design decisions of the Penn Treebank. The Penn Treebank uses a context-free backbone. Function tags indicate the grammatical function of a constituent and coindexed trace-filler constructions serve to indicate non-local dependencies. The NEGRA treebank combines a PS with a dependency grammar analysis (cf. Section 2.1.2). The nodes in the tree are labeled with syntactic categories whereas the edge labels denote grammatical functions. As branches are allowed to cross, no trace-filler device is necessary for discontinuous constituents/non-local dependencies. However the structures are still (discontinuous) trees, i.e. each constituent has a unique parent. Therefore so-called secondary edges are needed to indicate control constructions and structure sharing in coordinated constructions. Skut et al. (1997a) describe an algorithm for converting the NEGRA trees into standard phrase structures.

As no broad-coverage parser for German was available, the first treebank sentences had to be annotated completely manually (except PoS tags). Later a bootstrap approach was followed (Brants and Skut, 1998). With more and more annotated material, the annotation could be automated stepwise. In the simplest case, the annotator indicates which words or previously built constituents belong to a new constituent and what the new constituent's category should be. A Markov model then suggests the function labels, based on the PoS or categories of the children and the category of the parent. In the next step, the system also suggests the category label. Eventually, the system can predict the internal structure of NPs, PPs and APs of limited depth (< 3) by encoding this structure in a set of seven *chunk tags* and using another Markov model to predict the chunk tags given the PoS tags. Thus none of the models uses lexical information. Up to now, 20,602 sentences (355,096 tokens) of German newspaper text have been annotated.[10]

Some notable features of the annotation include:

[10] Source: `http://www.coli.uni-sb.de/sfb378/negra-corpus/`

- The annotation is rather flat. For example there is no extra level for finite VPs or SBARs.
- In general, heads are indicated by the HD edge label. However, not every construction needs a (unique) head. Determiners, adjectives, and nouns in NPs are all marked by the NK (noun kernel) label. This prevents problems if there is no noun in the NP that could serve as head. Verbless clauses simply have no head.
- Within coordination, all coordinated constituents carry the edge label CJ (conjunct). There are special node labels for the parent of coordinated constituents e.g. CVP for coordinated VPs.
- Punctuation is left unattached, in contrast to punctuation in the Penn Treebank. See e.g. the question mark in Figure 2.5, which is attached under SBARQ.
- Clear complements like subjects, accusative and clausal objects are marked by their edge labels (SB, OA, OC). However object and adjunct dative NPs are not distinguished (both DA). The general label MO(difier) is used for all adjuncts, but also includes prepositional objects.

The method for finding GRs that is developed in this thesis has only been applied to data from the Penn Treebank. However it should also be applicable to data derived from a treebank like NEGRA. To this end, heads, chunks and GRs would have to be defined. As heads and grammatical functions are mostly marked explicitly, they should not form a problem. The concept of chunks is trickier in a language like German (or Dutch) where premodifiers of nouns can themselves have complements, see (5), and prepositions and determiners can be merged into one word, see (6).

(5)
der	auf	seine	Tochter	stolze	Vater
the	of	his	daughter	proud	father

the father, (who is) proud of his daughter,

(6)
am	Abend
in the	evening

2.3.3 Summary

The previous two sections described two treebanks that differ in certain aspects. In summary, the Penn treebank uses trace-filler annotations for all kinds of non-local dependencies, whereas the NEGRA treebank mainly uses crossing branches for this purpose and secondary edges for control and structure sharing in coordinated constructions. Only the Penn Treebank differentiates between types of adjuncts and between PP objects and PP adjuncts of verbs. Only NEGRA marks heads explicitly.

There are more syntactically annotated corpora than the two treated here. In the CGN project a one million word corpus of spoken Dutch is being annotated in NEGRA-like style. New edge labels include PC (prepositional object), LD (locative or directional complement) and ME (measure complement) (Moortgat, Schuurman, and van der Wouden, 2001). In the SUSANNE project (Sampson, 1995) 130,000 words of the Brown Corpus were annotated with phrase structures and function tags. In the CHRISTINE project[11] a comparable number of words of spoken English was annotated.

2.4 Parsing

Parsing means assigning syntactic structure to sentences. However, what kind of structure is assigned precisely depends very much on the underlying grammar or grammar theory, or, in the case of parsers trained on treebanks, on the underlying annotation scheme. In addition it also depends on the goal of parsing. Sometimes parsing is done for a specific application. It might then be that not all the information that one normally finds in a parse tree is necessary for this application. Instead of *full parsing* one might then choose *partial parsing* (also called *shallow* or *light parsing*) which is more efficient in that it only derives the information that is needed by the application. Applications for which partial parsing has been mentioned as an alternative to full parsing include terminology extraction, lexicography, Information Retrieval and Information Extraction (Grefenstette, 1996), text summarization and bilingual alignment (Argamon, Dagan, and Krymolowski, 1998) and Question Answering (cf. Chapter 7).

In some applications, the output of parsing is used as the input for a component that builds a semantic representation of the text. These representations often use a formalism that is based on predicate logic. Subcategorization and the C/A distinction have a direct connection to semantic predicate structure, as complements and the subject denote arguments of the main verb's predicate whereas adjuncts introduce predicates of their own. The type of grammatical function determines the position of a complement within the predicate (e.g. subject is first argument, direct object is second, etc.) and might also determine the precise way in which an adjunct is integrated into the semantics of the sentence. In applications there might therefore be a separate step that takes parse trees as input and outputs lists of instantiated GRs. These lists then serve to build semantic representations.[12] Alternatively we might skip the intermediate full parsing step if our ultimate goal are lists of instantiated GRs, and extract GRs directly from sentences.

The next four sections describe the four subfields of parsing hinted at above: partial parsing (2.4.1), full parsing (2.4.2), extraction of GRs from full parse trees (2.4.3) and

[11] `http://www.cogs.susx.ac.uk/users/geoffs/RChristine.html`

[12] Note that this extra step is not necessary in LFG or HPSG-style parsers in which these representations are built simultaneously with parsing (as f-structure in LFG, or under the SEMANTICS feature in HPSG, cf. Sections 2.1.3 and 2.1.4).

direct extraction of GRs from partially parsed sentences (2.4.4). The last approach is the one taken in the remainder of this thesis.

2.4.1 Partial parsing

This section on partial parsing consists of three subsections: on chunking, PP attachment, and subject and object extraction. Chunking can either be used in isolation (e.g. for terminology extraction) or as a first step towards higher level parsing (e.g. full parsing or direct GR extraction). PP attachment focuses on a difficult subproblem of parsing as a means to compare various methods. Subject and object extraction occupies an intermediate position between partial parsing and direct GR assignment. We treat it under the former here because it does not yield a full syntactic or semantic description of the sentence and because the problem has frequently been approached with the same techniques as chunking.

2.4.1.1 Chunking

Abney Although Abney (1991) actually describes a full parser, we treat his work in this section because its major contribution is the introduction of the notion of a *chunk* and the idea of *chunking* as a first step in parsing. This idea is employed in our Memory-Based Shallow Parser (cf. Section 1.2). Abney cites psychological evidence for the existence of chunks. They are also related to prosodic patterns.

Chunks are non-overlapping continuous substrings of a sentence. They are defined through semantic heads.[13] Abney follows the CP, IP, DP and DegP analysis (cf. Section 2.1.1) so for many of his constituents the semantic head is different from the syntactic head. In practice this means that the main verb is the semantic head of a sentence or of infinitival or participial clauses, an adjective is the head of an AP, an adverb the head of an AdvP and a noun the head of an NP or a PP (of the "P+NP" type). Figure 2.6 shows an example of a sentence split up into three chunks. QPs and APs in NPs and AdvPs in APs and VPs/IPs are excluded from being chunks of their own as can be seen in the example grammar in Abney (1991).

Abney's parser consists of two parts, the chunker and the attacher. The chunker does not use lexical information. Chunk-internal ambiguity resolution, e.g. the correct bracketing of compound nouns, is left for a semantic component. The attacher uses lexical information in the form of subcategorization frames in addition to general heuristics to disambiguate the attachment. The parser is a non-deterministic LR parser that uses a hand-written context-free grammar (CFG) and employs a best-first search.

[13] The root node R of a chunk with semantic head h is the highest node in the parse tree T that has semantic head h. The syntactic structure of a chunk is the largest continuous subgraph of T that is rooted in R and that does not contain the root of another chunk. A subgraph of T is not necessarily a subtree of T as it might exclude some left or right dependents of some of its nodes.

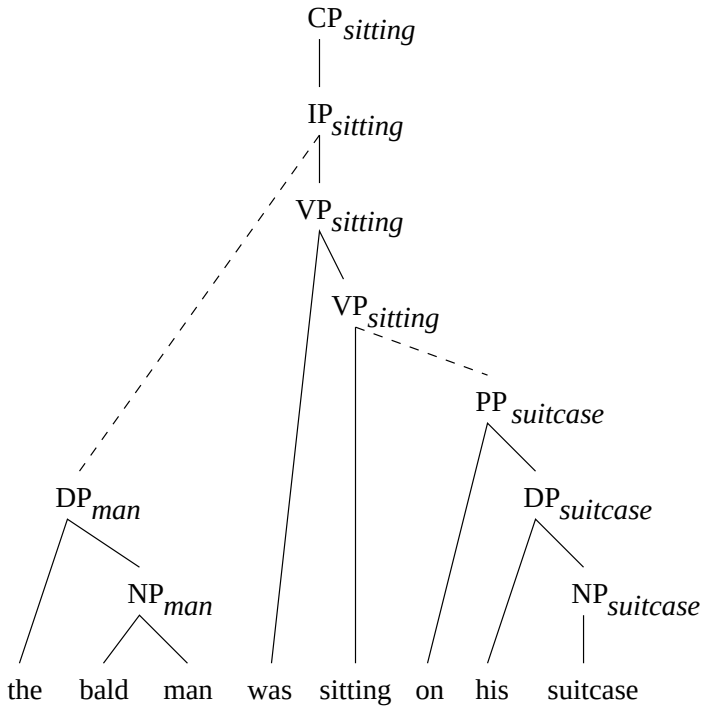

Figure 2.6: A parse tree annotated with semantic heads and split into chunks (at dashed lines): [the bald man] [was sitting] [on his suitcase].

Ramshaw and Marcus Ramshaw and Marcus (1995) use Transformation-Based Learning (TBL), see Brill (1993), to find "baseNPs", which are defined as "the initial portion of non-recursive noun phrases up to the head". Non-recursive here means "NPs that contain no nested NPs". Like in Abney (1991), possessive NPs are chunked as shown in (7).

(7) tree: (NP (NP John 's) house)
chunks: [John] ['s house]

The learner is trained and tested on material automatically derived from the WSJ. The most important contribution of the work is the definition of "chunking as tagging" (which is also employed in the chunker of our Memory-Based Shallow Parser). Ramshaw and Marcus (1995) assign each word a *chunk tag*. Possible chunk tags are I, O, and B (henceforth: IOB-tags), where *I* means that a word is *inside* a chunk, *O* means it is *outside* of any chunk, and *B* means that one chunks ends and another starts *between* this and the previous word. The following example shows a sentence with chunk brackets and with IOB tag annotation:

(8) [This] is [John] ['s house] .
This$_I$ is$_O$ John$_I$'s$_B$ house$_I$.$_O$

Inconsistent chunk tags can easily be corrected by changing B(etween) after O(utside) to I(nside).

Memory-Based Sequence Learning Argamon, Dagan, and Krymolowski (1998) present a supervised learning method called Memory-Based Sequence Learning (MBSL) and use it for finding the baseNPs of Ramshaw and Marcus (1995).[14] Whereas Ramshaw and Marcus's chunker makes decisions about elements (word/PoS tag pairs), namely which IOB tag to assign, MBSL makes decisions about sequences (of PoS tags), namely whether or not they are an instantiation of the target concept (here: baseNP). Like all memory-based approaches, it keeps the complete training material in memory, and derives test decisions directly from it. Given a test sentence, represented as a sequence of PoS, MBSL scores each subsequence (called a candidate) by computing its similarity to the training sequences. Once all candidates are scored, the final chunk division of the sentence is determined by recursively accepting the highest scoring candidate and removing all candidates that would overlap with it (as chunks have to be non-overlapping). The most important points of the method are that it uses (sub)sequences as basic units and that it does not use lexical information.

Muñoz et al. Muñoz et al. (1999) introduce another supervised learning method and apply it to the baseNP and the subject-verb task (the latter is discussed in Section 2.4.1.3). The learning algorithm SNoW (Sparse Network of Winnows) takes binary input vectors (one binary feature for each word and for each tag) and produces an output vector in which each component represents the activation (roughly: the amount of evidence) of one target node. Muñoz et al. (1999) try two different architectures. In the first, IOB tags are used. There are two classifiers, each with three target nodes (for I, O, and B). The most highly activated target node for each instance yields the final decision of this classifier. The two classifiers are *chained*, i.e. the second classifier takes the output of the first as extra input (in addition to the words and PoS).

In the second architecture, opening and closing brackets are predicted directly instead of IOB tags. Again chaining is used: the closing bracket classifier takes the predictions of the opening bracket classifier as input.[15] Predicted brackets are assigned a confidence value. The final bracket prediction is the combination of non-overlapping pairs with the highest sum of confidence values. The most interesting contribution of the work lies in this algorithm for matching brackets.

Li and Roth (2001) show that a chunker based on the architecture described above has a better performance on the chunking task and is more robust when presented with lower quality data than the state-of-the-art full parser of Collins (1997).

Veenstra, Tjong Kim Sang et al. Veenstra (1998) describes a chunker which uses the memory-based algorithms IGTree and IB1-IG (explained in Sections 4.1.1.2 and 4.1.2) to assign IOB tags for baseNP detection. Chaining helps for IGTree but not for IB1-IG. Daelemans, Buchholz, and Veenstra (1999) introduce a similar but unchained chunker for

[14] Application to subject-verb and verb-object pair extraction is discussed in Section 2.4.1.3.

[15] Unfortunately, nothing is said about the precise form of these features.

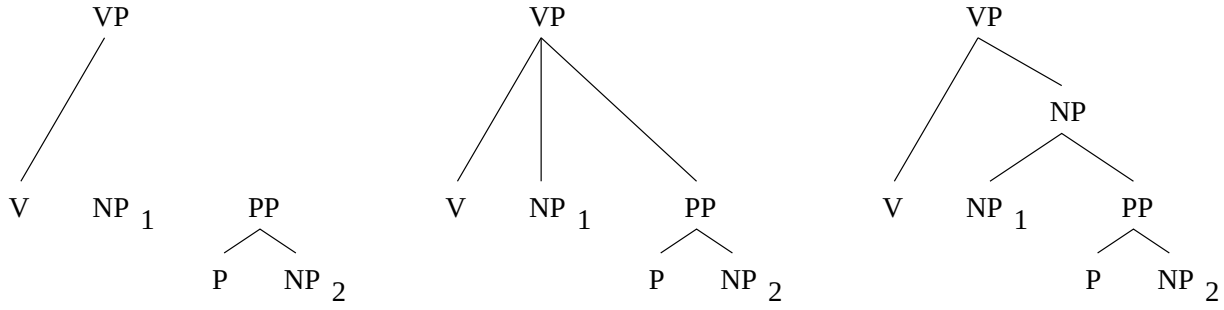

Figure 2.7: The PP attachment problem represented graphically. Given the partial structure at the left, decide whether the full structure should be like in the middle or at the right.

NP and VP chunks.[16] Tjong Kim Sang and Veenstra (1999) compare several variants of the IOB tag representation (e.g. marking all chunk-initial or all chunk-final words by a special tag) to an opening/closing bracket representation for baseNPs, using IB1-IG, chaining and classifier combination. The original IOB variant performs best but differences to other formats are not significant.

Many groups have participated in the Computational Natural Language Learning (CoNLL) 2000 shared task of chunking which involves ten types of chunks (the same as in this thesis; see Tjong Kim Sang and Buchholz (2000) for an overview). The best systems all use combinations of classifiers, a possibility that we do not pursue in this thesis (cf. Section 1.2.1). Veenstra and Van den Bosch (2000) describe an entry using only a single IB1-IG classifier but optimizing certain algorithmic parameters (described in Section 4.1.1). We use their findings for MBSP's chunker (see Section 6.1.4).

2.4.1.2 PP attachment

PP attachment is a particularly difficult subproblem of parsing and a benchmark task for machine learning systems for Natural Language Processing. In its most common formulation, a learner is given the partial structure as in the left part of Figure 2.7 and has to decide which of the two alternatives (middle and right) is the correct full structure. Note that this does not say anything about the C/A distinction. The PP can be a complement or an adjunct of the verb, or a complement or an adjunct of the noun.

Hindle and Rooth (1993) describe a system that tackles the PP attachment task using statistics on co-occurrence of two bigrams: the verb (V) and the preposition (P), and the head of the first NP (N_1) and the preposition (P). Ratnaparkhi, Reynar, and Roukos (1994) apply a Maximum Entropy (ME) model (Lau, Rosenfeld, and Roukos, 1993) to the same task; their model can use n-grams ($n \leq 4$) of the four headwords V, N_1, P and N_2 (the head of NP_2, i.e. the PP's *semantic* head). All headwords are assigned a class, and

[16] The IOB scheme can easily be extended to more than one type of chunk by having tags like I-NP, I-VP, B-NP, etc.

classes are ordered in a binary hierarchy, using Mutual Information clustering. Each word is then associated with a bit string that encodes the path from the root of the hierarchy to the word's class. The ME model can then use the values of the individual bits in the bit strings of the four headwords (V, N_1, P, N_2) as additional features. Using only the class information performs about 2% better than using only words. Using both helps for one corpus (Wall Street Journal) but not for the other (Computer Manuals).

Ratnaparkhi, Reynar, and Roukos (1994) compare their model to the performance of expert treebankers on WSJ material. In one setting the treebankers had to make their decision on the basis of the four headwords alone. They scored 10% better than the system (88.2% vs. 78.0% accuracy). In another setting the humans could read the complete sentence. Their performance then increased by another 5%. This shows that there must be useful information in the rest of the sentence.

Most of the subsequent work on the PP attachment task uses the same four headword features (Brill and Resnik, 1994; Collins and Brooks, 1995; Merlo, Crocker, and Berthouzoz, 1997; Zavrel, Daelemans, and Veenstra, 1997). Collins (1999) notes that ignoring the N_2 feature leads to a 1.1% decrease in performance in his PP attachment experiments. Note that the learning task described above is only a subset of the general PP attachment problem. For example, the same ambiguity arises in a context of "V PP PP", and in general there might even be other attachment sites for the PP outside the VP. In addition, the attachment of subordinate clauses poses problems similar to PP attachment (Yeh and Vilain, 1998).

2.4.1.3 Subjects and objects

Grefenstette Grefenstette (1996) describes a cascade of finite-state transducers (FST) to perform what he calls "light parsing". The input to the system is automatically PoS tagged text. The first transducer inserts markers for the beginnings and ends of noun and verb groups (e.g. `[VC VC]`). No formal definition of noun and verb group is given but the examples suggest that verb groups correspond to sequences of Abney-style verb chunks e.g. "[VC did not appear to affect VC]" whereas noun groups are basically noun phrases, i.e. in contrast to noun chunks they can contain coordinated NPs and include arguments and postmodifiers. In addition, a preposition preceding an NP is also included in the noun group. The second transducer marks the heads of noun and verb groups. Within noun groups, it distinguishes nominal heads of NPs from those of PPs, within the verb group, it distinguishes active, passive and copula heads. In addition, non-finite verbs in noun groups are marked as past or present participles or infinitives. In the case of coordination, multiple heads are allowed. The third and last transducer extracts surface subject-verb pairs.

Aït-Mokhtar and Chanod Aït-Mokhtar and Chanod (1997a) describe a similar FST system that can also recognize (limited) recursive structures and more grammatical func-

tions in French text. The input is again PoS tagged text. Simple groups (AP, AdvP, infinitival and present participle verbal chunks) are matched by single regular expressions. For more complex groups (NP, PP) potential beginnings and ends are marked first. These are then paired into definitive boundaries by choosing the longest possible sequence that does not include another potential end. The most complicated case, verb groups, which can be recursive, uses different kinds of begin markers to indicate different degrees of certainty. Sure beginnings are paired to ends first. In addition, verb group assignment builds on the previously found NPs, APs and PPs. After group finding (chunking), grammatical functions are assigned. No complete list of functions is given, but subject, direct object, PP objects and verb modifiers are mentioned.

Aït-Mokhtar and Chanod (1997b) basically describe the same system as Aït-Mokhtar and Chanod (1997a) but explain grammatical function extraction in greater detail and concentrate on subjects and objects. These are recognized in two steps: first groups/chunks are marked as subjects or objects in isolation. This is called *function tagging*. Next it is decided which is the subject or object of which verb (or verbs, in case of verb or VP coordination). The system can handle limited recursion, as in "[VC L'usine, [VC que le ministre devrait VC] [VC implanter VC] à Eloyes, représente VC] ...". To do this within the finite state calculus, verb groups at different levels of embeddings are matched by different regular expressions. Subject-verb and object-verb pairs are then only extracted from the same level. The main difficulties concern subjects and objects of embedded sentences or with coordinated verbs. Other phenomena treated include inverted subjects and relative clauses.

Memory-Based Sequence Learning, Muñoz et al. Argamon, Dagan, and Krymolowski (1998) and Muñoz et al. (1999) also applied MBSL and SNoW to the task of finding subject-verb (SV) pairs. Both use the same data set, which was extracted from the WSJ corpus using the `tgrep` tree extraction script (Argamon, Dagan, and Krymolowski, 1998). Argamon, Dagan, and Krymolowski (1998) also report on extracting verb-object (VO) pairs. SV pairs are annotated as non-overlapping substrings of a sentence, from the beginning of the subject noun phrase up to and including the first non-modal verb. VO pairs extend from the main verb to the head of the object. Thus in contrast to the task definition underlying the previously discussed work, heads are not explicitly extracted, pairs cannot be nested, subjects cannot be inverted, objects cannot precede the verb (as they do in relative clauses), and only one relation is extracted in cases of coordination. In addition, training as well as test set contain the *traces* from the treebank annotation. Argamon, Dagan, and Krymolowski (1999) also include results without the traces, which are worse than the original ones, especially for VO extraction.

Krymolowski and Dagan (2000) and Dagan and Krymolowski (2002) present a compositional extension of the algorithm in which already found NP and VP phrases (not only chunks) contribute to NPs or VPs of a higher level. However this approach has not been tested on the old SV/VO data. Krymolowski and Dagan are currently working on apply-

ing this method to general (unlabeled) dependency extraction (Krymolowski and Dagan, 2002). We compare MBSP and MBSL for dependencies in Section 6.2.1.

Daelemans et al. Daelemans, Buchholz, and Veenstra (1999) describe a memory-based system that extracts also embedded, inverted and coordinated SV and VO pairs. These relations are defined as pairs of headwords, not sequences. The system first applies the Memory-Based Tagger to text and then a memory-based chunker (using either IGTree or IB1-IG) that finds NP and VP chunks. Finally SV and VO relations are predicted for pairs of a VP and another chunk, again using IGTree or IB1-IG or a combination of both. Prediction is only attempted for pairs that have at most one other VP chunk between them (but evaluation is on all pairs).

2.4.1.4 Summary

This section described work on chunking, PP attachment, and subject and object extraction. The information that was used to solve these tasks is rather simple (see Table 2.1). Chunking is mainly done on the basis of PoS tags and/or words. Depending on the architecture the algorithm might also use chunk annotation that it assigned in earlier steps (TBL, chaining). Only limited local context is used, but actual context size differs. Different systems use different types and definitions of chunks/groups and their heads. Work on PP attachment mostly uses the four headwords. The P and N_2 features correspond to the syntactic and semantic head of the PP respectively. Ratnaparkhi et al. introduce a bit string encoding of word classes. Chunking might or might not be used as a first step towards subject and object extraction. Grefenstette is the only one to distinguish various verb types. Only Daelemans et al. use a distance measure. The following phenomena constitute problematic cases for chunking and subject and object extraction: possessives, coordination, inversion, embedding and non-local dependencies.

2.4.2 Full parsing

Full parsing has been studied for decades and there is a vast amount of literature. In this section we concentrate on recent trained, lexicalized parsers, as these are most comparable to our parsing framework. Most of these parsers are also trained and tested on the Penn Treebank (like ours) and all of them are probabilistic (unlike ours). Probabilistic parsers assign probabilities to parts of parses. The probabilities of the parts are then (usually) multiplied to compute the probability of the parse. The parse with the highest probability is returned as the parse of the sentence. Models differ as to how they split up the parse into parts and what information is used to condition the probabilities on.

One of the simplest probabilistic models is a Probabilistic Context-Free Grammar (PCFG). In this case, the parse is a PS tree. The parts of the parse are all the instances of rules used to derive the tree. The rule probability is conditioned on the left-hand side of the rule,

Reference	Method	Modules	Information
		Chunking	
Abney '91	LR parser, hand-written CFG	chunker, (attacher)	PoS, structure built by prev. rule applications, (chunks and subcategorization for attacher)
Ramshaw '95	TBL	chunker	word, PoS & prev. assigned IOB tag of current and 3 left/3 right context words
Argamon '98	MBSL	chunker	candidate PoS sequence, 3 left/3 right context PoS
Muñoz '99	SNoW	first and second level chunker	word, PoS & prev. assigned IOB tag or "[" of current and 3 left/3 right(?) context words
Veenstra '98	MBL	first and second level chunker	word, PoS & prev. assigned IOB tag of current and 2 left/1 right context words
Daelemans '99	MBL	chunker	word & PoS of current and 5 left/3 right context words
		PP attachment	
Hindle '93	co-occ. stat. of bigrams		V, N_1, P
Ratnaparkhi '94	MaxEnt		V, N_1, P, N_2, bit string encoding of word classes
	treebankers		only V, N_1, P, N_2 or whole sentence
		Subject-Verb/Verb-Object Extraction	
Grefenstette '96	FST	NP/VP grouper, head marker, SV extractor	PoS, beginning/end of noun/verb groups, verb types (active, passive, copula; present/past participle, infinitive)
Aït-Mokhtar '97	FST	grouper, function tagger, SV/VO extractor	PoS, beginning/end of various groups, recursion level
Argamon '98	MBSL	SV/VO extractor	candidate PoS sequence, 3 left/3 right context PoS, (traces)
Muñoz '99	SNoW	first and second level SV extractor	word, PoS & prev. assigned IOB tag or "[" of current and 3 left/3 right(?) context words, (traces)
Daelemans '99	MBL	(tagger), NP/VP chunker, SV/VO extractor	headword & head's PoS of VP chunk, other chunk and its 2 left/1 right context chunks, distance (in chunks) between VP and other chunk, other intervening VP chunks, intervening commas

Table 2.1: An overview of the partial parsing work described in this section. The last column shows the information on which the parser bases its decision (co-occ. stat. = co-occurrence statistics, prev. = previously).

i.e. the parent category. All the parsers described in this section improve upon the simple PCFG model. In particular, they are all lexicalized, mostly by conditioning probabilities on words.

2.4.2.1 Black et al.

As explained in Section 1.2, Black et al. (1992) introduced the concept of history-based grammars (HBG) in which all the information in the history (previously parsed sentences, read words and built-up structure) can influence every following parse decision. To prevent their generative probabilistic parser from suffering from too sparse data, Black et al. (1992) suggest training decision trees on the parse decision problem and using the equivalence class of the history under the decision tree as conditioning information. For practical reasons, they did not implement the full model. The system's history only contains information from the current node and its *immediate* and *functional parent*. The immediate parent is the constituent that immediately dominates the current node C. The functional parent is the lowest ancestor that has a different syntactic category than C.[17] Five features are associated with each node and only for the prediction of one of them is the decision tree approach used.

The features are the syntactic category, the rule, the semantic category and *two* lexical heads. Nothing much is said about the semantic category (there are about 50, they are assigned manually and they seem to encode (at least partially) domain-specific knowledge). "The primary lexical head H_1 corresponds (roughly) to the linguistic notion of a lexical head. The secondary lexical head H_2 has no linguistic parallel. It merely represents a word in the constituent besides the head which contains predictive information about the constituent." For the PP in their example, the primary head is the head noun of the PP, and the secondary head is the preposition. For the NP, the primary head is again the head noun, and the secondary head is the determiner. So it seems that the two heads correspond to the syntactic versus semantic heads proposed in linguistic theory.

Each feature is encoded in a bit string. Bit strings for syntactic and semantic categories and rules have been assigned manually, those for the words (the heads) are derived by automatic clustering. The parser uses a manually developed unification grammar and is tested on a computer manuals treebank.

2.4.2.2 Magerman

Magerman (1995) describes SPATTER (Statistical PATTErn Recognizer), a bottom-up parser that uses probabilistic decision trees. Each node in the tree carries four pieces of information: the headword, the head's PoS, the syntactic category (not for preterminals)

[17] In many cases the two parents are identical but with unary rules like N2 → N1, the immediate parent of N1 is N2, but the functional parent might e.g. be a VP. Black et al. (1992) do not mention this, but the parent distinction might also be useful for coordination.

and the "extension". The extension is similar to the IOB tags in chunking. There are five different values: for a first child of a constituent, for a last child, for any child that is neither first nor last, for a single child, and for the root node. There is one decision tree model to assign these extensions and another one that decides about the syntactic category of new nodes. PoS tagging is done by a third decision tree model.

SPATTER is the first parser that was trained and tested on the WSJ Corpus. Magerman also introduced the head table, which determines the head child of each rule and thus, when applied recursively, the head of a constituent.[18] For example, the preposition is the head of a PP and the leftmost verb (even a modal or auxiliary) or infinitival marker *to* is the head of a VP, S or SBAR.

2.4.2.3 Eisner

Eisner (1996b) describes several models for probabilistic dependency parsing.[19] The dependency structures are required to be projective, and dependencies are unlabeled. However, as the words are assigned parts-of-speech, Eisner sometimes talks about an NN-VB dependency, for example. He also claims that the same model could be used with labeled dependencies. The dependency structures are derived semi-automatically from the WSJ corpus. Sentences with coordination are excluded.

Eisner's way to determine heads is partially different from Magerman's and Collins's. For example, he considers the main verb the head of the verb phrase. He also "unflattens" some treebank structures by introducing extra constituents for sequences of proper nouns, common nouns, numbers or for the dollar token followed by a quantifier phrase. In addition, he automatically refines some PoS tags, by marking auxiliaries, premodifiers of nouns, and participial postmodifiers of nouns, and by distinguishing complementizers from prepositions (cf. our discussion of the Penn Treebank tag set in Section 2.3.1). He also uses two coarser versions of the tag set ("short" and "tiny" tags, with 22 and 7 tags, respectively) in the back-off probabilities. For example, both JJR and JJS (comparative and superlative adjective) are conflated to "adjective". Unknown words are "attenuated", i.e. replaced by a symbol that indicates some morphological properties of the word (digits, capitalization, last two letters, length). We will not go into the details of the three (Eisner, 1996b) respectively four (Eisner, 1996a) probabilistic models that Eisner compares.

2.4.2.4 Collins

Collins (1996) also uses a probabilistic dependency parser. The dependency labels are explicitly derived from the syntactic labels on the PS trees of the WSJ. A label is a trigram consisting of the category (or PoS) of the head, the category of its parent and the category (or PoS) of the dependent at the level where attachment occurs. Dependency

[18] See `http://www.research.att.com/~mcollins/papers/heads`

[19] Eisner (1996a) is a more detailed technical report.

structures are thus projective. Collins uses a modified version of Magerman's head table. For example the complementizer, not the verb is the head of an SBAR. This models the CP analysis (cf. Section 2.1.1). The labeled dependency output is mapped back to treebank trees for evaluation. Parsing is separated into three steps: tagging, NP chunking and dependency finding, where each step can use the output information from the previous steps (the history).

The NP chunker uses a probabilistic variant of IOB chunking with five tags (Begin, Inside, Between, End, Outside) instead of the minimal three. After chunking, all non-headwords of chunks are ignored. This reduces the set of possible dependencies to be considered. The same technique is used in our Memory-Based Shallow Parser. The dependency model conditions on words, PoS and the "distance", which combines information about the relative order (does the dependent precede or follow the head), adjacency, presence or absence of another verb between head and dependent, the number of commas (0, 1, 2, $>$ 2) between head and dependent and whether there are commas in special positions (immediately preceding or following the head respectively dependent).

Collins (1997) introduces a generative version of Collins (1996) which uses 0th order Markov processes. Generation of children starts with the head child and continues with the leftmost right sibling of the head, the second leftmost right sibling, and so on, until the special STOP child is predicted, which prevents generation of further children to this side. The left siblings of the head are then predicted in a symmetric way, from innermost to outermost (STOP).

The generative model just described is called Model 1. Its output are simple PS trees. Model 2 makes the complement/adjunct distinction by adding a tag -C, respectively -A, to the category labels of constituents that function as complements, respectively adjuncts. It does so by probabilistically generating a left and right part of a subcategorization frame for each head. Complements that are generated are deleted from the frame. Generating complements that are not in the frame is illegal, and the frame must be empty when the STOP child is generated. Model 3 of Collins (1997) resolves NP extraction from SBAR relative clauses[20] by distinguishing non-terminals which contain a gap from those without, by probabilistically propagating the gap through the tree (similar to GPSG) (Gazdar et al., 1985) and by discharging it if a trace is generated.

Collins uses special features for coordination and punctuation and introduces an additional syntactic category, SG, for clauses with an empty subject (mostly infinitives). The latter enables the algorithm to learn that infinitives behave syntactically very differently from finite clauses.

[20] An example for non-NP extraction is "This is the place where$_i$ I met him __$_i$". Collins's example for an infinitival relative clause is "I called a plumber$_i$ __$_i$ to fix the sink". Traces can also indicate control relations like in "The visitor$_i$ promised (him) __$_i$ to return."

2.4.2.5 Charniak

Charniak (1995) describes a method for lexicalizing a PCFG. It consists of two parts: defining heads and conditioning probabilities on heads (amongst others). The definition of heads is given only informally but seems to work similarly to Magerman's head table. The model can represent PP attachment preferences by using the head of the grandparent's node (V or N_1) as additional conditioning information. Probabilities are collected from text parsed with a CFG induced in earlier work. Charniak (1997) also uses a lexicalized PCFG. The grammar and the probabilities are directly induced from the WSJ treebank. Classes of words (acquired by clustering) are used as a back-off term for the words themselves. This is similar to the use of (manually assigned) categories in Black et al. (1992).

Charniak (1999; 2000) describes a new "maximum-entropy-inspired" model for conditioning and smoothing that easily allows for testing more conditioning events. Charniak (1999) uses a "Markov grammar" instead of a "treebank grammar". This is the same technique as that used by Collins (1997). However, whereas Collins conditions the generation of siblings only on the head child, Charniak experiments with different orders of Markov grammars. In a third order Markov grammar, which proved to work best, the generation of a left, respectively right, sibling is conditioned on the three previously generated left, respectively right, siblings (amongst other information). Two other subtle differences with other parsers are the explicit marking of auxiliaries and of coordinated structures.

2.4.2.6 Ratnaparkhi

Ratnaparkhi (1997) describes a statistical parser based on Maximum Entropy models (Lau, Rosenfeld, and Roukos, 1993). Like Magerman's, it is a bottom up parser. It consists of three separate modules which are applied in sequence: a tagger, an NP chunker, and a sort of shift/reduce parser. Instead of a simple shift, the parser determines to which category the unit (word or chunk or constituent) on the top of the stack should eventually be reduced. This is called BUILD. After that, it decides whether to reduce now to the previously determined category, or to shift the next unit, i.e. wait for more children. This is called CHECK.

2.4.2.7 DOP

In contrast to all the other parsers in this section, the Data Oriented Parsing (DOP) model (Scha, 1992; Bod, 1992) does not restrict the possibility of probabilistic (inter)dependencies to fixed, local configurations in a tree (e.g. parent/child, or sibling/sibling). The basic unit to which probabilities are assigned are subtrees. The parse trees in the training treebank are split up into subtrees in all possible ways which do not violate the unit of a rule. The unsmoothed probability of a subtree T with root R is the frequency of T (in the training data), divided by the total frequency of all subtrees with root R. For parsing, subtrees can be combined into larger trees by substituting a subtree with root X on another subtree

with leaf X. These subtrees then define one derivation of the larger tree. The probability of a derivation is the product of the probabilities of its subtrees. As a full tree can have more than one derivation, the probability of a full tree is the sum of the probabilities of its derivations. Note that in all other parsers, each tree had just one possible derivation. Subtrees can be lexicalized, i.e. they contain one or more words, or unlexicalized, i.e. they contain only non-terminals.

Bod (2001) shows that in general, subtrees as deep as fourteen levels (deeper trees were not tested) still contribute to parse performance. Only excluding lexicalized subtrees which contain more than twelve words or unlexicalized subtrees deeper than six levels increases performance. Excluding subtrees with more than eight non-headwords[21] at least does not influence performance negatively. These results show that structural and lexical dependencies span many more levels than other parsers assume. However they do not tell us whether *all* words in *all* constructions have this property or whether this is only a limited group.

2.4.2.8 Summary

As Table 2.2 shows, most full parsers in this section base their decisions on heads, PoS, (linear order of) chunks and/or syntactic categories. Heads are syntactic heads for some constituents and semantic heads for others, only Black et al. (1992) always use two types of heads. They also use a bit string encoding of the head (and other) information. Once the two bottom-up models have constructed a constituent, they only use this constituent's category and head (and Magerman also the head's PoS) as information when building higher structures. Likewise the generative models mainly use the parent's head, PoS and category. Black et al. (1992) also use information from the functional parent, and Charniak from the grandparent. Information from siblings and neighbours is also restricted to the two or three previously generated or built left and right ones. Thus the used information is mostly tree-local or string-local and focuses on heads. Exceptions are the distance features in the dependency parsers of Eisner and Collins, and the DOP model.

Some approaches use special mechanisms for the following phenomena: coordination, clauses with empty subjects, relative clauses, auxiliaries, punctuation, unknown words, and groupings of PoS.

2.4.3 Extracting grammatical relations after full parsing

The three approaches discussed in this section all apply an additional module to the output of a full parser. They differ however in how much of the information that is necessary to extract GRs is determined by the parser and how much is provided by the extra module.

[21] Non-headwords are all words except the headword of the subtree's root.

Reference	Method	Modules	Information
	PCFG	parser	rule: category
Black '92	generative, decision tree for equivalence	parser	syn. & sem. cat, rule, heads: bit strings for syn. & sem. cat, rule, index of node in rule, primary & secondary head of current node and/or immediate & functional parent
Magerman '95	bottom-up, decision tree	tagger, parser	extension & cat: head (30 binary features), head's PoS, cat, extension, number of children and number of words spanned of current node, of two left/right neighbours & of two left/rightmost children
Eisner '96	unlabeled dependencies, Model C generative	tagger, dependencies	presence or absence of dependency (Model A) / preferred parent & children (Model B) / ith left/right child (Model C): word & PoS of head & dependent, number of intervening children & words (distance), PoS of left/right sibling, relative order of head & dependent (direction), short & tiny tag, attenuation
Collins '96	labeled dependencies	tagger, NP chunker, dependencies	dependency: (word or chunk head) & PoS of head & dependent, distance (direction, adjacency, intervening verb, commas)
Collins '97	generative, 0th order Markov grammar for dependencies	tagger for unknown words, NP chunker integrated in parser	head child cat: parent's cat, head, head's PoS; non-head child cat & PoS: above & head child's cat, distance (Model 2 & 3: left/right subcategorization frame, Model 3: gap feature); non-head child's head: above & head child's cat & PoS
Charniak '95	generative, lex. PCFG	parser	rule: cat(?), head; head: cat, parent's head, grandparent's head (if parent is PP), direction
Charniak '97	generative, lex. PCFG	parser	rule: cat, parent's cat, head & word class; head: cat, parent's cat, parent's head & word class
Charniak '99	generative, 3rd order Markov grammar	parser	PoS: cat, parent's head & PoS & cat, grandparent's cat; head: PoS, cat, parent's head & PoS & cat, grandparent's cat; cat: parent's head & PoS & cat, cat of 3 prev. generated siblings
Ratnaparkhi '97	bottom-up, ME	tagger, NP chunker, parser	BUILD: head & (cat or PoS) & prev. BUILD decision on current node & on two left/right neighbours, features on parentheses, commas, periods; CHECK: proposed rule, head & (cat or PoS) of children & of two left/right neighbours
Bod '92	DOP	parser	subtree probability: root cat

Table 2.2: An overview of the full parsing work described in this section. The last column shows the information on which the parser bases its decision (cat = category, dep. = dependency, prev. = previous(ly), "X: Y, Z" = parse decision X based on Y and Z).

2.4.3.1 Blaheta and Charniak: function tag assignment

Most parsers that we have described in the previous section assign labeled trees to sentences, where the labels are syntactic categories like NP, VP, or S. However the second version of the Penn Treebank uses also function tags (`-SBJ`, `-TMP`, etc.). Blaheta and Charniak (2000) present a model for assigning these tags to the output of the Charniak (1999) parser. The conditioning information is basically the same as in the parser. The only new feature is the "alternate head" (and its PoS), which corresponds to the N_2 of PP attachment. Thus the features used for determining the kind of (grammatical) relation are nearly identical to the ones used for deciding whether there is a dependency relation in the first place. This is an argument for the integration of the two predictions. Indeed Blaheta and Charniak (2000) note that "there is no reason to think that this work could not be integrated directly into the parsing process, ...; the function tag information could prove quite useful within the parse itself, to rank several parses to find the most plausible." Once function tags are assigned, GRs could be extracted from trees in much the same way as we do for the original treebank trees (Section 3.1.2). However Blaheta and Charniak (2000) do not predict traces, so only local relations could be extracted.

2.4.3.2 Carroll et al.

Initially the GR extraction system of Carroll, Minnen, and Briscoe (1998) was used to evaluate the parser on precision and recall of GRs instead of labeled brackets. This new evaluation method was proposed by Lin (1995) and also used in the SPARKLE project (Carroll et al., 1997). In the system, text is tokenized, lemmatized, PoS tagged and then parsed by a probabilistic LR parser (the probabilities are derived from a treebank) using a hand-written, unification-based grammar which, following $\overline{\text{X}}$ (cf. Section 2.1.1) explicitly distinguishes complements from adjuncts through their attachment to different bar levels. The parser is basically unlexicalized, only information about subcategorization frame frequencies is used to rerank parses.[22]

The set of target GRs is different from the one defined by the function tags of the Penn Treebank e.g. it does not encode semantic distinctions of adjuncts. It is described in more detail in Section 6.2.2 where we compare the two relation finders. All relations hold between two (mostly semantic) heads. Some have additional slots for syntactic heads like prepositions or complementizers, and for underlying relations e.g. before passivization or dative shift.

The mapping from parse trees to GRs is performed by special-purpose software that uses information about which grammar rules introduce which GRs and which children provide the head and the dependent of the relation, or the filler for the additional slot. Only very few of these rules encode lexical information (an exception are rules for the verb "be"). As the grammar contains rules for *wh*-extraction and control verbs, also non-local relations can

[22] This method is more fully described in Section 2.6.6.

be extracted. Carroll, Minnen, and Briscoe (1998) report on extraction of arguments only, Carroll, Minnen, and Briscoe (1999) also include modifiers. Carroll and Briscoe (2001) describe a method in which relations are extracted from the top n parse trees returned by the parser. Each extracted relation is weighted by the number and probability of the parses it appears in. By varying a threshold on the summed weight of relations, Carroll and Briscoe (2001) are able to control the precision/recall trade-off of GR extraction.

2.4.3.3 Kübler

Grammatical relations as a means for parser evaluation have received much attention lately (Crouch et al., 2002; Clark and Hockenmaier, 2002; Briscoe et al., 2002) In this context, Kübler and Telljohann (2002) describe a method for extracting GRs from the output of the robust parser TüSBL (Kübler and Hinrichs, 2001b) for German. TüSBL uses a memory-based approach, as it stores complete training trees in memory. It is robust in the sense that it will output only a partial parse (with some constituents left unattached) for a test sentence if the memory does not contain enough evidence for a full structure. It works in three steps: PoS tagging, chunking and tree construction. The chunks discussed in Section 2.4.1.1 were defined as "not containing another chunk". By contrast, the chunks used in TüSBL are defined as "not containing another chunk of the same type" (Hinrichs et al., 2002). This allows for simple clause chunks that contain e.g. NP and VP chunks. Chunking is performed by a cascade of finite-state transducers (Abney, 1996). Based on the chunks of a test sentence, the tree construction algorithm searches for the most similar training tree in memory and uses it to construct a parse for this sentence. Ideally similarity is achieved at the word level and for complete chunks but otherwise the algorithm backs off to PoS or subchunks. If there is enough evidence, the tree construction module can correct earlier tagging or chunking errors (Hinrichs et al., 2002).

The parser is trained and tested on an English and a German treebank of transcribed spoken dialogues. The treebank annotation is similar to NEGRA in that it also has syntactic node labels and functional edge labels and explicitly marks heads. Therefore converting trees into dependencies after parsing is relatively straightforward. However the German treebank annotation does not use crossing branches, secondary edges or trace-filler-constructions. Non-local dependencies are expressed through functional labels on the (conceptual) filler such as `OA-MOD` (modifier of accusative object). Sometimes these labels do not uniquely identify the phrase that (conceptually) contains the trace, e.g. there might be two accusative objects. In that case heuristics determine the most likely underlying attachment position prior to GR extraction (Kübler and Telljohann, 2002). Control relations are not annotated in the treebank, so they are not extracted. The functional labels for German differentiate various types of complements but make no semantic adjunct distinctions.

2.4.3.4 Summary

This section described three different approaches to detecting GRs in parse trees. The parser used by Blaheta and Charniak (2000) does not distinguish complements from adjuncts, so this distinction (implicitly) has to be made when assigning the function tags. In addition the tags also differentiate between various semantically defined adjunct types. The method employs a machine learner and uses lexical heads as features (in addition to constituent labels and PoS). The parser used by Carroll, Minnen, and Briscoe (1998) already encodes the C/A distinction in the parse tree. The three types of adjuncts reflect only syntactic differences. The hand-crafted GR extraction system is mainly based on the identity of the parse rules in the tree. The parser used by Kübler and Telljohann (2002) already assigns function tags, which also distinguish complements from adjuncts, but make no semantic adjunct distinctions. It uses words, PoS and chunks. The extraction procedure is mainly based on the function tags but uses heuristics for resolving non-local dependencies.

2.4.4 Extracting grammatical relations after partial parsing

2.4.4.1 Buchholz et al.

Buchholz, Veenstra, and Daelemans (1999) describe an extension of the Memory-Based SV/VO extraction system to all types of GRs to verbs. As before, text is PoS tagged with MBT. Then IB1-IG is used to find five types of chunks: NP, VP, AdjP, AdvP, PP. The latter contain only prepositions (possibly multi-word) and their premodifiers. Next an IGTree combines PP chunks and NP chunks into PNP chunks (also used in this thesis) and another IGTree predicts whether a chunk has one of six adverbial functions[23]. That module is not used in this thesis. Finally GRs are assigned by another IGTree to pairs of a verb chunk and another chunk (the *focus* chunk). As before, the headword of a chunk and the head's PoS tag are used, for the verb chunk, the focus chunk, and its two left and one right context chunk (or chunkless word). Also the distance, the intervening verb chunks, and the intervening commas are counted. Additional features are the chunk type (for the focus and its context) and the preposition of a PNP chunk and the predicted adverbial function (for the focus only). The system was trained and tested on WSJ text. A breakdown of performance for some selected relations shows that locative adjuncts are especially difficult.[24]

[23] LOC (locative), TMP (temporal), DIR (directional), PRP (purpose and reason), MNR (manner), EXT (extension)

[24] While the harmonic mean of precision and recall ($F_{\beta=1}$) is over 80 for subjects and objects, it is 63 for temporal adjuncts and only 47 for locative adjuncts.

2.4.4.2 Ferro et al.

Ferro, Vilain, and Yeh (1999) describe a system that uses Transformation-Based Learning (Brill, 1993) to find GRs between headwords of chunks (i.e. also GRs to non-verbs). Raw text is automatically split into sentences, tokenized, PoS tagged (using Brill's tagger), and manually annotated for NP, VP, AdjP, AdvP and *IN*[25] chunks. NP chunks are further divided into Named Entity (NE) types like person or organization, and passive, infinitival or present participle VP chunks are marked. Attachment of prepositions and subordinating conjunctions is automatically estimated (Yeh and Vilain, 1998). All this information forms the input to the actual GR finder. In addition, some external resources are used: a lexicon for possible stems of words, WordNet (Miller et al., 1990) for semantic class, COMLEX Syntax (Grishman, Macleod, and Meyers, 1994) for subcategorization information, and lists of relative pronouns and partitive quantities (e.g. *some*).

The learner starts with an empty annotation of GRs and then learns rules to create or delete relations between a source and a target chunk (dependent and head, respectively). In principle, the rules can use all of the above information sources in positive as well as negative tests.[26] However, to keep the search space for possible rules manageable, certain restrictions were used (found by development on the training data). The most interesting ones are:

- The system searches for relations from source to target chunks that are at most three chunks away. 95% of the relations in training material fulfill this constraint.
- Conditions can only test the properties of the source chunk, the target chunk, their immediate neighboring chunks and the chunks in between source and target.
- Only a very limited number of words (*of*, "?", some determiners) other than the headword can appear in conditions. Tests on the PoS of non-headwords are allowed however.

About two dozen GRs are distinguished. No complete list is given, but argument relations include deep subject and object, surface subject and object of copulas, prepositional object, and directional object. Modifier relations include time and location as well as a generic modifier (not fitting any of the other classes). The method was trained and tested on a small body of elementary school reading comprehension tests. Note that these texts are probably less complex than for example the WSJ. The biggest problem is the distinction between locative, temporal and "other" adjuncts.

Yeh (2000a) compares the system of Buchholz, Veenstra, and Daelemans (1999) (without the PNP and adverbial function modules) to that of Ferro, Vilain, and Yeh (1999) on the training and test data of the latter on the task of finding GRs to verbs. He tries to find out

[25] IN chunks contain prepositions or subordinating conjunctions. The name derives from the PoS tag for these words in the Penn Treebank.

[26] e.g. headword is *house*, or headword is not *house*

systematically how much of the perceived performance differences can be explained by the different algorithms or by different feature representations. It turns out that none of the WordNet, COMLEX, stem, Named Entity, or PP/conjunction-attachment features seem to have much effect on the performance of the Ferro, Vilain, and Yeh (1999) system (i.e. some have an insignificant positive and some even a negative effect). The VP properties are only relevant for complement GRs, not for adjuncts. The non-headwords are irrelevant for complements (nothing is said about their relevance for adjuncts). The Buchholz, Veenstra, and Daelemans (1999) system is only better in finding adjuncts. Yeh (2000a) suggests that this is due to the additional feature that counts verb chunks between two potentially related chunks. However this feature does not seem to be necessary for finding complement relations. In addition, using several classifiers, each specialized in a particular focus chunk type, a particular distance and a particular relation improved performance.

Yeh (2000c) uses the same system as Ferro, Vilain, and Yeh (1999) but instead of starting from an empty GR annotation, the system starts from the output of either Buchholz, Veenstra, and Daelemans (1999) or Carroll et al. (1997) or a merge of both. This set-up improves performance, especially if the training set is small.

2.4.4.3 Summary

Both approaches to GR extraction described in this section operate on tagged and chunked text. Chunks are used to define context, measure distance, compress the representation and restrict the search space. The type of chunk, its headword and that head's PoS are used as features. Ferro, Vilain, and Yeh (1999) also use limited information from non-heads. In addition they use many external information sources. However these do not seem to contribute to performance. Difficult cases include locative, temporal and general adjuncts.

2.4.5 Summary: parsing

The work described in this section is diverse. Information used for parsing varies from headwords only (PP attachment) or PoS only (MBSL) over tree-local category and head information (lexicalized parsers) to many external sources (Ferro et al.). However no system uses all of this information at the same time. The challenge then is to choose from all the available information those parts that are relevant and to find a good way to combine them. Coordination, inversion, embedding, non-local dependencies, and types of adjuncts seem to cause most problems.

2.5 Subcategorization dictionaries

The parsers we have seen in Section 2.4.2 do not use external dictionaries. With the exception of Collins's model 2 and 3, they do not make the C/A distinction for which subcategorization is crucial. However, subcategorization information can even be useful if this distinction is not made, as it also influences attachment, which these parsers do resolve. Compare the following pairs of sentences which differ in the subcategorization of the verbs involved and, as a result, in their syntactic structure:[27]

(9) John $[_{VP}$ put $[_{NP}$ the cactus] $[_{PP}$ on the table]].
John $[_{VP}$ likes $[_{NP}$ $[_{NP}$ the cactus] $[_{PP}$ on the table]]].

(10) I expected $[_{NP}$ the man who smoked] to eat ice-cream.
I doubted $[_{NP}$ the man who liked to eat ice-cream].

If constructions like these occur often enough in a treebank, the parsers from Section 2.4.2 would implicitly learn these kinds of regularities too. However, these are undistinguishable from regularities that do not involve subcategorization as long as the annotation scheme does not distinguish complements from adjuncts or one type of GR from another. Parsers are not the only application for subcategorization dictionaries. They are also needed for natural language generation, language teaching programs and human foreign language learners.

2.5.1 COMLEX Syntax

COMLEX Syntax (Grishman, Macleod, and Meyers, 1994) is an electronic "moderately-broad-coverage" English lexicon. It contains detailed syntactic information for approximately 38,000 headwords. It uses 92 subcategorization frames for verbs, 14 for adjectives and 9 for nouns. The frames record the constituent structure as well as the grammatical function of complements and distinguish between four types of control.[28] The initial lexicon is constructed manually using corpus citations, intuitions and definitions and citations from several printed dictionaries. Meyers, Macleod, and Grishman (1994) and (1996) describe the criteria that were used to decide on the complement-hood or adjunct-hood of some dependent of a verb. Criteria for complement-hood are divided into sufficient criteria and "rules of thumb". This helps to resolve conflicts between the outcomes of different

[27] First sentence of (9) from Manning (1993), sentences in (10) from Brent (1991b).

[28] **Subject control**: She promised him to come. → She will come.
Object control: She advised him to go. → He should go.
Arbitrary control: She helped to save the child (It is not clear who saved the child in the first place.)
Variable control: She appealed to him to go. She appealed to him to be freed. → He should go./She should be freed. (Variable control verbs allow subject or object control. The interpretation is determined by contextual factors.)

criteria, which is crucial for consistency. Only some criteria are testable against the surface syntax of a single sentence (i.e. they do not refer to pairs, all occurrences, or semantics). The information they use are passivity of the verb, syntactic categories, prepositions and complementizers of typical complements and adjuncts, position in the sentence, and the *wh*-word.

The usability of the criteria was tested by letting four lexicographers mark all and only the complements in a random sample of example sentences. Average pairwise agreement was 91%. The upper bound on the task described in this thesis is probably substantially lower, as it also involves determining the type of complement respectively adjunct. Grishman, Macleod, and Meyers (1994) note that in actual lexicon construction work, the criteria work even better than these figures show, because in many cases where there was disagreement, the lexicographers noted that the example was difficult, ambiguous or involves figurative use and they would probably not base a lexical entry on these examples alone. In any case, they would discuss it with the others, which enhances consistency. When disregarding the problematic cases, pairwise agreement was 93%. This arguments points to a difference between lexicon work and treebanking and parsing: in the latter two cases, also difficult, ambiguous or figurative sentences have to be assigned some parse.

2.6 Automatic subcategorization acquisition

The goal of automatic subcategorization acquisition is to build large subcategorization dictionaries from corpora. These should not only specify which of many possible subcategorization frames can occur with which lemma (*paradigm* information) but also how likely each frame is with each lemma (*frequency* information), and ideally how the frames in a paradigm relate to each other (*diathesis alternations*) and which semantic restrictions (*selection restrictions*) are imposed on the arguments. However, most subcategorization acquisition systems to date aim at paradigm and frequency information, which can be acquired on a syntactic basis, leaving diathesis alternation and selection restrictions for a following acquisition step, which uses more semantic knowledge.

Manually built subcategorization dictionaries rarely include relative frequencies of frames, as compiling this information further complicates an already tedious process. Frame frequency information is essential to guide the parsing process of lexicalized probabilistic parsers but it can also be used to rank the parses returned by a non-probabilistic parser or rerank the parses returned by a probabilistic non-lexicalized parser (Carroll, Minnen, and Briscoe, 1998). In summary, automatic subcategorization acquisition methods generally have two advantages over manual approaches: they are faster and their output contains relative frequencies.

2.6.1 Brent

The first work in automatic subcategorization acquisition was done by Brent (Brent, 1991a; Brent, 1991b; Brent, 1993; Brent, 1994). He uses only raw, i.e. untagged text and regular expressions (encoding *cues*) to recognize verbs and five or six subcategorization frames. The cues have a very high accuracy because they only encode (nearly) unambiguous cases (like "verb+me/him/us/them+punctuation" for the NP frame). For the same reason, they have a very low coverage. This means that acquired frequencies cannot reasonably be converted to probabilities for probabilistic parsing. The algorithm counts how often a verb occurred and how often it occurred with each frame. As even the high-accuracy expressions sometimes match false positives, Brent applies a statistical test to the frequency counts to decide whether there is enough evidence for a frame to be accepted. This binomial hypothesis testing was later also used by Manning (1993), Carroll and Rooth (1998a), Ersan and Charniak (1995), and Briscoe and Carroll (1997). Brent (1991b) notes that the most common source of errors are purpose adjuncts that are mistaken for infinitival complements.

2.6.2 Manning

The system described by Manning (1993) recognizes more frames than Brent's (also including (at most) one PP[29]), uses a PoS tagger (Kupiec, 1992) and a finite-state NP chunker and distinguishes active from passive verbs. The cues are much more broad coverage than Brent's but some "complicated" constructions like participles, coordination or relative clauses are not covered.

2.6.3 Ushioda *et al.*

Ushioda et al. (1993) use similar techniques to Brent and Manning, but focus more on frequency information than on paradigm acquisition. Therefore they do not need a statistical test to filter out false positives: the typically low frequencies of false positives are already good approximations of the true zero frequencies. Noticeable features of the method are that the finite-state NP chunker eliminates temporal NP adjuncts and even some non-NPs like "two months ago" and that the transitive frame can also be recognized in a relative clause. Ushioda et al. (1993) note that the most frequent sources of errors are incorrectly chunked NPs and purpose adjuncts that are mistaken for infinitival complements.

[29]Manning does not make a distinction between prepositions and verb particles, as his example (3b) shows.

2.6.4 Carroll and Rooth

Like Ushioda, Carroll and Rooth (1998a; 1998b) focus more on acquiring frequency than paradigm information. They use a hand-written lexicalized PCFG. The probability parameters of the PCFG are estimated from a large, automatically PoS-tagged corpus with the Expectation Maximization (EM) algorithm (Dempster, Laird, and Rubin, 1977). The grammar has three levels: chunks, phrases, and the sentence level. Chunks exclude complements and trailing adjuncts but might contain other chunks (e.g. a possessive noun chunk inside another noun chunk), phrases consist of a head chunk and its complements, and the sentence level is modeled by a simple trigram model over phrases. Verb chunks are differentiated into finite, present participle, past participle, or infinitival and passive or active. Subcategorization frames are encoded in the rules for phrases. Clausal complements (e.g. *that*-clause, *wh*-clause) cannot be acquired due to the simplistic sentence-level grammar.

2.6.5 Ersan and Charniak

Ersan and Charniak (1995) use the lexicalized PCFG described in Charniak (1995), see Section 2.4.2.5. To derive subcategorization frequencies from the probabilities acquired for parsing, grammar rules for VPs are mapped onto 16 verbal frames by ignoring children that are adverbs or that appear as right siblings of punctuation, coordinating conjunctions or the first PP. Some rules could not be mapped to frames and are ignored. It is unclear how passives are treated.

2.6.6 Briscoe and Carroll

The system of Briscoe and Carroll (1997) acquires a significantly larger number of different frames than any of the previously described systems. The list of possible frames comes from merging the COMLEX and ANLT (Boguraev et al., 1987) frame inventory, plus some additions, and contained 160 frames at the time of the paper (since then, three frames have been added).

This larger number has two reasons. First, some frames were just ignored by earlier work, maybe because they were regarded as marginal or as too difficult. Second, the distinctions are much finer-grained, so that several frames in Briscoe and Carroll's system correspond to just one frame in earlier work. The following list illustrates both cases:

- Additional combinations with PPs: PP+infinitive, PP+*wh*-clause, PP+participial, AdjP+PP, two PPs.

- Further distinction of PPs into PNP *(They talk about him)*, PING *(They talk about leaving)*, PWH *(They talk about whether they should leave)*.

- Additional combinations with particles (recall that Manning did not differentiate prepositions and particles).
- More than two complements.
- Subcategorized ADVP.
- Alternations encoded in frames: particle movement, dative movement.
- Distinguishing equi from raising verbs, and the four types of control used in COMLEX Syntax (cf. Section 2.5.1: subject, object, variable and arbitrary control).
- Distinction between predicative and non-predicative AdjPs, AdvPs, NPs and PPs.
- Marking non-passivizable direct object: "It costs $3."
- Distinction between ADJP and passive participle: "He seems depressed/happy."
- Different possibilities for subject: clausal subject, *it*, extraposition.

So in principle, there are 160 distinct frames. However, not all of these distinctions can be made by the current system. In that case, instead of deciding just on one frame, the system systematically outputs a disjunction (e.g. subject control equi verb or object control equi verb or raising-to-subject or raising-to-object verb). There are 130 possible disjunctions. If instead of allowing disjunctions, the system is forced to make a choice (e.g. by always choosing the most common frame), 115 frames are distinguished, which is still substantially more than in any other system.

Like Ersan and Charniak, Briscoe and Carroll use a general purpose parser within their subcategorization acquisition system. It was already described in Section 2.4.3.2. In contrast to Carroll and Rooth and Ersan and Charniak, the parser makes the C/A distinction but is not lexicalized. After the text is parsed, so-called "patternsets" are extracted from the parse tree, which contain all the necessary information for the next component, the classifier, which assigns patterns to (disjunctions of) frames from the list, or rejects them as unclassifiable. Finally, binomial hypothesis testing is employed to decide whether enough evidence for a frame with a specific verb was seen.

Briscoe and Carroll (1997) show that using the acquired subcategorization frame frequencies in their parser reduces errors (measured in crossing brackets between the predicted and the treebank parse). The parser attaches a probability to every parse tree it returns. Normally the parse with the highest probability is then taken for evaluation. These scores are now refined by multiplying the original probabilities with the relative frequency of a verb's subcategorization frame for each verbal frame that is instantiated in the parse. Carroll, Minnen, and Briscoe (1998) show that the improvement is even more visible when evaluating on argument GRs instead of crossing brackets. GRs are extracted from the highest ranked parse tree as described in Section 2.4.3.2.

Reference	#	frames	for	freq?	Modules	Information
Brent '91–'94	5 (6)	NP, (NP+NP), *that*, NP+*that*, inf, NP+inf	verbs	no	reg. expr., BHT	closed-class words, punctuation, capitalization, typical suffixes, frame frequency
Manning '93	19	—, NP, NP+NP, *that*, NP+*that*, inf, NP+inf, PP(p), NP+PP(p), ing, XCOMP, NP+XCOMP, XCOMP+PP, ...	verbs	no	tagger, reg. expr. incl. NP chunker, BHT	PoS, NP chunks, passivity, punctuation, *that*, *to*, prepositions, subordinating conjunctions, frame frequency
Ushioda '93	6	NP, NP+NP, *that*, NP+*that*, inf, NP+inf	verbs	yes	tagged corpus, NP chunker, reg. expr.	PoS, NP chunks, *that*, *to*, list of temporal NPs
Carroll '98	15 (18)	—, NP, NP+NP, inf, NP+inf, PP, NP+PP, ing, NP+ing, AP, NP+AP, part, NP+part, PP+inf, (PP+PP, part+PP, AP+PP)	word forms of verbs, nouns, adjectives, prepositions	yes	PCFG, BHT	rule, verb chunk type, category of complement, heads, frame frequency
Ersan '95	16	—, NP, NP+NP, *that*, NP+*that*, inf, NP+inf, PP, NP+PP, ing, NP+ing, AP, NP+AP, wh, NP+wh, AP+inf,	word forms or lemmas of verbs, (nouns and adj. with PP frame)	no	PCFG, mapping, BHT	categories, heads, punctuation, coordinating conjunctions, frame frequency
Briscoe '97	160	—, NP, NP+NP, *that*, NP+*that*, inf, NP+inf, PP, NP+PP, ing, NP+ing, AP, NP+AP, part, NP+part, wh, NP+wh, ...	verbs	yes	tokenizer, tagger, lemmatizer, parser, patternset extractor, classifier, evaluator (BHT)	PoS, categories, frame instantiated by rule, passivity, *it*, prepositions, complementizers, *wh*-words, heads only after parsing, frame frequency

Table 2.3: An overview of the subcategorization acquisition work described in this section. The second and third column show how many and which frames the system recognizes (— = intransitive, *that* = tensed *that*-clause with or without *that*, inf = infinitive, PP(p) = PP parameterized for preposition, ing = participial VP, XCOMP = predicative AP or past participle, part = particle, wh = *wh*-clause complement). The fourth column lists for which lemmas frames are extracted, the fifth column notes whether frequencies are evaluated, too. The sixth column shows the modules of a system (BHT = binomial hypothesis testing, reg. expr. = regular expressions). The last column lists the information which the system uses.

2.6.7 Summary: automatic subcategorization acquisition

As Table 2.3 shows, most subcategorization acquisition work defines frames through the syntactic categories of the complements in them. As information, the systems typically

use PoS tags, chunks, punctuation and closed-class words (often syntactic heads). Those that employ a full parser can also use rules and categories, and heads if the grammar is lexicalized. Some also use verb (chunk) properties. Those that employ binomial hypothesis testing to filter out unreliable frames use frame frequencies. Problematic cases include purpose infinitives, participles, coordination, non-local dependencies, and temporal NP adjuncts.

2.7 Conclusion

In this chapter we introduced the basics of phrase structure and dependency structure, and treebanks and parsers that use one or the other. We showed how grammatical relations relate to these theories and how GRs are used to describe phenomena like expletives, control, predicative complements, and non-local dependencies. We reviewed what information can be used to decide about attachment, the C/A distinction, and finer-grained types of GRs. The information sources are diverse, ranging from semantic classes, automatic clusters, selection restrictions, frequency, heaviness, verb subtypes, and surface order over words, PoS, chunks, and categories to punctuation, capitalization and suffixes. The problematic cases are also diverse, including coordination, idiomatic expressions, extraposition, inversion, embedding, purpose infinitives, and the distinction between types of adjuncts.

Chapter 3

Data preparation, setup and evaluation

In the last chapter we reviewed the role and definition of GRs in various fields of computational linguistics and we saw what information can be used to detect them. In this chapter we describe how we define GRs and which information is available to our memory-based learner for finding them. For supervised learning, which we use, this amounts to explaining how training and test material is constructed (Section 3.1). In Section 3.2 we introduce the experimental setup and the evaluation method.

3.1 Data

As explained in Section 1.2.1, an instance for the memory-based learner consists of a fixed number of feature value pairs together with a class label. As no data set yet exists for our GR task in this format, we constructed it from a fully-parsed corpus. In this section, we describe how we convert the information from the parse trees of the Penn Treebank II (cf. Section 2.3.1) into the feature value format. The section consists of three parts. In Section 3.1.1 we describe the original data format of the Penn Treebank. This gives an idea of what information is present and how it is annotated. In Section 3.1.2 we explain how we convert this information from the original phrase structure trees into a dependency-based intermediate format which explicitly represents chunks, heads and grammatical relations between them. Finally, in Section 3.1.3, we describe how we construct machine learning instances from the intermediate format.

3.1.1 The original data: trees in the Penn Treebank II

As was already briefly described in Section 2.3.1, the Penn Treebank features parse trees expressed by labeled brackets. The labels of the leaf nodes are the parts-of-speech. The

```
( (S
    (NP
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (NP
        (NP (CD 61) (NNS years) )
        (ADJP (JJ old) ))
      (, ,) )
    (MD will)
    (VP (VB join)
      (NP (DT the) (NN board) )
      (PP (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director) )))
    (NP (NNP Nov.) (CD 29) ))
  (. .) )
```

Figure 3.1: The first sentence of the WSJ Corpus annotated in Penn Treebank I style. The sentence reads "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.". See Appendix A.1 for the list of PoS tags.

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

Figure 3.2: The first sentence of the WSJ Corpus annotated in Penn Treebank II style. The function tags are: `-SBJ`: subject, `-CLR`: closely related, and `-TMP`: temporal. NPs without function tags under VP or PP are objects.

non-leaf node labels of the first release (Treebank I) just denoted syntactic constituents, like NP, VP or PP.[1] In general only maximal projections are annotated, i.e. there are no intermediate bar levels.[2] The distinction between complements and adjuncts of verbs is indicated by the fact that complements attach under the VP whereas adjuncts attach under

[1] Lists of all part-of-speech tags and syntactic labels are included in Appendix A.1 and A.2.

[2] The exception is NX which is used to annotate structure below the NP level in case of shared and unshared prenominal modifiers in coordinated noun phrases. See Bies et al. (1995, p.140).

```
(NP the right (S (NP-SBJ *) (VP to (VP ...))))
(NP the fact (SBAR that (S (NP-SBJ the Navy) (VP ...))))
(NP (NP a project)  (SBAR (WHNP-1 that)  (S (NP-SBJ *T*-1)  (VP ...))))
(NP (NP conversations) (PP-CLR with (NP people)))
(NP (NP Editorials) (PP-LOC in (NP the Greenville newspaper)))
```

Figure 3.3: In NPs, S and SBAR complements are attached under NP (first two examples), all others are adjoined to NP. Function tags can indicate the function of these adjoined constituents (`-CLR`: closely related, `-LOC`: locative).

S. In the example in Figure 3.1 *"the board"* and *"as a non-executive director"* are complements of *"join"* whereas *"Nov. 29"* is an adjunct. However, this structural annotation of the complement/adjunct distinction did not allow for a satisfactory solution in cases of adjuncts appearing between the verb and one of its complements as in *"He was previously president of the company's Eastern Edison Co. unit."*.[3] We cannot attach *"previously"* under S while at the same time attaching *president* under VP in the context-free parse trees of the Treebank I.

This problem was solved in the second release by attaching *function tags* to the syntactic labels (Marcus et al., 1994). There are 20 function tags, belonging to four groups (Bies et al., 1995), as shown in Appendix A.3. Figure 3.2 shows the sentence annotated in Treebank II style. Complements as well as adjuncts are now attached under the VP and the difference is made by the function tags: NP is a direct object, thus a complement, whereas NP-TMP (temporal) is clearly an adjunct; "-CLR marks constituents that occupy some middle ground between argument and adjunct of the verb phrase" (Bies et al., 1995). In this case, the verb is a prepositional ditransitive (Quirk et al., 1985). In NPs and ADJPs, attachment is still used to indicate the complement/adjunct distinction for S and SBAR but function tags might further differentiate adjoined constituents (cf. Figure 3.3).

In addition to function tags, the Treebank II annotation style also "provides a set of coindexed null elements in what can be thought of as "underlying" position for phenomena such as *wh*-movement, passive, and the subjects of infinitival constructions" (Marcus et al., 1994), see Figure 2.5, p. 20.

3.1.2 From phrase structure trees to the dependency-based intermediate format

In the previous section we described the phrase structure trees in the treebank which are the basis for our training and test material for the task of predicting grammatical relations between heads of verb chunks and heads of other chunks. However the treebank does not explicitly represent chunks, heads and relations between them. Therefore we first convert the trees into an intermediate format which makes these concepts explicit. It is

[3] See wsj_0019.mrg

from this format that we will derive our machine learning instances. Conceptually the intermediate format resembles a dependency structure, but note that multiple heads are allowed. The structure does not only represent syntactic dependencies but also semantic ones. As was explained in Section 2.1.2.1, in order to convert a phrase structure tree into a dependency structure, we have to identify heads of constituents and develop a convention for dependency labels. These two steps will be explained in the next two sections. The transformation from parse trees to dependencies between words is performed by the Perl script `chunklink.pl`.[4] Figure 3.4 shows a sentence in Penn Treebank II style, graphically as a tree and as a dependency structure. Figure 3.5 finally shows it in the intermediate format. The next sections explain the details of the conversion process and of the output format.

3.1.2.1 Identifying heads

The treebank annotation does not always follow $\overline{\text{X}}$ conventions. Therefore identifying heads is more complicated than just following the projection line. The lexical head of a VP for example does not need to have a `V` PoS tag, it might also have `MD` (modal auxiliary) or `TO` (*to*). The head child of a small clause `S` might be some constituent with the function tag `-PRD` (predicative). NPs which do not contain an `N` are a well-known problem (cf. Section 2.3.2). Constituents like fragments (`FRAG`), parentheticals (`PRN`) and unclear cases (`X`) do not have any fixed type of head. And finally the treebank also contains many tagging errors like "`(S (NP-SBJ (PRP$ their) (JJ exclusive) (NN agreement) ) (VP (NNS ends) (PP-TMP (IN in) (NP (NNP March) (CD 1990) ))))`" in which *"ends"* is tagged as a noun although it is the head of a VP. All these cases are dealt with in the *head table* of the `chunklink.pl` program.[5] The table is used to determine which of the children of a non-terminal node are head children. The first respectively second half of the table lists which pre-terminal (PoS) respectively non-terminal (syntactic category) symbol can be the head child of which non-terminal by giving regular expressions that have to match from the start. It is applied as follows:

1. If a parent node X has some pre-terminal children that match the list entry for X, the rightmost of these children is the head child.
2. If no pre-terminal matches, any non-terminal that matches the list entry for X is a head child (cf. Section 3.1.2.5 about multiple heads).
3. If no matching pre-terminal or non-terminal can be found, X has no head child.

According to the first rule, `(NNP Vinken)` in Figure 3.4 is the head child of the first NP, `(JJ old)` that of the ADJP, `(VBD joined)` that of the VP, and so on. According to the second rule, the VP is the head child of the S and the first NP is the head of the NP-SBJ.

[4] `chunklink.pl` can be downloaded from the software section of `http://ilk.kub.nl/`.
[5] The table is reproduced in Appendix A.5.

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (VBD joined)
      (NP (DT the) (NN board) )
      (PP-CLR (IN as)
        (NP (DT a) (JJ nonexecutive) (NN director) ))
      (NP-TMP (NNP Nov.) (CD 29) ))
    (. .) ))
```

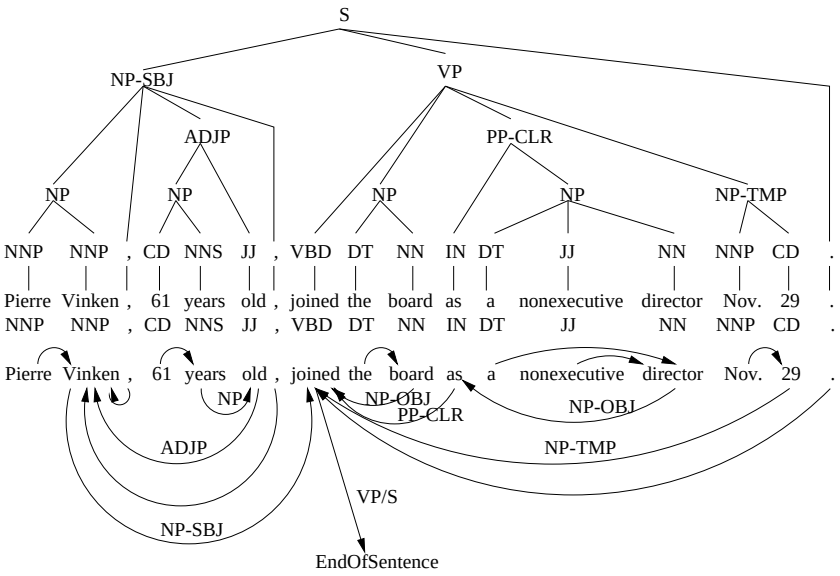

Figure 3.4: A slightly simplified version of the sentence in Figure 3.2 annotated in Penn Treebank II style , the same annotation graphically, and the corresponding dependency structure graphically.

The third rule should never apply in theory, but in practice it does, due to tagging errors in the treebank. Some tagging errors are so frequent that we decided to include the erroneous PoS tag in the head table but less frequent ones will result in headless constituents. This is somewhat arbitrary and we would prefer the treebank to indicate heads explicitly (e.g. along the lines of the NEGRA treebank, cf. Section 2.3.2) which would probably also enforce tagging consistency and prevent different researchers from using different head definitions. The left part of Figure 3.6 shows the result of the first step of head identification on our example: H marks a head child.

In the second step, pointers that indicate the head of the parent node are introduced into the tree. Initially each leaf has a pointer that points to itself (right part of Figure 3.6).

1	2	3	4	5	6	7	8	9
0001	1	0	I-NP	NNP	Pierre	NOFUNC	Vinken	1
0001	1	1	I-NP	NNP	Vinken	NP-SBJ	joined	7
0001	1	2	O	,	,	NOFUNC	Vinken	1
0001	1	3	I-NP	CD	61	NOFUNC	years	4
0001	1	4	I-NP	NNS	years	NP	old	5
0001	1	5	I-ADJP	JJ	old	ADJP	Vinken	1
0001	1	6	O	,	,	NOFUNC	Vinken	1
0001	1	7	I-VP	VBD	joined	VP/S	_	-1
0001	1	8	I-NP	DT	the	NOFUNC	board	9
0001	1	9	I-NP	NN	board	NP-OBJ	joined	7
0001	1	10	I-PP	IN	as	PP-CLR	joined	7
0001	1	11	I-NP	DT	a	NOFUNC	director	13
0001	1	12	I-NP	JJ	nonexecutive	NOFUNC	director	13
0001	1	13	I-NP	NN	director	NP-OBJ	as	10
0001	1	14	B-NP	NNP	Nov.	NOFUNC	29	15
0001	1	15	I-NP	CD	29	NP-TMP	joined	7
0001	1	16	O	.	.	NOFUNC	joined	7

Figure 3.5: The dependency structure in the intermediate format. The columns contain: the file number, the sentence number, the (source) word number, its IOB tag, its PoS, the (source) word, the dependency label, the target word and the target word number.

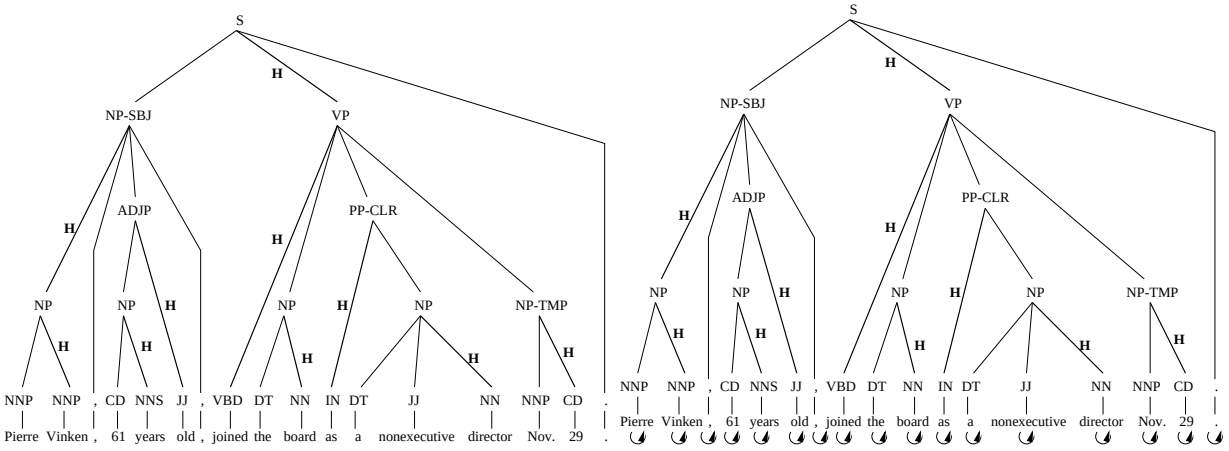

Figure 3.6: The process of percolating head pointers up (part one). First, the head child of each constituent is marked by H. Then, each word gets a pointer to itself.

The algorithm then proceeds bottom-up, at each non-terminal determining the head child according to the annotation added in the previous step, changing the pointers from each child and placing a pointer from the parent to whatever the head child's pointer is pointing at (see Figure 3.7). Conceptually the process is identical to lexicalization of the tree as performed e.g. by Collins (1996).

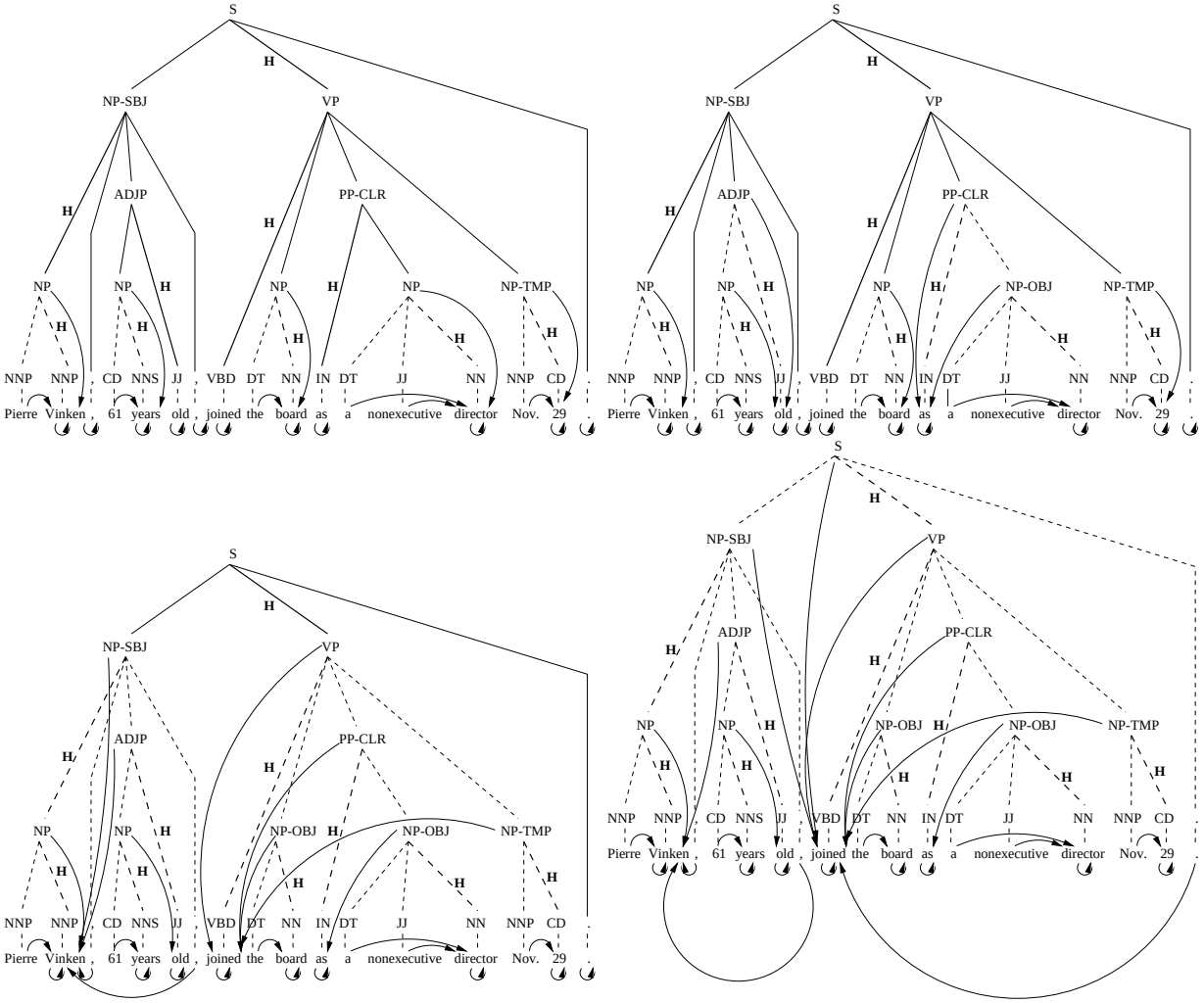

Figure 3.7: The process of percolating head pointers up (part two). At each percolation step, new pointers are introduced that point from a constituent to whatever word its head child is pointing at. Then all the constituent's immediate children are also made to point at this word. The introduction of the special tag `-OBJ` is described on page 59.

3.1.2.2 Percolating functions down: dependencies between words

The pointers introduced in the previous conversion step indicate dependency relations between syntactic constituents and headwords. Thus there is e.g. an NP-SBJ relation between the constituent *Pierre Vinken, 61 years old,* and the verb *joined.* As we are not interested in dependencies between constituents and words, but between pairs of words, the algorithm goes through the tree a last time, top-down, and pushes the labeled pointer of the parent down to the head child. The process is represented graphically in Figure 3.8. If the syntactic part of the pointer label and of the non-terminal head child are identical, the pointer label stays the same (cf. NP-SBJ with head child NP). If however the syntactic

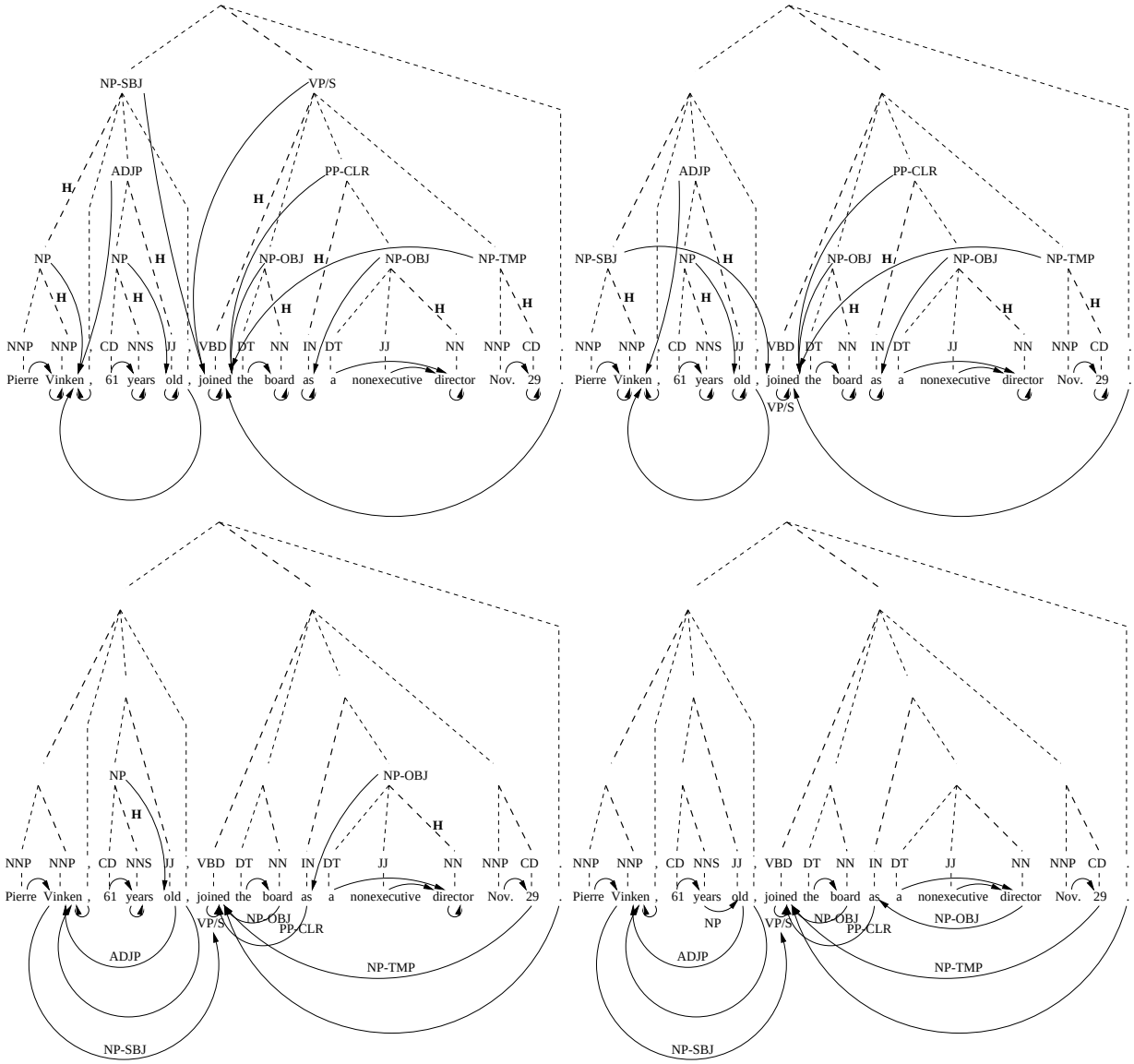

Figure 3.8: The process of percolating functions down. At each step the pointer from a constituent is "pushed down" to its head child. If parent and head child have a different syntactic category, the categories are concatenated.

part is different, the head child label is prefixed to the pointer label (e.g. S with head child VP results in a VP/S label).

At the end of the process just described, the main verb of a sentence carries a pointer to itself. We therefore added an extra step that redirects this pointer to a special token (depicted as `EndOfSentence` in Figure 3.4).[6] The effect can be seen in the third part of Figure 3.4.

[6] A similar end-of-sentence token was used by Eisner (1996b).

```
( (S
    (ADVP-TMP (RB Previously) )
    (, ,)
    (NP-SBJ (NNS regulators) )
    (VP (VBD insisted)
      (SBAR (IN that)
        (S
          (NP-SBJ (NNS franchisers) )
          (VP
            (VP (VB pre-register)
              (NP (JJ such) (NNS changes) )
              (PP-CLR (IN with)
                (NP (DT the) (NN state) )))
            (: --)
            (NP
              (NP (DT a) (JJ costly) (NN process) )
              (VP (VBG taking)
                (NP-TMP
                  (ADVP (IN at) (JJS least) )
                  (CD six) (NNS weeks) )))))))
    (. .) ))
```

Figure 3.9: The level of attachment of NPs without function tags determines its grammatical relation to the head verb: the NP "such changes" is a sibling of the verb and thus an object. The NP "a costly process ..." is a sibling of VP and thus an adjunct.

When "pushing down" the function pointers, we lose the information about the level at which they were originally attached. In most cases this information is not relevant as the function is defined by the combination of syntactic category and function tag. However, in the case of NP without function tag for example, the level of attachment makes the difference between a complement (object) and an adjunct (cf. Figure 3.9).

In order to preserve this distinction, we add the new function tag `-OBJ` to the following constituents if they occur without function tags and as siblings of lexical heads: `NP, VP, ADJP, S, SBAR, SINV, SQ, SBARQ`. This is done during the upward percolation of head pointers.

The final dependency structure is output by the `chunklink.pl` program in what we call the intermediate format (cf. Figure 3.5). In this format each non-empty line corresponds to one word in the input.[7] The information about the words is organized into columns. The first column contains an identifier for the file the sentence came from (e.g. 0001 for file `wsj_0001.mrg`). The second contains the number of the sentence (unique per input file), the third the word number (by default unique per output file; one output file may correspond to many input files). The fourth column contains the chunk tag which will be explained in more detail in the next section. The fifth and sixth column show the PoS

[7]Except lines starting with # which contain documentation about parameter settings

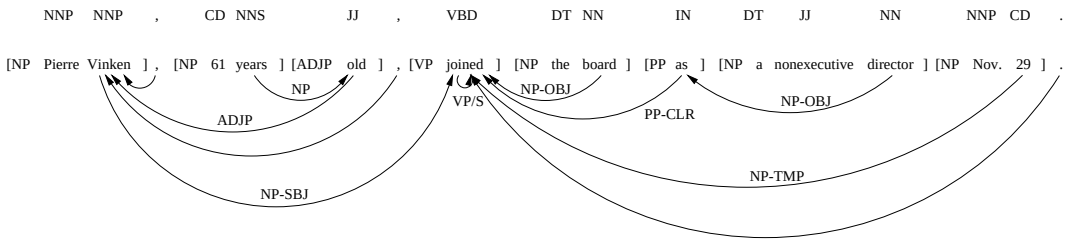

Figure 3.10: The chunks

and word, respectively. The seventh column contains the dependency label. "NOFUNC" marks unlabeled dependencies. The eighth and ninth column represent the target of the dependency arrow originating at the word, i.e. the word's head. The eighth column contains the target word and is intended for humans. The other column contains the target's unique word number and should be used by programs. If the word is the head of the sentence, the two columns contain "_" and -1, respectively.

3.1.2.3 Chunks

Our task is to find grammatical relations between head words of chunks. Like heads, chunks are not explicitly indicated in the treebank. We define a chunk to consist of a head (i.e. any word that has a *labeled* pointer) plus the continuous sequence of all words in front of it that have an *unlabeled* pointer to this head. This definition guarantees that the last word in a chunk is always a head and that we can thus disregard all non-last words of a chunk when searching for grammatical relations between heads. The type of a chunk is equal to the first syntactic part of the head's pointer label.[8] The above definition results in the chunks shown in Figure 3.10. Note that not all tokens are parts of chunks. Typical tokens outside any chunk are punctuation symbols, coordinating conjunctions, negation and auxiliaries in questions. In some cases, the above definition results in chunks that are slightly different from the chunks in e.g. Ramshaw and Marcus (1995) (cf. Section 2.4.1.1). The chunks are represented in the fourth column of the intermediate format by means of IOB tags. `B` here means "Between". `I` and `B` tags have a second part which indicates the type of chunk the word is in (e.g. `I-VP, B-NP`).

3.1.2.4 Pruning

The previous sections explained our general method for converting parse trees to chunks and dependencies. In some specific cases however, the method would yield unwanted chunks. According to the above rules, the partial tree `(NP (QP 10 billion) yen)` would be converted to the following chunks: $[_{QP}$ `10 billion ]` $[_{NP}$ `yen ]`. Previous work on

[8] An exception are labels starting with `WHNP, WHPP, WHADJP`, or `WHADVP`. The corresponding chunks are of types `NP, PP, ADJP` and `ADVP`.

```
(VP (VBP make)                    (VP (VBP make)
  (S                                (NP (PRP them)
    (NP-SBJ (PRP them)              (ADJP-PRD (JJ fearful) )))
    (ADJP-PRD (JJ fearful) ))))
```

Figure 3.11: Pruning of predicative small clauses results in the NP and the predicative ADJP becoming objects of the verb.

NP chunks however (cf. Section 2.4.1.1) always defined this whole phrase as one NP chunk (also called baseNP). To achieve a similar result, we introduced a pruning step before the actual conversion in which constituents like QPs are deleted and its children attached to its parent. From the resulting simpler tree `(NP 10 billion yen)` we get the NP chunk we wanted. The constituents that have to be pruned this way are listed in the beginning of the `chunklink.pl` script.[9]

A similar but slightly more complicated case arises if the NP contains an ADJP, e.g. `(NP (ADJP publicly listed) bonds)`. Again, previous NP chunk definitions regard the whole phrase as one chunk. Again the solution is to prune the ADJP. However there are two conditions: we only prune ADJPs in NPs (and not e.g. as children of VPs) and only if they precede the NP's head. Without the latter condition, the ADJP in Figure 3.4 would incorrectly be pruned. The constituents to be pruned are listed in `chunklink.pl`.[10]

The third group of pruning concerns VPs and Ss. The example in Figure 3.4 we used for illustration contains only one verb. The original sentence however contains two verbs (a modal and a main verb) and thus two VPs (cf. Figure 3.2). This would result in two VP chunks. Intuitively however the two verbs should go together into one chunk. The modal is the syntactic head of this chunk but the main verb is the semantic head. This is also the case in the original chunking work of Abney (1991). Also from the engineering point of view it is more efficient to have only one VP chunk $[_{VP}$ `will join ]` instead of two. To achieve this, we prune VPs as children of VPs if the corresponding flag is set in the beginning of `chunklink.pl`[11] and if some potential head or adverbial phrase and nothing else precedes the child VP. The latter conditions prevent pruning in cases like `(VP (VP ...) and (VP ...))` or `(VP permit (NP portfolio managers) (VP to (VP retain ...)))`.

Parallel to ADJPs in NPs and ADVPs in ADJPs we also prune ADVPs in VPs. Due to possible interactions with the VP pruning, the ADVP-in-VP pruning has its own flag and procedure in `chunklink.pl`.

There are two cases in which we prune S: predicatives and control verbs. If an S contains a subject and a child with the function tag `-PRD` (predicative), the S is pruned and the

[9] `$prune_always`
[10] `$prune_if_infrontof_head`
[11] `$prune_vp_in_vp_flag`

```
(S                                      (S
  (NP-SBJ-1 (PRP it) )                    (NP-SBJ-1 (PRP it) )
  (VP (VBZ is)                            (VP (VBZ is) (VBN expected) (TO to) (VB take)
    (VP (VBN expected)                      (NP (DT another) (JJ sharp) (NN dive) )))
      (S
        (NP-SBJ (-NONE- *-1) )
        (VP (TO to)
          (VP (VB take)
            (NP (DT another) (JJ sharp) (NN dive) )))))))

(S                                       (S
  (NP-SBJ-2 (PRP we)                      (NP-SBJ-2 (PRP we)
  (VP (VBD evaluated)                     (VP (VBD evaluated) (VBG raising)
    (S                                      (NP (PRP$ our) (NN bid) ))))
      (NP-SBJ (-NONE- *-2) )
      (VP (VBG raising)
        (NP (PRP$ our) (NN bid) ))))))
```

Figure 3.12: Two examples of pruning to join several VPs into one. The left side shows the original trees, the right side the result of pruning. "*" is the trace used for annotating control.

subject tag deleted,[12] see Figure 3.11. This makes both children directly dependent on the verb which is necessary because our shallow parser predicts only relations to verbs. Figure 3.12 shows two control examples before and after pruning. In a VP, there is an S without function tag[13] which contains an empty subject and another VP (infinitive or gerund). We prune the NP-SBJ and the S node.[14] The inner VP then becomes a direct child of the outer VP and will be pruned due to the previously discussed VP-in-VP rule. The result is just one verb chunk, which is more efficient for finding GRs. To restore the original information, we need to make the distinction between auxiliaries and main verbs and generate a predicate for every main verb in a chunk.[15] See Section 6.2.2 for more details.

As a concluding remark to this section about pruning in VPs we note that it is still possible to have two adjacent VP chunks, e.g. `(S (NP-SBJ The object) (VP is (S-PRD (NP-SBJ *) (VP to (VP (VB capture) (NP (NP profits) ...))))))` results in `[`$_{NP}$ `The object ]` `[`$_{VP}$ `is ]` `[`$_{VP}$ `to capture ]` `[`$_{NP}$ `profits ]`. `S-PRD` is not pruned because it carries a tag. The relation finder then has to determine the type of relation between *is* and *capture*.

[12] `$prune_s_in_vp_predicative_flag`

[13] This condition prevents pruning in cases like `S-PRP` (purpose and reason), `S-ADV` (adverbial) etc.

[14] `$prune_s_in_vp_empty_subject_flag`

[15] The distinction between subject and arbitrary control (coindexed and non-coindexed trace) is lost however. See also Section 3.1.2.6 on how to preserve the information of coindexed empty constituents.

3.1.2.5 Coordination

In the previous sections we explained the general method for the conversion of Penn Treebank trees to chunks and dependencies. In the current and the next section, we have a more detailed look at two of the most difficult problems in parsing: coordination and long-distance dependencies, and their interaction with the conversion. These phenomena deserve special attention as they touch upon the notion of head which is central to the conversion algorithm. Coordination introduces multiple heads, whereas long-distance dependencies are annotated in the treebank through traces and therefore have empty heads.

We already saw in Chapter 2 that coordinated structures need a special treatment in most theoretical and practical NLP work. Figure 3.13 and the first part of Figure 3.15 show some examples of constituent coordination from the treebank. Most syntactic categories can be coordinated and we see that the coordinating conjunction can consist of one or more, even discontinuous words (*either ... or*), that two or more constituents can be coordinated and that even two different syntactic categories can be coordinated (together forming a UCP, i.e. "unlike coordinated phrase"). In all cases the constituents are coordinated in the sense that neither of them is more important than the other, neither syntactically nor semantically. It is for this reason that we do not want to attribute the head child property to only one of them. Instead, all coordinated constituents are marked as head children. This in turn results in the coordinated constituent having multiple heads. When the pointers are percolated down the tree, they are copied onto each head child. This results in several chunks having the same relation to one verb (e.g. *unit* and *critics* to *complain*) or conversely one chunk having the same relation to several verbs (e.g. *trading* to *causes* and *increased*) (see Figure 3.14).

In addition to constituent coordination, the treebank also contains cases of coordinated words inside a constituent (see "Oct. 13 and 16" in Figure 3.13 and the last example in Figure 3.15). According to our head finding algorithm, only the rightmost of these coordinated words will be marked as head. This in turn yields chunks like $[_{NP}$ `N and N ]`. This result is in line with previous work on baseNPs. We have to deal with intra-chunk coordination when we compare MBSP to the GR extraction system of Carroll, Minnen, and Briscoe (1998), see Section 6.2.2.

3.1.2.6 Non-local dependencies

The treebank annotation provides different kinds of traces for different syntactic phenomena, e.g. : "*T*" for *wh*-movement and fronting and "*" for passives and control (see Bies et al. (1995, p.59ff.) for a full list). A constituent that only contains a trace is an empty constituent. Like any other constituent, it can have a relation indicated by function tags attached to the syntactic label. Grammatical relations hold between heads. Empty constituents also have empty heads. This is why they constitute a special case for our approach.

```
( (S
    (NP-SBJ
      (NP (DT The) (NNP Kemper) (NNP Corp.) (NN unit) )
      (CC and)
      (NP (JJ other) (NNS critics) ))
    (VP (VBP complain)
      (SBAR (DT that)
        (S
          (NP-SBJ (NN program) (NN trading) )
          (VP
            (VP (VBZ causes)
              (NP
                (NP
                  (NP (JJ wild) (NNS swings) )
                  (PP-LOC (IN in)
                    (NP (NN stock) (NNS prices) )))
                (, ,)
                (PP (JJ such) (IN as)
                  (PP-TMP
                    (PP (IN on)
                      (NP (NNP Tuesday) ))
                    (CC and)
                    (PP (IN on)
                      (NP (NNP Oct.) (CD 13)
                        (CC and)
                        (CD 16) )))))
              (, ,) )
            (CC and)
            (VP (VBZ has)
              (VP (VBN increased)
                (NP
                  (NP (NNS chances) )
                  (PP (IN for)
                    (NP (NN market) (NNS crashes) ))))))))
    (. .) ))
```

Figure 3.13: A treebank sentence featuring four coordinated structures.

Some traces are not coindexed with another constituent in the sentence. Examples include reduced relative clauses (Bies et al., 1995, p.21), arbitrary control, some raising constructions, and some unannotated control cases, as shown in Figure 3.16. Other traces are coreferent with another constituent (the filler), see Figure 3.17. Thus the relation indicated on the label of the empty constituent holds between the head of the filler and whatever is the head of the parent of the empty constituent. A head can have several relations, at most one local relation and one or more non-local relations.

During the conversion process from trees to dependencies, traces are treated like ordinary words. The relation that holds for the trace is copied onto the filler and indicated by

```
...
 0000  1  3 I-NP    NN    unit           NP-SBJ         complain            7
 0000  1  4 O       CC    and            NOFUNC         unit/critics        3/6
 0000  1  5 I-NP    JJ    other          NOFUNC         critics             6
 0000  1  6 I-NP    NNS   critics        NP-SBJ         complain            7
 0000  1  7 I-VP    VBP   complain       VP/S           _                  -1
 0000  1  8 I-SBAR  DT    that           SBAR-OBJ       complain            7
 0000  1  9 I-NP    NN    program        NOFUNC         trading            10
 0000  1 10 I-NP    NN    trading        NP-SBJ         causes/increased   11/31
 0000  1 11 I-VP    VBZ   causes         VP/S-OBJ       that                8
...
 0000  1 29 O       CC    and            NOFUNC         causes/increased   11/31
 0000  1 30 I-VP    VBZ   has            NOFUNC         increased          31
 0000  1 31 I-VP    VBN   increased      VP/S-OBJ       that                8
...
```

Figure 3.14: Parts of the sentence from Figure 3.13 in the intermediate format.

```
(NP-SBJ (CC either)
  (NP (DT the) (NNP CFC) (NN gas) )
  (CC or)
  (NP (DT the) (NN insulation) ))

(UCP
   (ADJP
    (NP (CD 55) (NNS years) )
    (JJ old) )
  (CC and)
  (NP
    (NP (JJ former) (NN chairman) )
    (PP (IN of)
      (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) ))))

(PP-DIR (TO to)
  (CC and)
  (IN from)
  (NP (DT the) (NN airport) ))
```

```
(NP
  (NP (NNP Bloomingdale) (POS 's) )
  (, ,)
  (NP (NNP Bon) (NNP Marche) )
  (, ,)
  (CC and)
  (NP (NNP Jordan) (NNP Marsh) ))
```

Figure 3.15: Examples of constituent (top) and non-constituent coordination (bottom) from the treebank.

additional columns in the intermediate format. A special case arises if an empty constituent that is coindexed and also contains a coindexed trace is pruned like `(NP-SBJ-3 (-NONE-*-2))`. The two indices are then unified before pruning. The right upper part of Figure 3.17 shows the tree after pruning and index unification. The lower part shows the resulting intermediate format.

In contrast to our parser, most full parsers (cf. Section 2.4.2) ignore traces by just deleting them in the training material. Consequently they are never predicted in test material.

```
(NP
  (NP (NN home) (NNS loans) )
  (VP (VBN insured)
    (NP (-NONE- *) )
    (PP (IN by)
      (NP-LGS (DT the) (NNP Federal) (NNP Housing) (NNP Administration) ))))

(S
  (NP-SBJ
    (NP (DT the) (JJ mere) (NN existence) )
    (PP (IN of)
      (NP (DT a) (JJ market-stabilizing) (NN agency) )))
  (VP (VBZ helps)
    (S
      (NP-SBJ (-NONE- *) )
      (VP (TO to)
        (VP (VB avoid)
          (NP (NN panic) ))))))

(S
  (NP-SBJ (EX there) )
  (VP (VBZ appears)
    (S
      (NP-SBJ (-NONE- *) )
      (VP (TO to)
        (VP (VB be)
          (NP-PRD
            (NP (DT a) (NN market) )))))))

(S
  (NP-SBJ (PRP he) )
  (VP (VBZ has)
    (NP (DT no) (NN inclination)
      (S
        (NP-SBJ (-NONE- *) )
        (VP (TO to)
          (VP (VB eliminate)
            (NP (NN program) (NN trading) )))))))
```

Figure 3.16: Examples in which the trace (-NONE- *) is not coreferenced with anything. In order: a reduced relative clause, arbitrary control, a raising construction, and an unannotated control case.

The only exception is Collins's model 3 which predicts traces from NP extraction in SBAR relative clauses.

```
( (SBARQ (WHNP-1 (WP Who))                          ( (SBARQ (WHNP-1 (WP Who))
       (SQ (VBD was)                                       (SQ (VBD was)
           (NP-SBJ-2 (-NONE- *T*-1))                           (NP-SBJ-2 (-NONE- *T*-1))
           (VP (VBN believed)                                  (VP (VBN believed)
               (S (NP-SBJ-3 (-NONE- *-2))
                  (VP (TO to)                                      (TO to)
                      (VP (VB have)                                (VB have)
                          (VP (VBN been)                           (VBN been)
                              (VP (VBN shot)                       (VBN shot)
                                  (NP (-NONE- *-3)))))))))         (NP (-NONE- *-2))))
       (. ?) ))                                            (. ?) ))

  1   2  3   4     5    6            7         8    9     10     11   12     13     14  15

0000  1  0  I-NP  WP   Who       WHNP/SBARQ  _    -1   NP-SBJ  *T*   6   NP-OBJ   *   6
0000  1  1  O     VBD  was       NOFUNC      shot  6
0000  1  2  I-VP  VBN  believed  NOFUNC      shot  6
0000  1  3  I-VP  TO   to        NOFUNC      shot  6
0000  1  4  I-VP  VB   have      NOFUNC      shot  6
0000  1  5  I-VP  VBN  been      NOFUNC      shot  6
0000  1  6  I-VP  VBN  shot      VP/SQ       Who   0
0000  1  7  O     .    ?         NOFUNC      Who   0
```

Figure 3.17: Groups of three extra columns are used in the intermediate format to indicate non-local dependencies, which the treebank annotates by traces. The first of each group indicates the non-local dependency label, the second the kind of trace and the third the target number. Note that this is not an actual treebank sentence, the number 0000 is a dummy. `-1` represents the EndOfSentence token.

3.1.2.7 Possessives

Like PPs and SBARs, possessive constructions (like "Peter's house") are an example of a mismatch between the syntactic and the semantic head. The possessive *'s* is the syntactic head as it enables the NP to function as a prenominal modifier. The rest of the NP however determines the content (i.e. whose house it is) and is therefore the semantic head. Possessive NP constructions are annotated in the treebank like `(NP (NP (NNP Peter) (POS 's) ) (NN house) )`. When applying our standard rules for conversion from trees to chunks we would get $[_{NP}$ `Peter 's ]` $[_{NP}$ `house ]` if we allow the part-of-speech POS to be the head of an NP, or $[_{NP}$ `Peter ]` `'s` $[_{NP}$ `house ]` if we do not. The former solution would yield a semantically empty head `'s` whereas the latter yields one extra element, i.e. less reduction. The approach taken in Ramshaw and Marcus (1995) (cf. Section 2.4.1.1) and followed here is to define the chunks as $[_{NP}$ `Peter ]` $[_{NP}$ `'s house ]`. This is done by a special condition in the `chunks` procedure in `chunklink.pl`.

3.1.2.8 Comparison of head tables

The idea of the head table comes from Magerman (1995). Collins used a similar table. The application of their tables is somewhat different from ours however. In our algorithm, the head is either the rightmost pre-terminal child that matches the (regular expression) PoS list in the table, or all non-terminal children that match the (regular expression) constituent list. Thus there is a preference for lexical over non-lexical head children but no preference within these groups. In their approach, by contrast, the list is ordered by preference and also has an associated direction (starting left or right). The head finding algorithm first tries to find a child of the kind indicated by the first element of the list in the direction indicated. It stops at the first one it finds. If no child of this kind can be found, the algorithm next looks for a child of the kind of the second element of the list, and so on, down to the last. If even the last kind of child cannot be found, the algorithm takes the left/rightmost child (of any kind) to be the head. There are special rules for finding the head of NPs and for coordinated constructions. It is hard to understand the practical consequences of the different approaches by their theoretical descriptions, so we will give some examples in which the head tables and algorithms yield different results. The biggest difference occurs with coordinated structures. Neither Collins nor Magerman allows for multiple heads, so their algorithms will take only the leftmost respectively rightmost coordinate as head. Our algorithm will always take all coordinates as heads. It is for this reason that we did not implement any preferences among non-terminal children.

Another difference occurs with tagging errors. Collins's algorithm always returns a head as the model needs heads to condition the probabilities on, whereas ours will assign no head if none of the elements in the head table matches. This feature can be used to point out tagging errors by using a very strict head definition and checking all cases where the algorithm could not find any head child.

Apart from the possibility to have multiple heads in coordinated structures, we do not think that the differences between Collins's and our head tables and algorithms are important. The precise definition of the head in case of erroneous tags or rare constructions presumably does not matter, as long as it is done consistently.

3.1.3 From the intermediate format to instances for machine learning

The intermediate format explicitly represents all the concepts that are essential for our task definition: chunks, heads and grammatical relations. However it does not constitute a format that is suited for a machine learning algorithm. Therefore we use another program[16] that takes the intermediate format as input and outputs instances. Classification is done per instance. When applying machine learning to a problem, the first decision always has to be "To what unit in our data does an instance correspond?" Sometimes the answer is

[16] `chunklink_inst.pl` can be downloaded from `http://ilk.kub.nl/`.

obvious, e.g. for the task of assigning one part-of-speech tag to each word, the obvious unit is the word. For our task, we define an instance to correspond to a pair of chunks: a verb chunk and another chunk.[17]

The "other chunk" will henceforth be called the *focus chunk*. As punctuation is outside any chunk, no instance is created for it. Likewise, negation and some auxiliary verbs in questions do not belong to any verb chunk and therefore do not give rise to instances. One restricting definition that might seem trivial is that the verb chunk and the focus chunk have to be in the same sentence. This is not a principled restriction of our method, in fact one could imagine using a similar kind of instances and a memory-based learner to decide about coreference, e.g. are "Pierre Vinken" and "He" coreferent in the sentence pair "This is Pierre Vinken. He is 61 years old." For grammatical relations however, the common definition is that they do not cross sentence boundaries.

Having decided that an instance represents a pair of verb and focus chunks, we now need classes which encode the task. In our case, the functions from the seventh column of the intermediate format are a natural choice. We will first ignore the non-local dependencies (from the tenth column onwards) and come back to them in Section 6.1.1. The last and most complex decision concerns the features. In principle all the information in columns 3, 4 and 5 of the intermediate format for one sentence could be represented in features of the instance (i.e. this information constitutes the history). However this intuitively seems to be too much information. In addition sentences vary in length so we would need very many features to accomodate the longest sentence. For short sentences most of these features would then be empty as the learner requires all instances to have the same number of features. This is inefficient.

Choosing among the possible features (and their representation) already touches on our central research question of what information is important for relation finding. Therefore the selection introduced here is only preliminary. We will explore other features in Chapter 5. The preliminary feature selection and representation is largely based on Buchholz, Veenstra, and Daelemans (1999) (see Section 2.4.4.1). It contains the following features:

- The *distance* from the verb chunk to the focus chunk, counted in elements (i.e. chunks or "outside" words or punctuation tokens). A negative distance means that we have to "go backwards" from the verb chunk to find the focus chunk, i.e. the focus chunk is to the left of the verb chunk. A distance of $(-)1$ means that the focus chunk is directly adjacent to the verb chunk. Collins (1996) already used a distance measure that combines, among other things, a binary direction feature and a binary adjacency

[17] If we did not restrict ourselves to the task of finding grammatical relations of verbs, we would have to make an instance for each pair of two (different) chunks. The alternative would be to create one instance for each word and to have the algorithm predict the relation type *and* the target, either in one single step or in two subsequent ones. The latter option is chosen in Aït-Mokhtar and Chanod (1997b). Van den Bosch and Buchholz (2002) describe a system that performs only *function tagging*, which is the first step in the two step approach. The two alternatives (given source and potential target, predict relation, or given source, predict relation and target) are similar to Models A and B of Eisner (1996b), respectively.

feature because "conditioning on the exact distance between two words ... leads to severe sparse data problems". Eisner (1996a) also has a binary direction feature, and a distance feature with the four possible values 1, 2, 3–6, and 7–∞.

- The number of other verb chunks between the verb chunk of interest and the focus chunk (*intervening verb chunks*). The distance measure of Collins (1996) also contains a binary feature indicating whether there are any verbs between the two words.
- The headword of the *verb* chunk. This feature should enable the algorithm to learn e.g. subcategorization preferences.
- The focus chunk itself. This chunk is described using four features:
  - The head *word*. If the element is a PNP chunk, this means the nominal headword. Most work in parsing and grammar theories agrees that the head carries important information.
  - The *part-of-speech* of the headword. Magerman, Collins, Eisner and Charniak (1999) all use this information.
  - The *chunk type* of the element. In our chunk-based shallow parser, this information corresponds to the syntactic category information used by full parsers.
  - The *prepositional* headword in case the element is a PNP chunk. Otherwise this feature is empty (i.e. the value is some special value, here: "–"). As we saw in the PP attachment literature (Section 2.4.1.2), both the preposition (the syntactic head) and the head noun (the semantic head) are useful for attachment decisions.
- The element directly preceding the focus chunk (henceforth called the *context–1*). It is common in parsing to use some information from the local context, cf. Ratnaparkhi (1998), Collins (1999). As a starting point we represent the context by the same four features as the focus. If the element is not a chunk (but e.g. punctuation) the chunk type feature is empty (i.e. "–").
- The element immediately following the focus chunk (*context+1*), again with the same four features.

Figure 3.18 shows the history (in the sense of history-based grammar, cf. Section 1.2) of our example sentence and illustrates the above features for it.

In addition to these features used for classification, we also use two "administrative" features. Each word in our data has a unique word number. Every instance then has one feature for the word number of the head of the focus chunk and one for the word number of the head of the verb chunk. During training and classification these features are ignored by the learning algorithm. However they are handy for evaluation and error analysis (e.g. to retrieve the original sentence).

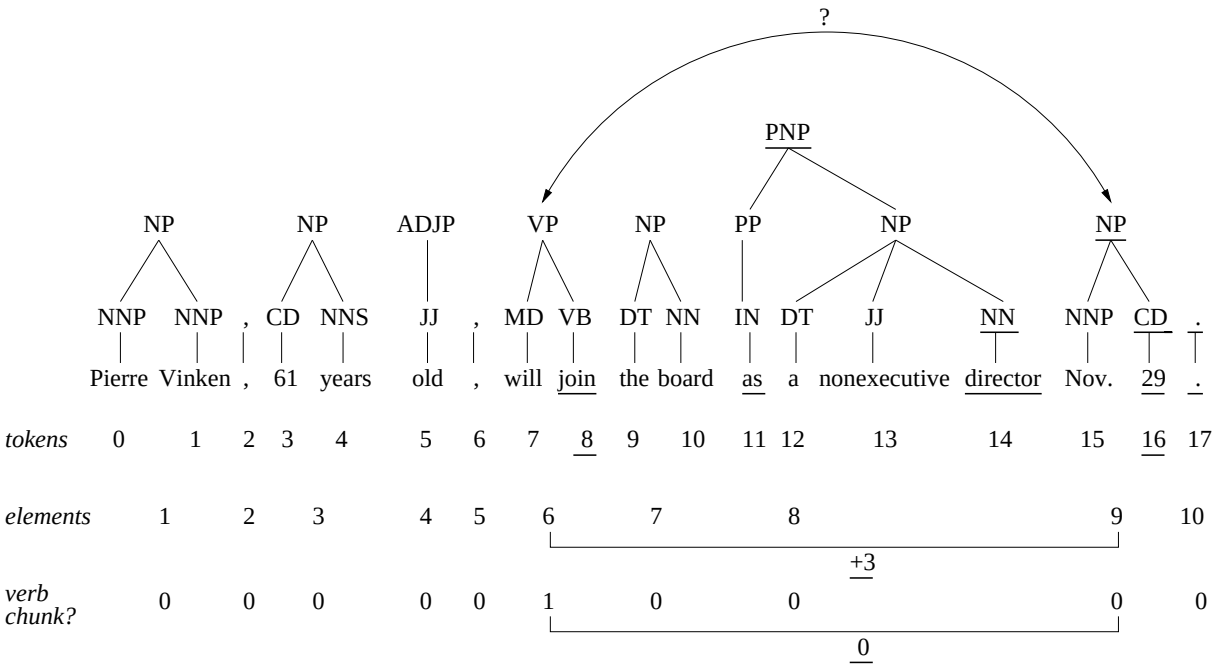

Figure 3.18: Our example sentence with parts-of-speech and chunks annotated, which is the conceptual input format to the relation finder (the history). The underlined parts are used as feature values of the instance that corresponds to the decision about the relation (marked by the question mark) between the verb and the "Nov. 29" chunk (cf. column 6 in Table 3.1).

Table 3.1 shows all the instances derived from the sentence in Figure 3.18. There is one exception to the rule that the classes correspond to the seventh column of the intermediate format. This is for the *by*-phrase of passives which in the treebank is annotated as:
`(VP (VBN seen) (NP (-NONE- *-1)) (PP (IN by) (NP-LGS (NNP Peter)))).`
The `-LGS` function tag marks logical subjects. In the intermediate format, this corresponds to:

```
0000  1  9 I-VP    VBN   seen      VP/S       _          -1
0000  1 10 I-PP    IN    by        PP         seen       9
0000  1 11 I-NP    NNP   Peter     NP-LGS     by         10
```

We internally change this to:

```
0000  1  9 I-VP    VBN   seen      VP/S       _          -1
0000  1 10 I-PP    IN    by        PP-LGS     seen       9
0000  1 11 I-NP    NNP   Peter     NP         by         10
```

before creating instances as it is the preposition that has a grammatical relation to the verb and therefore needs to carry the `-LGS` information.

features			6 instances					
	focus #	1	1	4	5	10	11	16
	verb #	2	8	8	8	8	8	8
	dist.	3	−5	−3	−2	+1	+2	+3
	VCs	4	0	0	0	0	0	0
	verb	5	join	join	join	join	join	join
context −1 (left)	prep.	6	–	–	–	–	–	as
	word	7	–	,	years	join	board	director
	PoS	8	–	,	NNS	VB	NN	NN
	chunk	9	–	–	NP	VP	NP	PNP
focus chunk	prep.	10	–	–	–	–	as	–
	word	11	Vinken	years	old	board	director	29
	PoS	12	NNP	NNS	JJ	NN	NN	CD
	chunk	13	NP	NP	ADJP	NP	PNP	NP
context +1 (right)	prep.	14	–	–	–	as	–	–
	word	15	,	old	,	director	29	.
	PoS	16	,	JJ	,	NN	CD	.
	chunk	17	–	ADJP	–	PNP	NP	–
	class		NP-SBJ	–	–	NP-OBJ	PP-CLR	NP-TMP

Table 3.1: The instances for the first sentence of the WSJ Corpus.

3.2 Experimental setup

The data described in the previous section is used in the machine learning experiments that will be reported in the next chapters. This section explains the general experimental setup.

3.2.1 Size of data set

Unless otherwise indicated, experiments are performed on sections 10 to 19 of the WSJ Corpus of the Penn Treebank II. These sections contain 515,390 tokens (words and punctuation) in 21,747 sentences (average of 23.7 tokens per sentence). This results in 555,539 instances, with 320 different classes (for local relations only). The 15 non-administrative features have between 2 and 20,809 feature values.

When trying to optimize the settings of the learning algorithm and the feature representation, the search space is huge. The various options of the algorithm alone combine to a nearly infinite number of settings. In addition, a nearly unlimited number of features can be extracted from the sentences and the linguistic structure built in previous stages (the history). A strategy followed in Veenstra et al. (2000) is to run an automatic large-scale search through a lot of algorithmic and representational parameters. This is possible if the

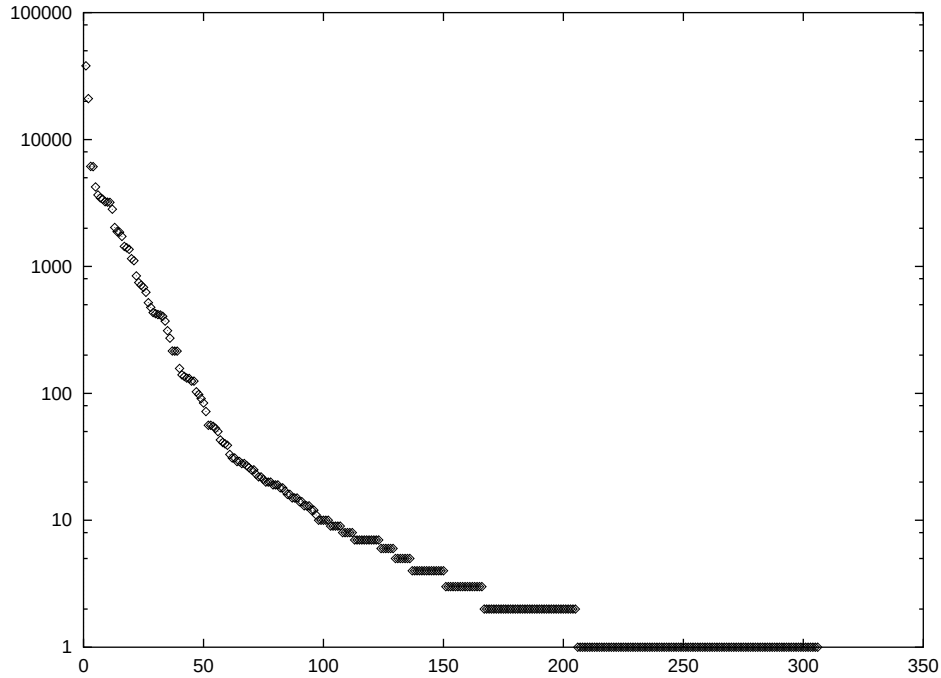

Figure 3.19: Zipf distribution of classes: x-axis is rank of class, sorted by frequency, y-axis is frequency (logarithmic scale) in the data set.

data sets are comparatively small. In the present research, we could have chosen to restrict our development set for parameter tuning to a small part of the training data. However, the frequencies of classes have a typical Zipf distribution (see Figure 3.19). Of the total of 319 non-default local classes in our standard data set there is one class (`NP-SBJ`) with frequency 38,100 and one (`NP-OBJ`) with frequency 20,898, but 40 with frequency 2 and 106 with frequency 1. We felt that due to this uneven distribution, restricting ourselves to a small development set would mean excluding most of the challenging cases, maybe not from the engineer's point of view but more from the point of view of the linguist. In addition, the smaller the development set, the higher the risk that the best settings found do not carry over to the much larger data set we are ultimately interested in. We therefore preferred to use a large development set and to restrict the number of tested parameter combinations.

3.2.2 Performance measures

In many machine learning experiments, "best" performance means the one with the best generalization accuracy on previously unseen test instances, i.e. with the highest number of correctly classified instances divided by the total number of instances. When applying machine learning to language data however, we frequently see that other measures are used in addition to or instead of accuracy.

For evaluating grammatical relations, precision and recall[18] have been used by e.g. Lin (1995), Carroll, Minnen, and Briscoe (1998) and Ferro, Vilain, and Yeh (1999). (Labeled) relation precision then means the number of *correctly* predicted relations divided by the *total* number of *predicted* relations whereas (labeled) relation recall means the number of *correctly* predicted relations divided by the *total* number of relations as given by the gold standard, i.e. in our case the relations derived from the treebank parse.

When comparing performance (for example with different parameter settings) it is useful to have only one figure to compare instead of two, precision and recall, which usually show a trade-off. For this reason, we also use the F_β measure (van Rijsbergen, 1979) which combines precision (P) and recall (R): $F_\beta = \frac{(\beta^2+1)\cdot P\cdot R}{\beta^2\cdot P+R}$
As is also common in the chunking literature we use $\beta = 1$ which gives no preference to either precision or recall.

3.2.3 Tenfold cross validation and significance

We randomize the sentences in our data set and split them up into ten *folds* of 2175 sentences each (one with only 2172) from which we extract ten folds in the intermediate format as explained in Section 3.1.2. We then construct instances for our machine learner from each intermediate format fold (cf. Section 3.1.3). We use these folds to conduct *tenfold cross validation* experiments: in each of ten learning runs, a different one of the folds is the test data whereas the other nine constitute the training data (Weiss and Kulikowski, 1991). This way we get ten sets of precision, recall and F_β scores. In reporting the results, we will use the average of these ten values.

As precision denotes a percentage of the number of relations predicted, which varies slightly per fold, the average of ten precision values is slightly different from the precision we would get if we merged all ten folds and computed precision over the total. The same holds for recall because each fold is based on the same number of sentences which does not necessarily translate into the same number of relations. However, as variance is low (typically 1–2%), this difference is small.

To test the difference in results of two tenfold cross validation experiments for statistical significance, we will use a one-tailed t-test on the two sets of ten F_β scores. Yeh (2000b) discusses alternative tests and proposes computationally-intensive randomization tests, referring to Cohen (1995) and Noreen (1989). These tests involve shuffling and reassigning the differing responses from the two (variants of) systems. If there are n different responses, there are 2^n different ways to do this. Given that our test sets contains hundreds of thousands of instances, and that variants of our learner often predict different classes for thousands of them, exhaustive enumeration is not feasible. It can be approximated by randomized sampling but even a representative sample would be huge.

[18] These or similar measures (sometimes under different names) are also used in other fields, e.g. Information Retrieval or medical tests (sensitivity/predictive value).

	accuracy	precision	recall	F_β
always predict "no relation"	78.42	100	0	0
always predict NP-SBJ	6.85	6.85	30.73	11.20
most probable class for focus chunk type/PoS	78.42	100	0	0
most probable class for focus word	78.20	31.21	1.07	2.07
most probable class for distance	82.81	49.43	37.30	**42.51**

Table 3.2: Some possible baselines. The simple heuristic to predict NP-SBJ for every chunk at distance -1 and NP-OBJ for every chunk at distance 1 ("most probable class for distance") already yields an F-score of 42.51.

3.2.4 Baselines

An important concept in machine learning is the *baseline*. This is the performance of the simplest classifier one can think of. For part-of-speech tagging for example the baseline is usually taken to be the accuracy achieved when predicting always the most probable tag of a word. It is only with respect to this baseline value that performance results of learners are meaningful. For grammatical relation finding, the definition of the baseline is less clear than for PoS tagging. The most probable class of an instance is "no relation". Always predicting "no relation" would already result in an accuracy of 78.42% on the data set described in Section 3.2.1. However, precision would be undefined or 100%, recall would definitely be 0% and thus F_β would be zero too. A more reasonable baseline is achieved when always predicting the most frequent relation (NP-SBJ: subject). F_β would then be 11.20 (cf. first two rows of Table 3.2). Parallel to the PoS tagging baseline, one could also predict the most probable relation for each chunk type. However, as this is always "no relation", it would again result in $F_\beta = 0$. Predicting the most probable relation for each focus word results in $F_\beta = 2.07$. The precision of 31.21% seems partly due to the strategy to assign capitalized common nouns (like "Revenue" or "Sales") the subject relation. Capitalization here implies that the word occurs at the beginning of a sentence. Predicting the most probable relation for each distance yields $F_\beta = 42.51$. This strategy follows three simple rules: predict "NP-SBJ" for every chunk at distance -1, "NP-OBJ" for every chunk at distance 1 and "no relation" for all other chunks.

3.3 Summary

In this chapter we introduced the data set and the experimental setup that will be used in most experiments reported in the main part of this thesis. We described the original treebank data, explained the complex conversion from phrase structure trees to dependencies between chunk heads and motivated the choices we made for this conversion. We likewise described the intermediate format and introduced the preliminary form of our instances. Finally we explained the experimental conditions (size and origin of data set, tenfold cross

validation), performance measures (precision, recall and F-score), and statistical significance test and computed several baselines. In the following chapter we will introduce the Memory-Based algorithm that we will use for our experiments and we will report on first experiments conducted with the setup just described.

Chapter 4

Memory-Based Learning and optimization of its parameters

This chapter consists of two main parts. In the first part, we will describe the Memory-Based Learning algorithms and their implementations that are used for the experiments reported in this thesis. In the second part, we will apply these algorithms to our data in order to find out which algorithm with which parameter setting performs best. In addition to these two main parts this chapter also contains an in-between section in which we develop a preprocessor to speed up subsequent experiments.

4.1 Memory-Based Learning: theory

The central idea of Memory-Based Learning (MBL) is to store *all* training instances in memory during learning. This memory is called the instance base. For testing, those training instances that are most *similar* to the test instance are retrieved from memory and their labels are used to assign a label to the test instance. This direct use of all training instances (lazy learning) contrasts with the eager learning of e.g. decision tree or rule learning algorithms which derive an abstract representation from the training instances and then use only this representation when processing test instances.

MBL is also known as similarity-based, exemplar-based, example-based, analogical, case-based, and instance-based learning (Stanfill and Waltz, 1986; Cost and Salzberg, 1993; Kolodner, 1993; Aha, Kibler, and Albert, 1991; Aha, 1997). All details of algorithms and available parameters described in this section and used in the next ones are based on the TiMBL[1] software package (Daelemans et al., 2001) version 4.1 unless noted otherwise.

[1] Tilburg Memory-Based Learner. See `http://ilk.kub.nl/` under "Software".

4.1.1 The IB1 algorithm

In the classic k-Nearest Neighbor (k-NN) algorithm (Cover and Hart, 1967), an instance can be thought of as a vector defining a unique point in a high-dimensional space. Similar instances are close together in this space. When a test instance needs to be classified, the algorithm looks for its k closest training instances (the k Nearest Neighbors) and assigns the class with the highest frequency in this Nearest Neighbor set (majority voting). If all features are numeric, distance between two instances can be computed as Euclidean distance (possibly after normalization of features). The distance $\Delta(X, Y)$ between two instances X and Y which have only numeric features is shown in Equation 4.1. n is the number of features and x_i is the value of the ith feature of instance X.

$$\Delta(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.1}$$

However in NLP most features are symbolic. In the IB1 algorithm (Aha, Kibler, and Albert, 1991), distance (or conversely, similarity) is measured as the number of features on which two instances differ (or conversely, for which they have the same value). The distance between two instances which have only symbolic features is formally defined in Equations 4.2 and 4.3 where $\delta(x_i, y_i)$ is the distance in the ith feature of the instances. This metric is called the overlap metric.

$$\Delta(X, Y) = \sum_{i=1}^{n}\delta(x_i, y_i) \tag{4.2}$$

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases} \tag{4.3}$$

4.1.1.1 The number k of Nearest Neighbors and the resolution of ties

In the original k-NN algorithm the parameter k referred to the number of Nearest Neighbors that should contribute to the classification of a test instance. If features are numeric, values are real numbers and distance is Euclidean, it rarely happens that two neighbors have the same distance. With integer values or with symbolic features and the overlap metric, however, this happens regularly. Therefore in the TiMBL implementation of IB1, k refers to the number of distances from which the elements of the Nearest Neighbor set are gathered. If e.g. there are three training instances at a distance of 0 to the test instance and four at a distance of 1, the NN set would contain three instances if $k = 1$ and seven if $k = 2$. According to majority voting, the final classification of the test instance is the class that is most frequent in this NN set. In case of a tie (i.e. two or more classes are equally frequent), the tied class that has the highest frequency in the complete training set is taken. If this still does not solve the tie, the tied class that was first seen in the training material is chosen. In general, odd values of k help preventing ties if most of the distances are associated with only one instance.

4.1.1.2 Feature weights and the IB1-IG algorithm

Especially for tasks with many features, a common intuition is that some features are more important than others for classification. To capture this intuition, features are assigned weights in the IB1-IG algorithm (Daelemans and Van den Bosch, 1992) and the distance is computed as in 4.4 where w_i is the weight of the ith feature. This metric is called weighted overlap.

$$\Delta(X, Y) = \sum_{i=1}^{n} w_i \times \delta(x_i, y_i) \tag{4.4}$$

There are many possible ways to determine the weights of the features. Four of them are implemented in the TiMBL software package and will be described here.

Quinlan (1986) proposes Information Gain (IG) for feature ordering in a decision tree. Daelemans and Van den Bosch (1992) use it for feature weighting in IB1-IG. IG is a concept from Information Theory. It is based on entropy (defined in Equation 4.5), in our case on the entropy of the class distribution, which expresses the uncertainty of the learner about which class to predict for an instance. If there is less uncertainty if we know the value of a feature than if we do not know it, the feature contains information. Thus the Information Gain that a feature brings about is defined as the difference in entropy with and without knowledge of the feature value (see Equation 4.6). C denotes the set of class labels and V_i are the values of the ith feature.

$$H(X) = -\sum_{x \in X} P(x) log_2 P(x) \tag{4.5}$$

$$w_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v) \tag{4.6}$$

Quinlan (1993) claims that IG overestimates the importance of features with many values and introduces the variant Gain Ratio (GR) in which IG is normalized by dividing by the entropy of the feature's values, the so-called split info. GR is defined in Equation 4.7. This normalization is relevant for many NLP tasks in which words are used as feature values, as there are many different words.

$$w_i = \frac{H(C) - \sum_{v \in V_i} P(v) \times H(C|v)}{H(V_i)} \tag{4.7}$$

An alternative to IG and GR is the Chi-squared weighting (χ^2) and its normalized version Shared Variance (SV), proposed by White and Liu (1994). χ^2 is defined in Equation 4.8. O_{nm} and E_{nm} are the observed and expected frequencies from a contingency table t which records how often each feature value occurred with each class.

$$w_i = \chi_i^2 = \sum_{n=1}^{|V_i|} \sum_{m=1}^{|C|} \frac{(E_{nm} - O_{nm})^2}{E_{nm}} \tag{4.8}$$

O_{nm} is just the content of the cell t_{nm}. E_{nm} is the number of cases which one would expect in cell t_{nm} if the null hypothesis (of no predictive association between feature and class) were true. It is defined in 4.9. $t_{.m}$ is the sum over column m of the table. This is equal to the number of instances with class c_m. $t_{n.}$ is the sum over row n of the table, i.e. the number of instances with value v_n. $t_{..}$ is the sum over all the cells of the table, i.e. the total number of instances.

$$E_{nm} = \frac{t_{.m} t_{n.}}{t_{..}} \tag{4.9}$$

SV normalizes χ^2 by correcting for the degrees of freedom. N is the number of instances.

$$w_i = SV_i = \frac{\chi_i^2}{N \times (\min(|C|, |V_i|) - 1)} \tag{4.10}$$

A problem with all of the above feature weighting schemes is that they compute the weight of a feature independently from all the other features. In real tasks however, features are often not independent, so that the computed weights do not reflect feature informativity in context.

4.1.1.3 Numeric features and the Modified Value Difference Metric

The overlap metric and the weighting schemes as introduced above can only be applied to symbolic features. As some learning tasks are best described by a mixture of symbolic and numeric features, the following addition is implemented to deal with numeric features:

$$\delta(x_i, y_i) = \frac{|x_i - y_i|}{\max_i - \min_i} \quad \text{if } x_i \text{ and } y_i \text{ numeric, else see 4.3} \tag{4.11}$$

For weight computation, numeric features are first discretized into a number of bins,[2] and each bin is then treated like one symbolic value.

In the standard overlap metric, two symbolic feature values are either identical or different. However one frequently has the intuition that some values are more (dis)similar than others. For example although "Monday" is not identical to "Tuesday", the two are less different than e.g. "Monday" and "horse". To capture this intuition, Stanfill and Waltz (1986) and Cost and Salzberg (1993) developed the (Modified) Value Difference Metric (MVDM) which computes the difference between two feature values by comparing the associated class distributions, as defined in Equation 4.12 where $v_1, v_2 \in V_i$.

$$\delta(v_1, v_2) = \sum_{c \in C} |P(c|v_1) - P(c|v_2)| \tag{4.12}$$

[2]In TiMBL 4.1, the space between the largest and the smallest value of a feature, as seen in training data, is split into 20 equally-sized subspaces (bins). If values are not equally distributed, this results in some "full" and possibly some empty bins. In TiMBL 4.2 bins are chosen in such a way that they contain equally many values (as far as possible). The number of bins is a parameter. Note that in any case, the discretization is used only for weight computation, not for classification.

A known problem is that the MVDM value of rare feature values is based on very little evidence. In the extreme case of values that occur only once, the modified value difference is either minimal (0: values are treated as identical) or maximal (2: completely different) depending on whether the values occur with the same class or not.

One of the effects of MVDM is that possible distances between instances get much more diversified. This in turn means that the number of instances in the Nearest Neighbor set is usually lower, so that classifications are based on less evidence. To counteract this effect, it is normally advantageous to use a larger number of k with MVDM. In the TiMBL implementation, it is possible to indicate per feature which metric (overlap, numeric or MVDM) should be used.

4.1.1.4 Distance weighted class voting

If k is large, many training instances influence the classification through their vote in the majority voting. On the one hand this might increase robustness against e.g. annotation errors. On the other hand, it ignores the fact that some NN are more similar/closer to the test instance than others and thus their vote should be more important. This intuition is implemented by distance weighted class voting. Each NN gets assigned a weight that decreases with increasing distance to the test instance. Classification is then determined through a weighted voting. There are many monotonously decreasing functions that could be used for distance weighted class voting. Three of them are implemented in TiMBL: Inverse Linear (Dudani, 1976), Inverse Distance (Dudani, 1976) and Exponential Decay (based on Shepard (1987)). The weight functions are defined in Equations 4.13, 4.14 and 4.15, respectively. w_j is the weight of the neighbors at the jth distance ($1 \leq j \leq k$) and d_j denotes the distance from these neighbors to the test instance. ϵ is a small constant[3] which avoids division by zero, α is a parameter that influences the slope of the decay function and β is set to 1 in the TiMBL implementation.

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1} & \text{if } d_k \neq d_1 \\ 1 & \text{if } d_k = d_1 \end{cases} \tag{4.13}$$

$$w_j = \frac{1}{d_j + \epsilon} \tag{4.14}$$

$$w_j = e^{-\alpha d_j^{\beta}} \tag{4.15}$$

4.1.2 The IGTree algorithm

As the previous sections showed, the IB1-IG algorithm is a very flexible tool. However it has the disadvantage that while learning is relatively fast (as it just consists of storing all

[3] ϵ is defined by the C compiler as the smallest number for which $1 + \epsilon \neq 1$.

```
 1    2    3    4        5   class

-1 arrived -  John     NP  NP-SBJ                        John arrived on Friday.
 1 arrived on Friday   PNP PP-TMP
-1 depends -  Peter    NP  NP-SBJ                        Peter depends on Mary.
 1 depends on Mary     PNP PP-CLR
-1 saw     -  Mary     NP  NP-SBJ                        Mary saw John on Monday.
 1 saw     -  John     NP  NP-OBJ
 2 saw     on Monday   NP  PP-TMP
-1 left    -  Peter    NP  NP-SBJ                        Peter left at midnight.
 1 left    at midnight PNP PP-TMP
```

Figure 4.1: Some toy instances for finding GRs and the sentences they are derived from.

instances and computing the feature weights), testing is rather slow. Although internally, testing is implemented more efficiently than a sequential comparison of the test instance to all training instances (Daelemans et al., 2001), it might still be too slow for practical applications. Therefore Daelemans, Van den Bosch, and Weijters (1997) introduced IGTree, a fast approximation of IB1-IG. The training instances are stored in an oblivious decision tree. An oblivious decision tree is one in which each level contains tests on the same feature (as opposed to e.g. C4.5 (Quinlan, 1993) in which different branches of the tree can test different features at the same level). In IGTree, features are associated with levels in the tree according to one of the weighting schemes introduced in Section 4.1.1.2, with the highest-weighted feature at the top. However while in IB1-IG the relative differences between the weights are important, IGTree only uses the ordering they induce. As in all decision trees, branches in the tree are annotated with a feature value. Every path in the tree from the root to some node represents the subset of all the training instances for which the features corresponding to the levels have these values. Each node in the tree carries the class distribution of the training instances it represents. The class with the highest frequency in this distribution is the node's default class. Figures 4.1 and 4.2 show an example.

During testing, the tree is traversed from the root downwards, following the branches that match the test instance's feature values. If a leaf node is reached or all branches downwards mismatch (e.g. due to a previously unseen feature value), the node's default class is returned as classification of the test instance. The tree can be made smaller without effect on classification by recursively pruning each leaf node whose default class is the same as its parent's default. Especially for large data sets, IGTree can be orders of magnitude faster in testing than IB1-IG, as testing time is maximally proportional to the number of features times the (average) branching factor, whereas testing time of IB1-IG is proportional to the number of features times the number of instances. The downside is that IGTree usually performs worse than IB1-IG if there are several features with similar weights, or if the features with the highest weights regularly have values or value combinations in test instances that did not occur in training.

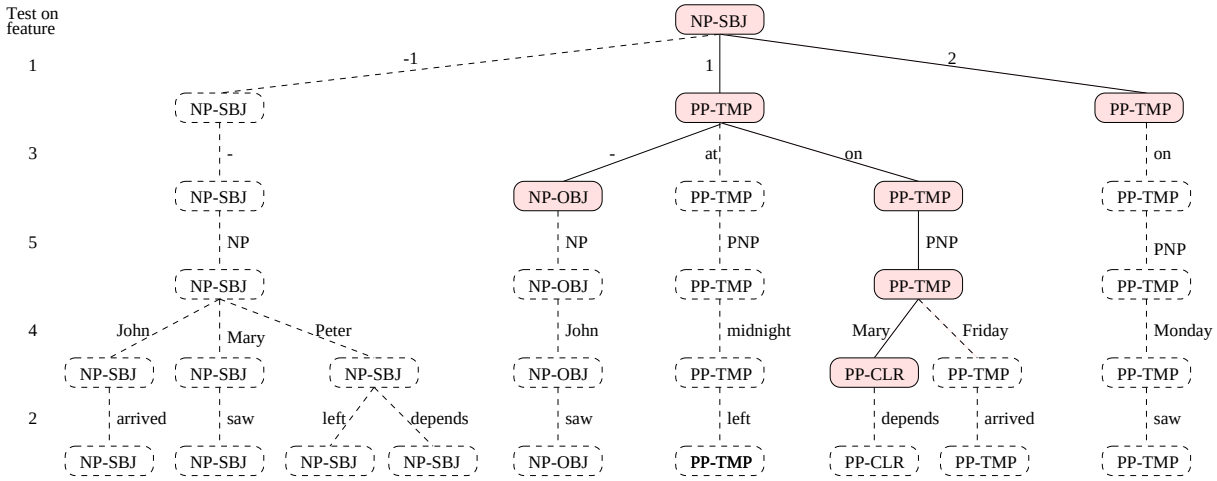

Figure 4.2: The IGTree for the toy instances. The feature order is 1, 3, 5, 4, 2. The class distributions on nodes are suppressed, only the default class is shown. All the dashed parts are deleted during the pruning phase.

4.1.3 The hybrid TRIBL

As a compromise between classification speed and classification accuracy, Daelemans, Van den Bosch, and Zavrel (1997) introduced the hybrid TRIBL. This algorithm has an extra parameter q. For the q highest weighted features, an IGTree is built. Each leaf of the tree points to an instance base of all the instances that are represented by this node. During testing, first the IGTree part is traversed, then an IB1-IG-style search is performed on the instance base belonging to the node that was reached, if any. All extra options that can be applied to IB1-IG (like MVDM, distance weighted class voting etc.) can also be applied to this part of the TRIBL data structure. However the problem of unseen values/value combinations in testing remains for the IGTree part. A rule of thumb is to set q to the level where the relative difference between feature weights starts to become small.

4.1.4 Summary

We introduced the MBL algorithms that are implemented in the TiMBL software package. IB1-IG is the most flexible one, supporting four feature weighting and four class voting schemes, three similarity metrics, including one for numeric features, and the k parameter to control the amount of evidence considered. IGTree is the fastest algorithm of the three but supports only feature ordering. The hybrid algorithm TRIBL allows the user to control the trade-off between performance and speed. Previous applications (e.g. chunking) have shown that the algorithms can handle large number of instances with many features, feature values and classes in reasonable time and with manageable memory requirements. This is an important prerequisite for our task. We saw in Chapter 2 that there are many possible information sources for finding GRs (which translates into many features), that words

are one of them (which translates into many values for at least some features) and that there are many different types of GRs (see e.g. Section 2.2.1 for semantic (sub)types of adjuncts, and Section 2.6.6 for types of complements), which translates into many classes. In addition it is important that our algorithm can handle many instances because only in that case will we be able to observe a statistically significant effect of the less important information sources.

Due to a different bias, different learning algorithms are known to perform differently on the same data. However we hope that, in general, addition of an informative feature would increase performance for all algorithms, although probably by different amounts. Therefore the conclusions that we draw from experiments with the MBL algorithms introduced above should also be of interest to researchers working with other learning paradigms.

4.2 Preprocessing: restricting the search space

In Chapter 3 we described the data and the experimental setup. In the previous section, we introduced the MBL algorithm. To give an example of the way MBL classifies new instances, Table 4.1 shows the Nearest Neighbors of the six instances from Table 3.1 (for our example sentence) when training on our data set (which does not contain this sentence) using the TiMBL default setting (IB1-IG, Gain Ratio weights, $k = 1$, overlap metric, majority voting). The last three rows show the distance from the test instance to the NN, their class distribution and the classification resulting from majority voting (which in this case is simple as all NN agree on the class).

The first row of Table 4.2 shows results if we apply TiMBL with the default settings to the data in a tenfold cross validation experiment. The sixth and seventh column show the amount of RAM and the CPU time (in hours and minutes) needed. TiMBL allows to save the instance base in its compressed internal format to disk after training and to read it back into RAM for testing. This option was used and the maximal amount of RAM needed for any of the ten folds during testing was noted (using the UNIX command `ps`). This information is useful if one wants to design practical applications, to get an idea of the hardware needed. Time is the time needed on a Pentium III, 733 MHz, 1024 MB RAM, 133 MHz SDRAM, for the complete tenfold cross validation experiment and was measured with the UNIX command `time`.[4] The precise values are less relevant but what is important is the proportional increase or decrease in time when testing various non-default settings or new features.

The eighth to tenth column of Table 4.2 show the performance of the experiment. The F_β score is already much higher than any of the baselines from Table 3.2. Precision is slightly higher than recall. This is due to the overwhelming majority of instances with the default "no-relation" class. As we can see in the second and fourth column of Table 4.2, only about

[4] This time includes system and user time. Note that it also contains the time for saving and reading back the instance base which is not strictly necessary.

features			Nearest Neighbors					
	focus #	1	104144	272606/...	272607/...	289015	182360/...	494881/...
	verb #	2	104158	272610/...	272610/...	289013	182356/...	494873/...
	dist.	3	−5	−3	−2	+1	+2	+3
	VCs	4	0	0	0	0	0	0
	verb	5	join	*elected/ succeeds/ ...*	*elected/ succeeds/ ...*	join	*viewed/ touts/ describes/ defines*	*vote/ redeem/ take*
context −1	prep.	6	–	–	–	–	–	*at/of/ from*
	word	7	–	,	years	join	*offering/ service/ flunky/ failure*	*meeting/ stock/ steelmaker*
	PoS	8	–	,	NNS	VB	NN	NN
	chunk	9	–	–	NP	VP	NP	PNP
focus chunk	prep.	10	–	–	–	–	as	–
	word	11	*Kemper*	years	old	*sort*	*defense/ hour/ one/ company*	*13/ 8/ 31*
	PoS	12	NNP	NNS	JJ	NN	NN	CD
	chunk	13	NP	NP	ADJP	NP	PNP	NP
context +1	prep.	14	–	–	–	of	–	–
	word	15	,	old	,	*club*	*that/ you/ who/ that*	.
	PoS	16	,	JJ	,	NN	*WDT/ PRP/ WP/ WDT*	.
	chunk	17	–	ADJP	–	PNP	NP	–
NN distance			0.084703	0.046646	0.046646	0.183440	0.328053	0.258882
NN distribution			NP-SBJ: 1	-: 25	-: 29	NP-OBJ: 1	PP-CLR: 4	NP-TMP: 3
classification			NP-SBJ	-	-	NP-OBJ	PP-CLR	NP-TMP

Table 4.1: The Nearest Neighbors of the instances in Table 3.1, their distance, their class distribution (by coincidence always unanimous) and the resulting classification (which in these cases is also the correct class). Mismatching feature values are marked in italics. Some of the sentence parts from which these neighbors are derived (verb and focus head marked in bold face): **Kemper**, the biggest holder of senior SCI TV bonds, has refused to **join** the bond-holders committee ... Mr. Conway, 42 **years old**, was **elected** chairman, president and chief executive of Manhattan Life Insurance Co. ... ... he also gave Mr. Peterson $50,000 to **join** a **sort** of investment club ... ..., analysts have **viewed** the rights offering as a takeover **defense** that ... Stockholders will **vote** on the proposal at a meeting Dec. **13**.

120.000 of the 556.000 instances actually encode a relation. This makes the learner rather cautious about predicting relations. It also means that most of its work (as measured by the time and memory requirement) is invested into instances which do not really count for the final performance. In the following we therefore develop a preprocessor to reduce this work.

On the algorithmic level, the learner classifies instances. On the conceptual level however, the task is a search task: the search for grammatical relations of verbs in a sentence. The instances define the search space. As there is only one verb chunk in our example sentence from Figure 3.18 and the sentence is relatively short, the number of instances in Table 3.1 is rather low. But in general, in a sentence of n chunks, of which m are verbs, we would need to generate $m \times (n-1)$ instances. On average a sentence of k tokens (words plus punctuation) contains $0.5 \times k$ chunks, of which $0.1 \times k$ are verb chunks. Thus we would generate $0.05k^2 - 0.1k$ instances. Although this is quadratic, it does not sound too bad: for the average sentence length of 23.7 tokens, we would generate 25.7 instances. Still, the further apart two chunks are, the less likely it is that they are related. And the further apart two chunks are which have a relation, the less likely it is for the learner to find it. Thus using more instances might in fact not even increase performance. Therefore we tested whether we could ignore the instances with the most "distant" chunk pairs (i.e. restrict the search space) without hurting performance. The first question then is: what is "distant"? The preliminary feature representation in Section 3.1.3 already contains two distance measures: distance in elements (1 meaning adjacent) and distance in number of intervening verb chunks.

Table 4.2 shows results for various values of both distance measures. A restriction on elements of i/j means that the focus chunk may maximally be the ith element to the left or the jth to the right of the verb chunk. A restriction on verb chunks of i/j means that there may be maximally i intervening verb chunks between a focus chunk to the left of the verb, and j between a focus chunk to the right of the verb chunk. Table 4.2 shows how severely various values restrict the search space in terms of instances, absolute (column 2) as well as in percents (column 3) of the number of instances when not applying any restriction and in terms of relations that are covered by the (restricted) search space, absolute (column 4) as well as in percents (column 5). The number of instances is correlated to memory and time requirements (columns 6 and 7). The number of relations covered corresponds to the upper bound on the theoretically possible recall.

We see that the distance in elements places a much more severe restriction on the search space than the distance in verb chunks. It is for this reason that we tested only symmetrical restrictions ($i = j$) for the former but also asymmetric ones for the latter. In general, a smaller search space leads to a higher precision but a lower recall. This is the precision/recall trade-off known from many problems. The best F_β values are reached with a 6/6, 7/7 or 8/8 restriction on elements or a 1/0 restriction on verb chunks respectively. The latter achieves $F_\beta = 74.40$ which is a significant improvement ($p < 0.01$, $t = 2.954$) over the performance without any restriction. The percentage of instances fulfilling the

	# inst.	% inst.	# rel.	% rel.	MB	h:m	precision	recall	F_β
no restriction (baseline)									
	555539	100.00	119836	100.00	154	2:30	75.77	72.19	*73.94*
restriction on elements									
1/1	90423	16.27	70139	58.52	31	0:07	86.98	51.24	64.49
2/2	155545	27.99	91603	76.44	48	0:15	83.66	62.98	71.86
3/3	215738	38.83	100785	84.10	64	0:24	81.39	66.81	73.38
4/4	270769	48.73	106169	88.59	78	0:32	80.03	68.74	73.95
5/5	319245	57.46	109928	91.73	91	0:39	79.15	70.03	74.31
6/6	360716	64.93	112581	93.94	101	0:48	78.43	70.83	**74.44**
7/7	396287	71.33	114545	95.58	111	0:56	77.87	71.31	**74.45**
8/8	426206	76.71	115935	96.74	119	1:05	77.49	71.65	**74.45**
restriction on verb chunks									
0/0	285587	51.40	110630	92.31	83	0:49	78.40	70.56	74.27
0/1	364611	65.63	112512	93.88	104	1:08	77.92	70.78	74.17
1/0	361342	65.04	115650	96.50	103	1:06	77.20	71.79	**74.40**
1/1	440366	79.26	117532	98.07	123	1:27	76.71	71.99	74.27
1/2	476005	85.68	117860	98.35	134	1:41	76.59	71.98	74.21
2/1	475892	85.66	118922	99.23	133	1:40	76.24	72.13	74.13
2/2	511531	92.07	119250	99.51	143	1:52	76.20	72.15	74.12
3/3	540075	97.21	119693	99.88	151	2:11	76.00	72.18	74.04

Table 4.2: Different ways of reducing the search space. All experiments use TiMBL's default setting (IB1-IG, Gain Ratio, overlap, $k = 1$, majority voting).

[$_{NP}$ **PACIFIC GAS & ELECTRIC CO.**] , [$_{NP}$ San Francisco] , [$_{NP}$ electric , gas and water supplier] , [$_{NP}$ annual sales] [$_{NP}$ $ 7.6 billion] , [$_{NP}$ some minor damage] {$_{PNP}$ to headquarters } , [$_{NP}$ undetermined damage] {$_{PNP}$ to four nearby substations } , [$_{NP}$ severe structural damage] {$_{PNP}$ to a major power plant } {$_{PNP}$ at Moss Landing } , [$_{NP}$ extensive damage] {$_{PNP}$ to gas lines and electric lines } , [$_{NP}$ 400,000 residences] {$_{PNP}$ without electricity } and [$_{NP}$ 69,000] {$_{PNP}$ without gas } , [$_{VP}$ **can not reconnect**] [$_{NP}$ electricity] [$_{SBAR}$ until] [$_{NP}$ it] [$_{VP}$ is] [$_{ADJP}$ certain] [$_{NP}$ there] [$_{VP}$ are] [$_{NP}$ no gas leaks] , [$_{NP}$ no predictions] [$_{PP}$ on] [$_{ADVP}$ when] [$_{NP}$ this] [$_{VP}$ will happen] .

[$_{NP}$ Other technology stocks] [$_{NP}$ that] [$_{VP}$ were] [$_{ADJP}$ weaker] [$_{VP}$ **included**] [$_{NP}$ Intel] , [$_{NP}$ which] [$_{VP}$ fell] [$_{NP}$ 1 1/4] {$_{PNP}$ to 33 1/2 } {$_{PNP}$ on 1.9 million shares } , [$_{NP}$ Mentor Graphics] , [$_{ADVP}$ down] [$_{NP}$ 3/4] {$_{PNP}$ to 16 1/4 } {$_{PNP}$ on 1.6 million shares } , [$_{NP}$ Sun Microsystems] , [$_{NP}$ which] [$_{VP}$ slipped] [$_{NP}$ 3/8] {$_{PNP}$to 18 1/4 } , and [$_{NP}$ **MCI Communications**] , [$_{ADVP}$ down] [$_{NP}$ 1] {$_{PNP}$ to 42 3/4 } .

Figure 4.3: Between the bold face chunks: A subject relation spanning 28 elements and an object relation spanning 24.

restriction is nearly the same for 6/6 and 1/0 (64.93% vs. 65.04%). The percentage of relations included in these instances however is more similar between 8/8 and 1/0: 96.74% vs. 96.50%. This means that the restriction on verb chunks is slightly more effective in excluding non-relation instances. As its time and memory requirements are also within the range of the 6/6 to 8/8 restriction, we chose to use the 1/0 verb chunk restriction for all following experiments. It can be thought of a preprocessor that classifies all instances that do not fulfill the requirement directly as "no relation" and invokes the real machine learning classifier for the others. This decreases the memory needed from 154 to 103 MB and the time from two and a half to one hour, which is clearly worthwhile.

For the following sections, readers should keep in mind, first, that the upper bound on recall is 96.50%, and second, that this is not a principled but a practical decision. Indeed we could always return to more/all instances, although using all data slightly *decreases* performance (from 74.40 to 73.94). In fact, it is a good sign that performance decreases only this little, given that most of the additional data is "useless" in the sense that very few of its instances (2.16%) indeed encode relations. As a concluding remark on "distant" relations, let us have a look at three extreme examples from the Wall Street Journal Corpus. Figures 4.3 and 4.4 show relations spanning many elements (chunks plus isolated words/punctuation) or many verb chunks respectively. As the relations in the second and third example span too many verb chunks, they are handled by our preprocessor and (incorrectly) classified as "no relation". The first example is classified by the Memory-Based Learner as a subject relation, which is correct.

[$_{NP}$ **The man**] [$_{NP}$ who] [$_{VP}$ *wore*] [$_{PRT}$ out] [$_{NP}$ his shoes] [$_{VP}$ *wandering*] {$_{PNP}$ around Guadalajara } {$_{PNP}$ in 1958 } , [$_{VP}$ *describing*] {$_{PNP}$ in his travel book } " [$_{NP}$ Viaje a la Alcarria] " [$_{ADVP}$ how] [$_{NP}$ he] [$_{VP}$ *scrounged*] {$_{PNP}$ for food } and [$_{VP}$ *stayed*] {$_{PNP}$ in squalid inns } , [$_{ADVP}$ now] [$_{VP}$ **tours**] [$_{NP}$ Spain] {$_{PNP}$ in a Rolls-Royce } .

Figure 4.4: Between the bold face chunks: A subject relation spanning 5 verb chunks (in italics).

4.3 Memory-Based Learning: practice

In this section we will test the algorithms and parameters that we introduced in Section 4.1 on our data (restricted to a distance of 1/0 verb chunks, as explained in the previous section). We will start with the default setting that was also used in the previous section. This is: IB1-IG, Gain Ratio weights, the overlap metric, $k = 1$ and majority voting. We will then change one parameter at a time (Sections 4.3.1 to 4.3.6) to find its optimal value, which will be used in subsequent experiments. It is clear that this might not result in the overall best parameter setting, as some parameters are known to interact (e.g. k and MVDM, or k and distance weighted class voting). However an exhaustive search for the best setting is infeasible (cf. Section 3.2.1). The best setting found will be used in subsequent chapters in which we try to answer our central research question: what information is important for finding grammatical relations and why?

There are three reasons for approaching this question with the best parameter setting instead of the default one. First, we want our results to be also of practical value. Next to the data representation, parameter settings are a factor that influences performance and thus needs to be optimized. Second, using a suboptimal parameter setting means that the learner is not using the information contained in the data in an optimal way. Then there is a high probability that the new features (i.e. information) we add will also not be used optimally and that this influences our judgement about them.[5] Third, finding out why certain parameter settings perform better than others is also interesting from a theoretical point of view. This analysis is carried out in Section 4.3.7.

4.3.1 Feature weights

As explained in Section 4.1.1.2, all features have equal weights in IB1, but for IB1-IG, TiMBL implements four different weight schemes. Table 4.3 shows the performance of each of the weightings. Gain Ratio (the default) is best but, perhaps surprisingly, using no weights performs second best. The difference (74.40 vs. 74.03 F_β) is significant however ($p < 0.05$, $t = 2.288$). The other three weightings perform even worse. Chi-square is worst (59.81), Shared Variance is second worst (67.45) and Information Gain is in the middle (72.00).

[5]It is clear that there might still exist informative features that the algorithm is unable to exploit but using optimal parameters at least reduces this risk.

algorithm	weights	MB	h:m	precision	recall	F_β
IB1	none	103	8:33	77.66	70.72	74.03
IB1-IG	Info Gain	103	38:10	72.48	71.54	72.00
	Gain Ratio (default)	103	1:06	77.20	71.79	**74.40**
	Chi-square	103	69:09	59.38	60.24	59.81
	Shared Variance	103	5:01	67.98	66.93	67.45

Table 4.3: Performance with different feature weightings. The default, Gain Ratio, performs best. (1/0 restriction on intervening verb chunks, IB1(-IG), overlap metric, $k = 1$, majority voting)

global metric	weights	MB	h:m	precision	recall	F_β
overlap (default)	Gain Ratio (default)	103	1:06	77.20	71.79	*74.40*
MVDM	none (default)	107	9:56	78.74	73.73	76.15
	Gain Ratio	107	3:18	79.32	74.20	**76.67**

Table 4.4: Performance with different global metrics. As the default weighting for MVDM is different from that for overlap, we tried two different weightings in that case. (1/0 restriction on intervening verb chunks, IB1-IG, $k = 1$, majority voting)

The fourth column of Table 4.3 shows that the run times with the five weightings differ dramatically. However this is due to the implementation in which the internal organization of the instance base is based on a feature order according to the Gain Ratio divided by the number of feature values (see Daelemans et al. (2001) for details). This means that the more similar a weight ordering is to this internal order, the faster testing is performed. In following experiments we will continue to use Gain Ratio weights.

4.3.2 Global metric

We tested two global metrics: overlap and MVDM.[6] Table 4.4 shows the results. MVDM without weighting (which is the default for MVDM) performs significantly ($p < 0.001$, $t = 10.793$) better than the default overlap metric (76.15 vs. 74.40). MVDM with Gain Ratio performs even better (76.67, $p < 0.001$, $t = 3.886$). We will therefore use MVDM (with Gain Ratio weights) in all following experiments unless noted otherwise. Unfortunately this triples the run time of the experiment. Memory requirement rises only slightly, due to storage of MVDM tables.

4.3.3 The number k of Nearest Neighbors

As Table 4.5 shows, using larger values for k dramatically improves precision and therefore F_β on our data set (from 76.67 to 79.62). We will therefore use $k = 9$ in all following

[6] See Section 4.3.4 about using the numeric metric for some individual features.

k	MB	h:m	precision	recall	F_β
1	107	3:18	79.32	74.20	*76.67*
3	107	4:26	84.43	74.29	79.03
5	107	4:52	85.29	74.43	79.49
7	107	5:11	85.67	74.26	79.56
9	107	5:26	85.97	74.14	**79.62**
11	107	5:40	86.10	73.99	79.59
13	107	5:52	86.12	73.76	79.46

Table 4.5: Performance with different values of k, the number of Nearest Neighbors considered. To prevent many ties, we used only uneven values of k. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, majority voting)

experiments unless otherwise noted. However this again decreases speed.

4.3.4 Feature-specific metrics

In Section 4.3.2, we saw that using MVDM as a global metric improves performance. However it might still be the case that another metric is better suited for some individual features. For most of the features, the alternative metric is overlap. The distance feature (number 3) and the "intervening verb chunks" feature (4) however can easily be interpreted as numeric features.[7] Table 4.6 shows the results of using the alternative metric for individual features (MVDM still being the global metric, i.e. for the other features). We treated features that contain the same kind of information (e.g. words, or PoS) as one group. We see that a numeric metric for the distance feature performs much worse than MVDM (see Section 4.3.7 for a discussion). It also takes much longer. For the (binary) intervening verb chunks feature, the numeric metric does not change performance, but increases speed. Using overlap decreases performance for all individual (types of) features, except for the "intervening verb chunks" feature. The difference is statistically significant only for the words and PoS ($p < 0.001$, $t = 9.382/t = 9.760$) and for the chunk types ($p < 0.05$, $t = 2.531$). In addition overlap doubles the run time for the word features. This means that we could use overlap for the distance, the intervening verb chunks, the verb and the prepositions features. This would also increase speed. However using MVDM for these does not harm performance, and as it is much easier to use one metric for all features, we will continue to use MVDM only. For practical applications, overlap would be an attractive option for these features.

[7] Due to our search space restriction, the "intervening verb chunks" feature (number 4) can only take the values 0 or 1. Therefore it is rather a binary feature than a true numeric one.

feature-specific metric	MB	h:m	precision	recall	F_β
none (all MVDM)	107	5:26	85.97	74.14	**79.62**
distance (3) as numeric	107	7:16	80.93	70.05	75.10
intervening verb chunks (4) as numeric	107	4:32	85.97	74.14	79.62
distance (3) as overlap	107	3:04	86.03	73.75	79.42
intervening verb chunks (4) as overlap	107	4:34	85.97	74.14	79.62
verb (5) as overlap	106	4:16	85.37	74.28	79.44
prepositions (6,10,14) as overlap	107	4:44	85.87	73.77	79.36
words (7,11,15) as overlap	104	9:49	83.92	73.10	78.14
PoS (8,12,16) as overlap	107	2:23	84.97	72.12	78.02
chunk types (9,13,17) as overlap	107	2:49	85.76	73.52	79.17

Table 4.6: No feature-specific metric performs better than global MVDM (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, $k = 9$, majority voting)

distance weights		MB	h:m	precision	recall	F_β
none (majority voting, default)		107	5:26	85.97	74.14	*79.62*
Inverse Distance		107	5:01	85.76	75.03	**80.04**
Inverse Linear		107	5:01	84.28	75.47	79.63
Exponential Decay	decay factor $\alpha = 2$	107	4:48	85.58	74.71	79.78
	5	107	5:01	85.61	74.75	79.81
	10	107	4:48	85.67	74.90	79.92
	20	107	4:48	85.61	75.20	80.07
	30	107	4:48	85.41	75.39	**80.09**
	40	107	4:48	85.07	75.45	79.97

Table 4.7: Performance of different distance weighted class voting schemes. Inverse Distance and Exponential Decay are better than majority voting. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$)

4.3.5 Distance weighted class voting

In Section 4.3.3 we saw that larger values for k are better for our task, i.e. that basing the decision on more Nearest Neighbors is better. Intuitively, however, the nearer a neighbor the more important it should be. This is achieved by distance weighted class voting. Table 4.7 shows results for different methods, one of them parametrized. We note that all methods perform better than majority voting. The improvements with Inverse Distance and Exponential Decay with factor 30 are even statistically significant ($p < 0.01$, $t = 2.695/t = 3.118$). We will use Exponential Decay with $\alpha = 30$ in all following experiments unless otherwise noted as it achieves the best performance and is also faster than the other alternatives.

algorithm		MB	h:m	precision	recall	F_β
IB1-IG (default)		107	4:48	85.41	75.39	**80.09**
IG-Tree		21	0:19	77.33	68.31	72.54
TRIBL	threshold (default) 1	274	4:40	85.41	75.39	**80.09**
	2	274	4:06	85.38	75.33	**80.04**
	3	274	1:34	85.30	75.04	79.84
	4	274	0:56	84.55	74.55	79.24
	5	274	0:55	84.53	74.56	79.23
	6	273	0:49	84.19	74.47	79.03

Table 4.8: Performance of different algorithms. As TRIBL has an extra theshold parameter, we also tested various values for this. (1/0 restriction on intervening verb chunks, Gain Ratio; for IB1 and TRIBL: MVDM, $k = 9$, Exponential Decay weighted class voting with $\alpha = 30$)

4.3.6 Algorithms

As explained in Section 4.1, our MBL implementation allows for three different algorithms: the default IB1(-IG), IGTree and the hybrid TRIBL. Table 4.8 shows the performance of the algorithms, with various threshold values for TRIBL. We see that IB1-IG and TRIBL with $q = 1$ perform best (80.09). IGTree is much faster and needs much less memory but scores about 7.5 points worse. As was to be expected, the performance of the hybrid TRIBL is in between those of IB1 and IGTree. On the other hand, TRIBL with $q = 2$ does not perform significantly worse ($t = 0.329$) than IB1-IG and it is a bit faster. However, it needs much more memory. We will continue to use IB1-IG for the experiments in the following chapters, but for practical applications for which speed is important, TRIBL would be a better alternative.

4.3.7 Discussion

In the previous sections we presented the effects of TiMBL's parameters on performance. In this section we will discuss why certain options work and others do not, and what this tells us about the task of finding grammatical relations.

4.3.7.1 Feature weights

The five different feature weighting schemes resulted in quite different performances (cf. Table 4.3, p. 90). Figure 4.5 shows the actual weights on our (restricted) data set. The y-axis scale is not important in this comparison, only the relative height of a bar is. We see that the weights are quite different. Information Gain and Chi-square predict the focus word feature *(wo)* to be the most important feature while Gain Ratio and Shared Variance favour the focus chunk type *(ch)*. This difference can be explained by the fact that Gain Ratio and Shared Variance are variants of Information Gain and Chi-square in

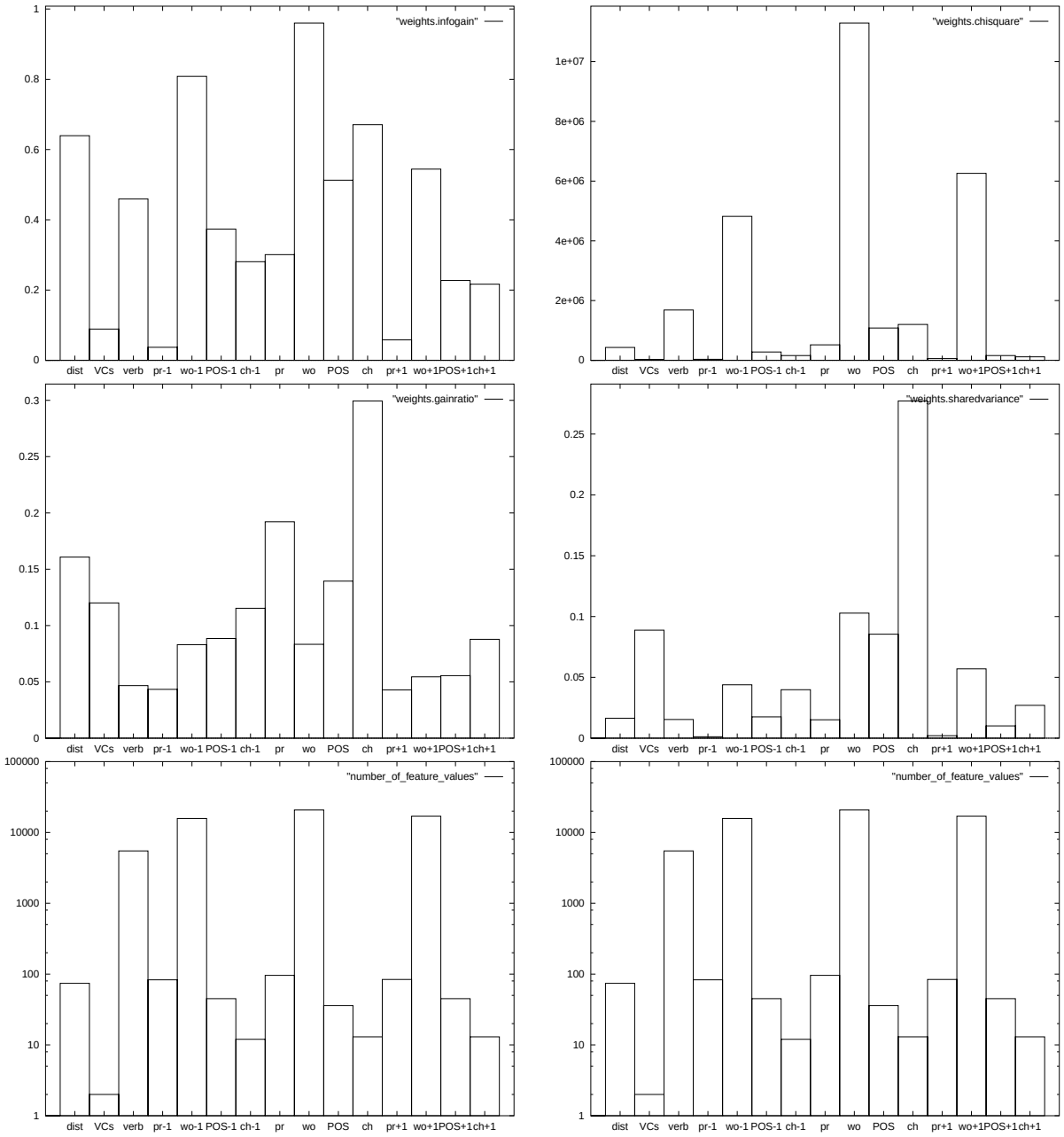

Figure 4.5: Feature weights assigned by Information Gain, Gain Ratio (left), Chi Square and Shared Variance (right) and number of values per feature (two times at bottom, with logarithmic y-axis). See Table 3.1 on page 72 for the list of features. The first two features ("administrative" features) are ignored.

which weights are normalized so that features with many values are not overestimated. The bottom of Figure 4.5 shows the number of values per feature.

There are many more differences between the four weightings. Chi-square, which performs

	−5	−4	−3	−2	−1	1	2	3	4	5
−5	0	*0.02*	*0.09*	0.19	1.21	1.65	0.89	0.42	0.24	0.18
−4	*0.02*	0	*0.06*	0.19	1.21	1.63	0.88	0.40	0.22	0.17
−3	*0.09*	*0.06*	0	0.19	1.20	1.58	0.84	0.35	0.20	0.16
−2	0.19	0.19	0.19	0	1.02	1.66	0.91	0.43	0.27	0.28
−1	1.21	1.21	1.20	1.02	0	1.65	1.27	1.28	1.29	1.30
1	1.65	1.63	1.58	1.66	1.65	0	1.17	1.44	1.55	1.58
2	0.89	0.88	0.84	0.91	1.27	1.17	0	0.52	0.70	0.76
3	0.42	0.40	0.35	0.43	1.28	1.44	0.52	0	0.20	0.26
4	0.24	0.22	0.20	0.27	1.29	1.55	0.70	0.20	0	*0.08*
5	0.18	0.17	0.16	0.28	1.30	1.58	0.76	0.26	*0.08*	0

Table 4.9: MVDM for the most frequent values of the distance feature. The lowest numbers (< 0.1) are marked in italics.

worse, does not agree on any of the four most important features with Gain Ratio, which performs best *(wo, wo+1, wo−1, verb* vs. *ch, pr, dist, PoS)*. In particular, both Chi-square and Shared Variance give too low a weight to the distance feature *(dist)*, which, as we already saw when computing baselines (Table 3.2, p. 75), is very informative.

This difference might be due to the fact that Chi-square adds only very little to the feature weight sum if a hapax value (i.e. one that occurs only once) occurs with the default class, but adds a lot if the hapax value occurs with a non-default class (because the E_{mn} factor is different). Information Gain treats both cases alike (because the entropy is zero). Chi-square as well as Shared Variance give low weight to the focus prepositional feature *(pr)* which ends up second-highest in Gain Ratio through the normalization.

In summary the focus chunk type, its preposition and the distance seem to be the most important information sources for finding grammatical relations. Although Gain Ratio seems to estimate feature weights better than the other weighting schemes, it does not perform much better than using no weighting at all. This seems to suggest that Gain Ratio is a far from optimal weighting scheme. A known flaw of all these weightings is that they compute the weight of each feature independently, whereas we know that the features are far from independent. For example a head noun *house* necessarily means that the PoS is `NN` and the chunk type is probably `NP` and maybe `PNP`.

4.3.7.2 Metrics

The MVDM metric increases performance over the overlap metric by more than two percent (Table 4.4, p. 90). This is mainly due to its better treatment of word and PoS-valued features (Table 4.6, p. 92). In addition, it is also much better than a numeric treatment of the distance feature. We will therefore take a closer look at the actual value differences for selected values of these features. Recall that low numbers mean that the two feature values are similar, high numbers mean they are dissimilar.

Table 4.9 shows the computed value differences for the distances −5 to 5 (five elements

	NN	VBZ	NNP	VBN	PRP	VBD	JJ	NNS	CD	RB
NN	0	0.77	*0.35*	0.75	0.55	0.77	0.70	*0.11*	*0.41*	0.78
VBZ	0.77	0	0.67	*0.10*	0.89	*0.05*	0.79	0.76	0.84	0.81
NNP	*0.35*	0.67	0	0.66	*0.35*	0.67	0.70	*0.28*	0.65	0.78
VBN	0.75	*0.10*	0.66	0	0.88	*0.11*	0.76	0.74	0.83	0.80
PRP	0.55	0.89	*0.35*	0.88	0	0.89	0.82	*0.48*	0.74	0.87
VBD	0.77	*0.05*	0.67	*0.11*	0.89	0	0.79	0.76	0.85	0.82
JJ	0.70	0.79	0.70	0.76	0.82	0.79	0	0.69	0.74	0.75
NNS	*0.11*	0.76	*0.28*	0.74	*0.48*	0.76	0.69	0	*0.48*	0.78
CD	*0.41*	0.84	0.65	0.83	0.74	0.85	0.74	*0.48*	0	0.81
RB	0.78	0.81	0.78	0.80	0.87	0.82	0.75	0.78	0.81	0

Table 4.10: MVDM for the most frequent values of the focus PoS feature. The lowest numbers (< 0.5) are marked in italics.

to the left to five to the right). The largest values (around 1.6) all occur between 1 and the negative distances. This is due to the configurational nature of English in which the position directly after the verb occupies a special place. Objects, for example, cannot occur in front of the verb.[8] However they sometimes occur at position 2, e.g. if a verb particle intervenes. The lowest values can be found in the corners of the table, between values $-5/-4$, $-4/-3$ and $5/4$. The further away the position, the less pronounced preferences for certain relations are, and the more often the class will be "no relation". This results in the difference between $-5/5$ being roughly the same as between $-5/-2$. These observations also explain why a numeric treatment of the distance feature does not perform well: it will judge the difference between $-1/1$ to be as big as that between $-5/-3$ and the difference between $-5/5$ much bigger than that between $-5/-2$. Eisner (1996a) achieved a similar effect as the MVDM encoding by manually mapping the distance values into the groups 1, 2, 3–6, and 7–∞.

Table 4.10 shows some value differences for the focus PoS feature. At one level the differences reflect major part-of-speech groups: NNS (plural noun) is similar to NN (singular noun), NNP (proper noun), PRP (pronoun), CD (numeral) but dissimilar to VBx (verbs), JJ (adjective) or RB (adverb). At a finer level NNS is more similar to NN and NNP than to PRP and CD and at a yet finer level it is more similar to NN than to NNP. Thus the MVDM values encode information about a hierarchy of PoS. Eisner (1996a) achieves a similar effect by manually grouping PoS at two levels and using these "short tags" and "tiny tags" for back-off probabilities. Most other parsing work however treats the PoS as unstructured.

The value differences for the focus word (Table 4.11) implicitly also encode this PoS hierarchy, e.g. *it* is similar to *they* and *he* and slightly less similar to *company*. However they also encode semantic knowledge, e.g. the three nouns *year, million* and *company* are pairwise dissimilar whereas the noun *company* and the pronoun *it* are similar. Similarity

[8] In a *wh*-construction, the *wh*-element has a non-local object relation to the verb. However, we are only concerned with local relations here.

	it	said	as	be	who	which	they	year	million	that	company	he
it	0	0.85	0.83	0.85	0.85	0.85	*0.17*	0.82	0.68	0.81	*0.18*	*0.18*
said	0.85	0	0.54	*0.31*	*0.05*	*0.05*	0.83	0.54	0.87	*0.41*	0.71	0.86
as	0.83	0.54	0	0.54	0.54	0.54	0.84	*0.49*	0.80	0.52	0.69	0.86
be	0.85	*0.31*	0.54	0	*0.32*	*0.32*	0.84	0.55	0.88	*0.42*	0.72	0.86
who	0.85	*0.05*	0.54	*0.32*	0	*0.01*	0.84	0.54	0.88	*0.41*	0.71	0.86
which	0.85	*0.05*	0.54	*0.32*	*0.01*	0	0.84	0.54	0.88	*0.41*	0.71	0.86
they	*0.17*	0.83	0.84	0.84	0.84	0.84	0	0.83	0.84	0.81	*0.27*	*0.02*
year	0.82	0.54	*0.49*	0.55	0.54	0.54	0.83	0	0.80	0.52	0.67	0.85
million	0.68	0.87	0.80	0.88	0.88	0.88	0.84	0.80	0	0.83	0.70	0.84
that	0.81	*0.41*	0.52	*0.42*	*0.41*	*0.41*	0.81	0.52	0.83	0	0.67	0.83
company	*0.18*	0.71	0.69	0.72	0.71	0.71	*0.27*	0.67	0.70	0.67	0	*0.30*
he	*0.18*	0.86	0.86	0.86	0.86	0.86	*0.02*	0.85	0.84	0.83	*0.30*	0

Table 4.11: MVDM for the most frequent values of the focus word feature. The lowest numbers (< 0.5) are marked in italics.

	company	companies	share	shares	year	years
company	0	*0.15*	0.58	0.49	0.67	0.62
companies	*0.15*	0	0.57	0.51	0.43	0.54
share	0.58	0.51	0	*0.71*	0.50	0.52
shares	0.49	0.43	*0.71*	0	0.79	0.77
year	0.67	0.57	0.50	0.79	0	*0.45*
years	0.62	0.54	0.52	0.77	*0.45*	0

Table 4.12: MVDM for selected singular and plural nouns of the focus word feature. The numbers for singular/plural pairs are marked in italics.

	Monday		rose		persuaded
0.08	week	0.109	climbed	0.123	urging
0.11	Friday	0.194	slid	0.172	urge
0.126	winter	0.217	fell	0.218	ordered
0.132	night	0.225	jumped	0.22	permit
0.144	month	0.233	surged	0.224	encouraging
0.154	summer	0.251	slipped	0.226	requires
0.18	Mondays	0.276	plunged	0.226	watching
0.194	nights	0.279	dipped	0.231	teach
0.229	afternoon	0.285	decreased	0.233	encourage
0.23	February	0.291	advanced	0.233	marketed
0.235	Tuesday	0.294	dropped	0.236	missed

Table 4.13: The most similar words (according to MVDM) to *Monday* as head of the focus chunk, and to *rose* and *persuaded* as head of the verb chunk.

can also cross major PoS categories, e.g. *who/which* are very similar to *said* because all three hardly ever have any relation to a verb chunk as *who/which* attach to nouns and *said* is mostly the main verb.

Table 4.12 shows that even the singular and plural forms of the same noun are not necessarily similar, e.g. *share* and *shares* are rather different. This is due to constructions like "$ 22.26 a share" which are only possible in the singular. This and the previous example show that MVDM encodes task-specific similarity and can therefore not be replaced by information from general semantic knowledge bases like WordNet (Miller et al., 1990).

As a last example, Table 4.13 shows the most similar words to *Monday, rose* and *persuaded*. The latter two words demonstrate how MVDM forms semantic clusters of verbs on the basis of their behaviour with respect to grammatical relations. The connection between the subcategorization and meaning of verbs was studied extensively in Levin (1993). It should be noted that despite this intuitively correct grouping, MVDM does not help significantly for the verb feature. Maybe its failure to do so is related to the fact that the weight of the verb feature is so low: it is the third least important feature.

4.3.7.3 The number k of Nearest Neighbors

Increasing the number of Nearest Neighbors from 1 to 9 raises performance by nearly 3 points (Table 4.5, p. 91). A rule of thumb known from other applications of memory-based learning says that when using the MVDM metric, it is often useful to increase k (Daelemans et al., 2000, p.12). This is due to the fact that MVDM makes much more fine-grained distinctions so there will typically be much fewer neighbors at each distance than with the overlap metric. Additional experiments showed that a slightly larger k is

sentence	distance	class
Consumer spending in Britain rose 0.1% ... the Central Statistical Office **estimated Friday**.		NP-TMP
"..." **said** Barbara **May**.	0.117434	NP-SBJ
The final proration factor will be **announced Monday**.	0.117542	NP-TMP
The directors' action, taken Oct. 10 but **announced Friday**, ...	0.124292	NP-TMP
Likewise, certificates of deposit on average posted lower yields in the week **ended Tuesday**.	0.129763	NP-TMP-CLR
Among the possible suitors is Italy's Fiat S.p.A., analysts **said** last **week**.	0.134853	NP-TMP
"..." **admits** Mr. **Peters**.	0.134955	NP-SBJ
"..." the Jaguar chairman **asserted yesterday**.	0.138952	NP-TMP
... a stop-gap spending bill that the House Appropriations Committee is to **consider Monday**.	0.140691	NP-TMP
... the Intelsat VI commercial communications satellite is set to be **launched Friday**.	0.140792	NP-TMP

Table 4.14: The sentences belonging to a test instance (first row) and its Nearest Neighbors (other rows). NNs are ordered by increasing distance (column 2). The last column shows the instance's class, i.e. the relation between the head of the verb chunk and the head of the focus chunk (both marked in bold face).

even advantageous with the overlap metric but $k = 9$ is disadvantageous. Thus most of the advantage of the larger k seems to be caused by the use of MVDM.

We also looked at some of the instances that were classified incorrectly with MVDM, $k = 1$ but correctly with MVDM, $k = 9$. Table 4.14 shows the sentences from which a test instance and its Nearest Neighbors are derived. With $k = 1$ the rather exceptional use of "May" as a last name in the first NN leads to the wrong classification (NP-SBJ). At larger values of k the right class (NP-TMP) gets the majority (6 out of 9). With the overlap metric, the instance would also be misclassified, due to the overwhelming majority of constructions with *said* plus inverted subject.

4.3.7.4 Distance weighted class voting

The performance increase through the best distance weighted class voting (Exponential Decay with $\alpha = 30$) is only 0.4 (Table 4.7, p. 92). Class voting only makes sense if $k > 1$. It helps to counteract the disadvantage of large k's (too much "noise" in the NN set) while preserving the advantage (robustness).

This is demonstrated by the case shown in Table 4.15. The first two NNs are much more similar to the test instance than the other seven, and have the correct class (NP-SBJ). With

sentence	distance	class
The **issue**, which is puttable back to the company Nov. 1, 1994, was **priced** at a spread of 70 basis points above the Treasury's five-year note.		NP-SBJ
The noncallable **issue**, which is puttable back to the company Nov. 1, 1994, was **priced** at a spread of 62.5 basis points above the Treasury's five-year note.	0.000000	NP-SBJ
The noncallable **issue**, which has a one-time put Oct. 15, 1999, was **priced** at a spread of 66 basis points above the Treasury's 10-year note.	0.003088	NP-SBJ
A new **issue**, Exabyte, surged 2 1/8 from its initial offering price to **close** at 12 1/8.	0.015745	–
The refunding **issue**, which had been in the wings for two months, **was** one of the chief offerings overhanging the market ...	0.028618	NP-SBJ
The Bravo Zulu **award**, the Navy accolade for a "job well done," is **bestowed** on Federal's workers who surpass the call of duty.	0.032538	NP-SBJ
The strong **dollar**, which reduces the value of overseas earnings and revenue when they are **translated** into dollars, is expected to ...	0.034045	–
The **broker**, Thomas Beairsto of Merrill Lynch's Morristown, N.J., office, refuses to discuss the matter with a reporter, **referring** inquiries to Merrill Lynch officials in New York.	0.038348	–
The **company**, which reported that its loss for the fiscal quarter **ended** Aug. 26 widened from a year earlier, cut its semiannual dividend ...	0.039516	–
Per capita **income**, a widely used measure of a nation's economic health, hit a record in 1988, **rising** 1.7% after inflation adjustment to $13,120.	0.039623	–

Table 4.15: The sentences belonging to a test instance (first row) and its Nearest Neighbors (other rows). NNs are ordered by increasing distance (column 2). The last column shows the instance's class, i.e. the relation between the head of the verb chunk and the head of the focus chunk (both marked in bold face).

$k = 1$ the test instance is classified correctly by the first Nearest Neighbor alone. With $k = 9$ and majority voting however, the correct class gets outvoted by the "no relation" class with 5 against 4. With $k = 9$ and Exponential Decay voting, the vote of the first two NNs gets much more influence than that of the other seven, so NP-SBJ "wins" with 2.71208 against 1.91036.

4.3.7.5 Algorithms

As Table 4.8 (p. 93) shows, IGTree scores worse than IB1-IG (72.54 vs. 80.09). This can partially be explained by the fact that IGTree does not allow for most of the options that improved performance of IB1, like MVDM, larger k and distance weighted class voting. But even without these options, IB1 is better (F_β=74.40, cf. Table 4.3, p. 90). IGTree usually performs worse than IB1-IG if there are several features with similar weights, or if the features with the highest weight regularly have value combinations in test instances that did not occur in training. This is always the case if a high-valued feature can have unseen values, which is typically the case with word-valued features.

The same reasoning also explains why TRIBL with larger values of q performs worse than IB1-IG. It is however interesting to note that performance stays the same for $q = 1$ or 2, although this makes the MVDM option unavailable for the first one or two features (focus chunk and preposition) and Table 4.6 showed that MVDM is useful especially for the focus chunk. Apparently this handicap of TRIBL is outweighted by the advantage of a forced match on the most important features.

4.4 Summary

In this chapter we described three Memory-Based Learning algorithms, their parameters and details of their implementation. We then tested various settings on our relation finding data set: feature weights, global and feature-specific metric, number of Nearest Neighbors, distance weighted class voting schemes and algorithms. The setting that yields the best F_β performance (80.09) is IB1-IG with Gain Ratio weighting, the MVDM metric, $k = 9$ and Exponential Decay weighted class voting ($\alpha = 30$). This is an important increase over the default settings (overlap, $k = 1$, majority voting: 74.40). The downside of this improvement is much slower classification: 4:48 vs. 1:06 hours.

The setting performs well because Gain Ratio recognizes the importance of the focus chunk and the distance feature, MVDM encodes a non-linear scale of the distance and task-specific syntactic and semantic hierarchies of words and PoS, a larger k increases robustness for MVDM, distance weighted class voting lessens the effect of noise in the NN set, and IB1-IG allows to exploit all these options for all features. The new setting will be used in all following experiments.

Chapter 5

Feature representation improvement

In the previous chapter we optimized the parameters of the learning algorithm. This increased F_β from 74.40 to 80.09. We will use the optimal parameter setting in the present chapter where we try to determine what information is important for finding grammatical relations. We do this by varying the features of our data and testing whether performance changes. As our preliminary features (cf. Section 3.1.3) were a somewhat arbitrary selection out of all the information present in the history, there are many possible ways to improve on this.

There is a conceptual difference between the information conveyed by (a set of) features and the actual feature representation, due to the bias of the learner. To take a simple example: suppose we want to convey information about the type of a chunk and the possible values are NP, VP, ADVP and ADJP. We could choose to represent this information with one feature having four possible values, or with four binary features of which exactly one must be "true" at a time or with two binary features (cf. Section 2.1.1) or even with two or three ternary features, which would be a redundant representation.

In practice however, it is not possible to clearly separate the value of the information from the merit of the representation. The information value poses an upper bound on the merit of any of its representations, but the reverse is not true: if performance does not improve significantly when adding a certain feature this does not prove that this *information* is useless, it only teaches us that *either* the information is useless *or* our algorithm cannot exploit it properly *or* the effect is too small to be significant. For any practical purpose, the three amount to the same. The solution to try all possible representations of a given piece of information is infeasible. However, we will sometimes try several representations of (nearly) the same information.

Despite this caveat the various ways of changing the features that we will test in this chapter can be associated with either a change of information or a change of representation. Using fewer features (Section 5.1) normally entails using less information unless a feature is completely redundant. Combining several features into one (Section 5.2) or conversely splitting one feature into several (Section 5.3) is a change of representation. Adding features

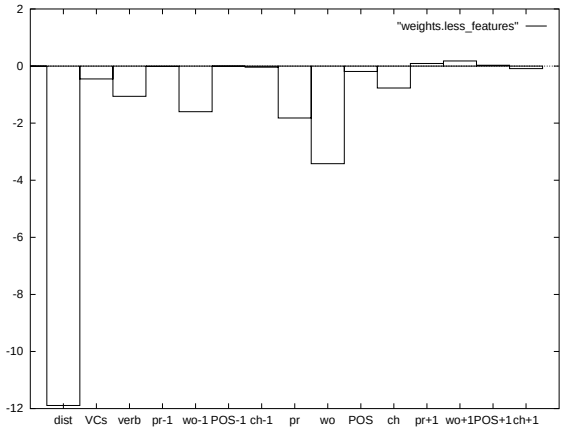

Figure 5.1: Effect on the F_β value when ignoring individual features. The F_β of 80.09 with all features corresponds to the zero line here. The differences up to feature "VCs" are all statistically significant with at least $p < 0.01$. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$)

(Section 5.4) entails adding information unless the features are redundant. Adding different (sets of) features that encode similar information in different ways (cf. page 113) combines changing information with changing representation. Adding values to an existing feature (Section 5.5) is like adding a new feature and simultaneously combining it with an old one. Similarly deleting values (Section 5.7) is like first splitting one feature into two and then ignoring one of them. Although the main focus of this chapter is on testing what information is important for finding grammatical relations, we will continue to note how memory and run time requirements change when changing features.

As in the previous chapter, we will describe the experiments and results in the first sections, and discuss results and their implications in the last section (5.9).

5.1 Fewer features

In this section, we will start with the easiest possibility of feature change: deleting features. We let the algorithm ignore one feature at a time and record the decrease in performance. Table 5.1 and Figure 5.1 show the results. At first sight, it seems that many features are useless and can safely be ignored. However, this measure suffers from the opposite fault as the feature weightings discussed in Section 4.3.7.1. The latter computed all weights *independently*, whereas the former measures "uselessness" *in the presence of all other features*. So if performance does not decrease significantly (or even increases) when ignoring either of the four features of the right context, this does not mean that we can safely ignore all of them at the same time, as shown in the 19th row of Table 5.1: the F_β value then drops from 80.09 to 78.78 which is significant ($p < 0.001$, $t = 11.141$). The same holds for the prepositional, PoS and chunk type features of the left context ($p < 0.001$, $t = 3.993$). We

ignored feature(s)		feature #	MB	h:m	precision	recall	F_β
	none		107	4:48	85.41	75.39	*80.09*
	dist.	3	97	5:34	71.11	65.52	68.20
	VCs	4	106	5:44	85.14	74.81	79.64
	verb	5	91	2:26	84.50	74.22	79.03
context −1	prep.	6	103	4:42	85.40	75.39	**80.08**
	word	7	94	2:35	83.47	74.08	78.49
	PoS	8	101	5:23	85.45	75.36	**80.09**
	chunk	9	105	5:56	85.49	75.26	**80.05**
focus	prep.	10	103	4:58	84.27	73.07	78.27
	word	11	90	2:20	82.40	71.69	76.67
	PoS	12	101	5:07	85.51	74.97	79.90
	chunk	13	106	7:07	84.79	74.52	79.32
context +1	prep.	14	103	4:39	85.50	75.49	**80.18**
	word	15	92	3:09	85.35	75.75	**80.27**
	PoS	16	100	4:48	85.28	75.55	**80.12**
	chunk	17	104	5:23	85.26	75.35	**80.00**
three of context −1		6,8,9	95	7:14	85.17	74.63	79.55
all of context +1		14–17	78	3:07	83.77	74.35	78.78
two of context −1		6,8	98	5:40	85.45	75.35	**80.08**
three of context +1		14–16	82	3:04	85.08	75.76	**80.15**
five of context −1/+1		6,8,14–16	77	3:27	85.11	75.79	**80.18**

Table 5.1: Performance when ignoring features (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$)

can however ignore two features of the left context, or three of the right, or even all five together. The latter result shows that information in the two context elements is mutually independent.

Ignoring these five features decreases the memory requirement from 107 to 77 MB and run time from 4:48 to 3:27 hours while increasing performance (insignificantly, $t = 0.735$) from 80.09 to 80.18. We will therefore use the reduced feature set in subsequent experiments. We could also have ignored the focus PoS with a decrease in memory requirement and only an insignificant performance loss ($t = 1.162$). However run time would increase. As no application dictates a preference, we rather arbitrarily choose to keep the focus PoS.

5.2 Combining features

We saw in the previous section that we can ignore some features without harming performance. This effect is probably due to redundancy. If features are only partially redundant, it might be a good idea to combine them into one feature by concatenating their values.

feature representation	MB	h:m	precision	recall	F_β
separate prep. and chunk type feature	77	3:27	85.11	75.79	**80.18**
combined prep. and chunk type feature	77	3:54	85.11	75.78	80.17

Table 5.2: Combining the prepositional and the chunk type feature of the focus (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 features)

feature representation	MB	h:m	precision	recall	F_β
combined direction and absolute distance	77	3:27	85.11	75.79	*80.18*
separate direction and absolute distance	77	2:46	85.16	75.94	**80.29**

Table 5.3: Splitting the distance feature speeds up classification. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 features)

That way, the algorithm can better compute the joint weight, and the number of features is reduced. One linguistically plausible combination consists of the prepositional and the chunk type feature (the syntactic head and the syntactic type, respectively). So instead of four feature-value pairs *(as, director, NN, PNP)* to represent a chunk like *"as a nonexecutive director"*, we now have only three *(director, NN, PNP-as)* but these still carry the same information. Table 5.2 shows the results with the new representation. Performance does not change significantly (from 80.18 to 80.17) and the separate representation is faster. We will therefore keep the two features separate.

5.3 Splitting features

Just as we can combine features, we can also split features. A good candidate for splitting is the distance feature. Collins (1996) already made a distinction between direction (modifier to the left or right of the head) and adjacency (modifier directly next to the head) and Eisner (1996a) between direction and distance. So instead of having one feature value, say "−7", meaning the focus is seven chunks to the left of the verb chunk, we now have two features with values "−" and "7" respectively. Table 5.3 shows the results: we notice a slight insignificant performance increase (from 80.18 to 80.29, $t = 0.719$) but a clear increase in speed. We will therefore adopt this new representation in subsequent experiments.

A much more radical way to split features is the use of binary features. This means that instead of one symbolic feature with i values we have i features (with two values each). This representation is very redundant (as only exactly one of the binary features can be "on" in any given instance). Its effect is that the algorithm now in fact computes the importance (i.e. weight) of a (former) feature value but the generalization expressed by (former) feature weights is lost. A similar approach is taken by the SNoW and Maximum

feature representation	MB	h:m	precision	recall	F_β
Symbolic multi-valued features	279	20:51	80.01	73.26	76.49
Binary features	556	85:57	78.92	74.60	**76.70**

Table 5.4: Using symbolic versus binary features with Fambl. (1/0 restriction on intervening verb chunks, IB1-IG, overlap metric, $k = 1$, majority voting, ignoring 5 features, split distance)

Entropy algorithms (cf. also Sections 2.4.1.1 and 2.4.2.6). A disadvantage of this approach is that especially for the word-valued features, many feature values appear very infrequently. Therefore their importance cannot be estimated reliably (the well-known sparse data problem).

TiMBL does not have any built-in option for converting symbolic to binary features. Doing the conversion separately would mean presenting the algorithm with instances having about 40,000 features. As this is not feasible in this implementation we chose to use another MBL implementation for this experiment: Fambl (Van den Bosch, 1999b; Van den Bosch, 1999a), which has an option for unpacking multi-valued features into binary ones.[1] As it is unclear whether the advantage of the MVDM metric and the larger value of k carries over to a binary representation, we chose to use the default settings (overlap metric, $k = 1$, majority voting). Table 5.4 shows that the use of binary features increases performance significantly from 76.49 to 76.70 ($p < 0.05, t = 1.829$) but also that it needs much more memory and run time.[2] For practical reasons we will not use binary features in the remaining experiments. More research needs to be done in this direction.

5.4 More features

The feature changes treated in the previous sections were more successful in achieving the same performance with less memory or run time than in increasing performance as none of them adds information. In this section we will add information in the form of new features. Figure 3.18 (p. 71) showed the initial feature choice for an example instance. All the pieces of information that are not underlined represent potential new features. In addition we can also use information that the main part of the figure contains only implicitly like the distance and the intervening verb chunks. Let us have a more systematic look at the possible information sources now. Example values refer to our example sentence. Underlined values are used already.

[1] With the "-b" option Fambl behaves identical to default TiMBL. Setting the "-a" option triggers the use of binary instead of symbolic multi-valued features.

[2] The performance increase from 74.40 (Timbl with IB1-IG, overlap metric, $k = 1$, majority voting, initial features, cf. Table 4.3) to 76.49 (Fambl with IB1-IG, overlap metric, $k = 1$, majority voting, ignoring 5 features, split distance) must be caused by the improved feature representation, which seems to have much more effect with the default parameter setting than with the best setting, where it only caused improvement from 80.09 to 80.29 (cf. Tables 5.1 and 5.3).

- The focus chunk
  - The type of chunk: *(NP)*
  - The (semantic) head: *(29)*
  - Its part-of-speech: *(CD)*
  - The syntactic head of a PNP: *(–)*
  - The other words/PoS of the focus chunk besides the head (henceforth called the *rest of the focus chunk*): *Nov./NNP*
- The verb chunk
  - The type of chunk is used implicitly in instance construction as instances correspond to pairs of verb and other chunks.
  - The (semantic) head: *(join)*
  - Its part-of-speech: *(VB)*
  - The other words/PoS, besides the head, of the verb chunk (the *rest of the verb chunk*): *will/MD*
- All the other elements in the sentence. As the verb and the focus chunk are the two fixed points in the sentence, we can divide the other elements with respect to these two chunks.
  - First there is one part comprising all the elements in front of the verb chunk (respectively in front of the focus chunk if it is first) (*front material*): *[NP Pierre/NNP Vinken/NNP] ,/, [NP 61/CD years/NNS] [ADJP old/JJ] ,/,*
  - Then there is the part of all the elements in between the verb and the focus chunk (*intervening material*): *[NP the/DT board/NN]* {*PNP [PP as/IN] [NP a/DT nonexecutive/JJ director/NN]*}
  - And finally there is the part following the focus chunk (resp. the verb chunk if it is last) (*back material*): *./.*

  As for the verb and focus chunk, we can use the headwords, their PoS, the chunk types or the non-headwords/PoS of all the elements in the front, intervening and back material.

The above list shows possible new information sources. However, when adding new features we also have to decide upon their representation. The "rest of the verb/focus chunk" as well as the "front/intervening/back material" are sequences of elements. There is no principled upper length to these sequences, although on average, the "rest of the verb/focus chunk" tends to be shorter than the "front/intervening/back material". As with any application of a propositional learner to language data, we have to find a mapping from potentially unbounded sequences to a fixed number of propositional features. There are at least two ways to do this:

- Windowing: We define a window of a certain width around a fixed point in the sequence and use only information from elements that fall into this window (Sejnowski and Rosenberg, 1987). This approach is widely used when applying propositional learners to language data. The fixed points in the verb and the focus chunk are the start and the head of the chunk. The fixed points in the complete sentence are the verb and the focus chunk (which correspond to the end of the front material, the start and the end of the intervening material, and the start of the back material). Up to now, we have used a window of three chunks, centered on the focus chunk.

- Global features: Up to now, we used two global features of the intervening material: its length (in the distance feature) and the number of verb chunks in it. These are properties of the sequence as a whole, not only of some part close to some fixed point. In general, there are three types of features:

  - Symbolic: As an example, take the intervening material of our example sentence. We might have a symbolic feature with the value "the_board_as_a_non-executive_director" or "DT_NN_IN_DT_JJ_NN" or "NP_PNP" or any variation of these.
  - Numeric: The distance feature is an example of a numeric feature of the intervening material. It counts all the elements in the sequence. Alternatively, we can count specified elements only, as we did for the "intervening verb chunks" feature.
  - Binary: Every numeric feature can be mapped to a binary feature. Collins (1996) used a binary feature indicating whether there was a verb between the focus chunk and the head.

Given the possibilities described above, there is a nearly infinite number of possible new features yet to be explored. For each of the five sequences, we could try the windowing approach with all different window sizes, and the global feature approach with all different kinds of feature types and elements represented/counted. As this is infeasible, we started with the more straightforward options and explore other possibilities only if the first experiment looks promising. The following new features will be tested in this chapter:

- The PoS of the verb (Section 5.4.1).

- Window features around the focus and the verb chunk (Section 5.4.2). Our preliminary representation corresponds to a 1/1 context window around the focus chunk. We will gradually increase context window size. The context window around the verb chunk was 0/0 up to now. We will also increase its size.

- Global features of the front, back and intervening material (Section 5.4.3). For the intervening material different representations will be tested.

- Global features of the rest of the focus and verb chunk (Section 5.4.4).

New feature	MB	h:m	precision	recall	F_β
none	77	2:46	85.16	75.94	*80.29*
PoS of verb	80	2:57	85.20	76.19	80.44

Table 5.5: The part-of-speech of the verb as extra feature (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

Focus context features	MB	h:m	precision	recall	F_β
w−1, ch−1, ch+1	77	2:46	85.16	75.94	*80.29*
w−1, ch−1, ch+1, ch+2	81	3:10	85.23	76.00	80.35
w−1, ch−1, ch+1, w+2	93	4:34	85.33	75.54	80.14
ch−2, w−1, ch−1, ch+1	81	2:56	85.20	76.30	80.51
w−2, w−1, ch−1, ch+1	92	4:41	85.79	76.29	**80.76**
w−2, ch−2, w−1, ch−1, ch+1	96	5:00	85.85	76.51	**80.91**
w−3, w−2, w−1, ch−1, ch+1	107	6:49	85.90	76.16	**80.74**

Table 5.6: Using more context of the focus chunk. "w" is the word feature, "ch" means chunk type feature. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

For the sake of efficiency and to check for feature redundancy the new features will all be tested separately and only combined later (Section 5.6).

5.4.1 PoS of the verb

As Table 5.5 shows, addition of the PoS of the verb increases performance only insignificantly ($t = 1.440$). As it also increases memory and run time, it does not seem to be a useful feature. However we will return to this point in Section 5.9.3.3.

5.4.2 Context window around focus and verb chunk

Up to now we have been using information from one element to the left of the focus chunk and one element to its right (in addition to the focus chunk itself). Table 5.1 showed that using *less* focus context is worse. We will now test whether *more* context is better. As we also saw that only the word and the chunk type are useful for the left context (−1) and only the chunk type for the right context (+1), we will concentrate on these features and ignore the prepositional and PoS features. Table 5.6 shows the results. Neither the chunk type nor the word of the second element to the right has a significant effect ($t = 0.885$ and $t = 1.165$). We therefore did not explore further elements in this direction. For the left context, the chunk type of the second element improves performance significantly

Verb context features	MB	h:m	precision	recall	F_β
	77	2:46	85.16	75.94	*80.29*
chunk+1	81	2:57	85.11	76.04	80.32
word+1	91	4:57	85.36	76.04	80.43
chunk−1	81	3:02	85.28	76.22	80.49
word−1	92	5:41	85.51	76.37	**80.68**
word−1, chunk−1	94	5:36	85.59	76.51	**80.79**
word−2, word−1	104	8:47	85.81	76.40	**80.83**

Table 5.7: Using more context of the verb chunk (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

($p < 0.05, t = 2.090$) from 80.29 to 80.51. However adding the headword of this element increases performance even more, to 80.76. And once the word is added, the chunk type yields only an insignificant additional increase (to 80.91, $t = 1.278$) but decreases memory and time efficiency. Likewise adding the headword of the third element to the left does not help performance (80.74). In summary the headword of the second element to the left is the only useful feature we found in the window around the focus chunk. Unfortunately its addition also increases memory and run time requirements considerably.

We performed similar experiments for the window around the verb chunk. The size in the preliminary representation is zero on both sides. Results mirror those for the focus window: the context to the right does not improve performance significantly ($t = 0.262$ for chunk, $t = 1.302$ for word). The chunk type of the context to the left adds very little performance ($t = 0.885$) once the headword of this element is known, whereas the headword raises performance significantly from 80.29 to 80.68 ($p < 0.01, t = 3.446$). The headword of the second element to the left still increases performance (to 80.83) but only insignificantly ($t = 1.192$). So the only useful feature in the verb context window is the headword feature of the element to the left. Again its addition increases memory and run time requirements considerably.

5.4.3 Global features of the front, back, and intervening material

For the global features of the front, back and intervening material we first test a symbolic representation, first using only the chunk type information of each element (or its PoS if the element is not a chunk but e.g. punctuation). For our example sentence in Figure 3.18, the front material would thus be represented as *"NP , NP ADJP ,"*. To prevent excessively long feature values, we restricted the number of elements represented per feature arbitrarily to a maximum of 30. Table 5.8 shows results for the three new features. Only the intervening feature improves performance but this improvement is considerable: from 80.29 to 81.46. All three features take much time and memory but the intervening least so. This is clearly

Global feature	representation	av. len.	# values	MB	h:m	precision	recall	F_β
none				77	2:46	85.16	75.94	*80.29*
front	chunks, symb.	5.4	58,122	108	7:37	84.91	75.82	80.11
intervening	chunks, symb.	2.6	45,366	102	5:27	86.02	77.36	**81.46**
back	chunks, symb.	6.5	64,516	128	9:10	84.70	74.82	79.45

Table 5.8: Global features of the other elements in the sentence besides the verb and the focus chunk. The representation is symbolic, consisting of a sequence of chunk types (or PoS for non-chunks), covering a maximum of 30 elements. The third column shows the average length of the sequence. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

Global feature	representation	av. len.	# values	MB	h:m	precision	recall	F_β
none				77	2:46	85.16	75.94	*80.29*
intervening	chunks, symb.	2.6	45,366	102	5:27	86.02	77.36	**81.46**
intervening	PoS, symb.	4.9	129,056	149	10:04	85.70	75.67	80.37
intervening	words, symb.	4.9	193,454	226	14:48	85.05	75.31	79.89

Table 5.9: Global features of the intervening material. The representation is symbolic, either consisting of a sequence of chunk types (or PoS for non-chunks), covering a maximum of 30 elements, or of a sequence of PoS respectively words, covering a maximum of 15 elements. Using the chunk type information of the elements performs best. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

related to the different average lengths of the sequences, which result in different numbers of feature values (columns 3 and 4). Based on the results of this first experiment we decided to continue experimentation with the intervening feature only.

Instead of using the chunk types of the sequence elements, we tried using the sequence of PoS and of (head and non-head) words. In this case the number of elements covered was restricted to 15. Table 5.9 shows that the PoS representation increases performance only insignificantly (from 80.29 to 80.37, $t = 0.719$), whereas the word representation decreases performance. This might be related to the much larger number of feature values. We will continue experiments with the chunk-based global feature.

We convert the symbolic sequence of chunk types (or PoS) to a set of numeric features by counting occurrences of all symbols that occur more than 1000 times in training material. These are in decreasing frequency order: NP, PNP, comma, VP, ADVP, CC, SBAR, ADJP, ", :, ", PRT, RB,), and (.[3] Thus a sequence like *", NP PNP PNP ,"* would be converted into NP: 1, PNP: 2, comma: 2, VP: 0, ADVP: 0, etc. Conversion of numeric features to

[3]Conveniently this set is the same for all ten folds. See Appendix A.1 for a list of PoS tags.

Global features	representation	MB	h:m	precision	recall	F_β
none		77	2:46	85.16	75.94	*80.29*
intervening	chunks, symbolic	102	5:27	86.02	77.36	**81.46**
intervening	chunks, numeric	93	2:12	85.50	77.76	**81.45**
intervening	chunks, binary	86	1:46	85.32	77.33	81.13

Table 5.10: Global features of the intervening material. The representation is either symbolic, consisting of a sequence of chunk types (or PoS for non-chunks), or numeric (15 features), or binary (15 features). The maximum number of elements covered is always 30. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

Global features	representation	MB	h:m	precision	recall	F_β
none		77	2:46	85.16	75.94	*80.29*
intervening	chunks, numeric, 15 features	93	2:12	85.50	77.76	**81.45**
intervening	chunks, numeric, 3 features	80	2:15	85.46	77.32	81.18
intervening	chunks, numeric, 7 features	84	2:05	85.64	77.77	**81.52**

Table 5.11: Numeric global features of the intervening material. Using subsets of the original 15 features. The maximum number of elements covered is 30. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

binary ones just means replacing all values higher than 1 by 1. As Table 5.10 shows, the numeric representation performs as well as the symbolic one while requiring less memory and much less run time. Interestingly classification is even faster with the new numeric features than without them. The binary representation performs significantly worse (81.13, $p < 0.01, t = 3.519$) than the numeric one but still better than without any new features ($p < 0.001, t = 8.830$) and even faster, so it might be a worthwhile alternative for fast applications. We will continue experiments with the numeric representation.

To find out which numeric features are most important we added each of them separately and then tried combinations of them. The "3 features" in Table 5.11 are the ones that alone had increased performance significantly (comma, CC, SBAR), the "7 features" are the ones that alone had increased performance at all (addition of NP, ", :, "). Although addition of the "3 features" increases performance to 81.18, the "7 features" increase performance significantly more (to 81.52, $p < 0.01, t = 3.420$). They even perform (insignificantly, $t = 0.707$) better than the original 15 numeric features, and they need less memory and run time, so we will use these 7 features to represent the intervening material.

"Rest of" feature	representation	av. len.	# values	MB	h:m	precision	recall	F_β
none				77	2:46	85.16	75.94	*80.29*
focus	words, symb.	1.11	43,759	100	6:18	84.96	75.64	80.03
focus	PoS, symb.	1.11	5,750	83	3:43	85.41	76.25	**80.57**
verb	words, symb.	0.55	2,348	83	4:29	85.39	75.97	80.40
verb	PoS, symb.	0.55	383	82	3:03	85.29	76.07	80.42

Table 5.12: Adding global features of the rest of the focus and verb chunk, in symbolic representation. Sequences consist either of PoS or of words. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

5.4.4 Global features of the rest of the focus and verb chunk

The example of the intervening feature showed that the symbolic representation of a global feature makes a useful first experiment. We therefore also tried this option with the "rest of the focus chunk" and "rest of the verb chunk" features. Sequences consist either of PoS or of words. Table 5.12 shows that the rest of the focus chunk represented as words decreases performance (from 80.29 to 80.03) whereas the PoS representation increases performance significantly ($p < 0.05, t = 2.492$). The rest of the verb chunk yields an insignificant improvement in both representations ($t = 1.064$ for words, $t = 1.183$ for PoS). We will thus add the rest of the focus chunk in PoS representation to the list of useful features.

5.5 More feature values

As we already explained, simultaneously adding a new feature and combining it with an existing one amounts to adding new values to a feature. In this paragraph we will describe two such approaches, concerning prepositions and VPs respectively.

Recall that during chunking, PP chunks are found. These consist of just one word in most cases. However there are also some frequent multi-word prepositions like "because of", "instead of", "such as", "rather than", "due to". During PNP-finding a PP and one or more NP chunks are combined into one PNP chunk. A focus PNP chunk is represented by four features in the current representation: the preposition, the headword, the PoS and the chunk type. The headword is the last word of the NP chunk. Likewise the prepositional feature value is the last word of the PP chunk. This works fine if the PP chunk just consists of one word, if the extra words are just modifiers to the preposition (like "shortly before", "particularly in" "only in", "not including") or if the preposition is coordinated (e.g. "to and from", which would have the head "from"). For the true multi-word prepositions however, information is lost. The learner is unable to make the difference between e.g. "because of him", "instead of him" and "of him". To solve this problem we decided to use

New feature (values)	MB	h:m	precision	recall	F_β
none	77	2:46	85.16	75.94	*80.29*
multi-word prepositions	77	2:53	85.18	76.10	80.38
VP type for focus	77	2:37	85.30	76.31	**80.55**
VP type for focus and verb chunk	82	2:57	85.68	76.87	**81.04**

Table 5.13: Difference in performance with and without extra values in the prepositional feature for multi-word prepositions and types of VPs (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

a concatenation of all the words in the PP chunk as the value of the prepositional feature instead of just the last word. In the above-mentioned cases of modifiers or coordinations, this approach leads to a loss of generalizations (as "shortly before" is now different from "before"). However as these cases are less frequent than the true multi-word prepositions, the overall effect is positive (from 80.29 to 80.38, cf. Table 5.13) but not significantly so ($t = 0.902$).

The prepositional feature contains the syntactic head of a PNP chunk whereas the head feature in fact contains the semantic head. This distinction between syntactic and semantic heads is also applicable to VPs. Consider a VP like "will visit" or "to visit". The semantic head clearly is "visit" as the semantic predicate-argument structure of the clause containing this VP would be something like "visit(X,Y)". The syntactic head however is "will" respectively "to". With the former, the VP can be part of a finite clause whereas the latter makes it an infinitive. We therefore introduced eight other values into the prepositional feature. These values appear only with VPs and represent the PoS of the syntactic head (MD, TO, VB, VBD, VBG, VBN, VBP, VBZ). We call this information the "VP type". Note that in our current representation only the focus chunk has a prepositional feature, thus the new VP type information is added only to the focus and not to the verb chunk. This significantly increases F_β from 80.29 to 80.55 ($p < 0.05$, $t = 2.441$). In a second experiment we also added a feature containing the VP type information for the verb chunk. This significantly increases F_β further to 81.04 ($p < 0.001$, $t = 4.286$).

5.6 Combinations of new features

In the previous sections, we saw that several new features or feature values improved performance when added separately. However not all of the new features are independent of each other, e.g. the two left context words and the intervening material partially convey the same information, as do the rest of the focus chunk and the VP type of the focus chunk. For this reason, we first checked combinations of features which we suspect to interact. Table 5.14 shows that performance is lower when adding the VP type of the focus chunk *and* the rest of the focus chunk (80.51) than when just adding one of them

New feature (values)	MB	h:m	precision	recall	F_β
none	77	2:46	85.16	75.94	*80.29*
VP type for focus	77	2:37	85.30	76.31	80.55
rest of focus chunk (PoS)	83	3:43	85.41	76.25	80.57
VP type for focus & rest of focus chunk (PoS)	84	3:11	85.28	76.25	80.51
focus context word−2	92	4:41	85.79	76.29	80.76
verb context word−1	92	5:41	85.51	76.37	80.68
verb context word−1 & focus context word−2	106	8:49	86.16	76.70	81.16
intervening (7 numeric)	84	2:05	85.64	77.77	81.52
verb context word−1 & focus context word−2 & intervening (7 numeric)	113	5:30	86.48	78.38	82.23
VP type (for focus & verb)	82	2:57	85.68	76.87	81.04
verb context word−1 & focus context word−2 & intervening (7 numeric) & VP type (for focus & verb)	119	5:58	86.87	79.24	**82.88**

Table 5.14: Combinations of new features and feature values (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

(80.55 and 80.57). As the VP type option reduces run time and keeps memory constant whereas the rest of the focus chunk increases both, we will only use the former in future experiments.

The two left context words and the intervening features together increase performance to 82.23, which is more than with any of them separately. The last row of Table 5.14 shows performance with all useful new features/values together. In total, performance increases from 80.29 to 82.88. However memory and run time also increase by 55% and 115%, respectively.

Figure 5.2 shows which information from the history is new in this representation. Table 5.15 gives the instances from Table 3.1 in the new representation.

5.7 Fewer feature values

The new representation developed in the previous section has a superior performance but also higher memory and run time requirements. This is mainly caused by the two new word-valued context features. When using the MVDM option, the learning algorithm has to compute a value difference for each pair of values for each feature. If the number of values of a feature is reasonably low (e.g. for the chunk type features), this matrix is computed once and stored. If on the other hand a feature has many values, only the distribution of classes with each value is stored and the actual MVDM value is computed on the fly,

features			6 instances					
	focus #	1	1	4	5	10	11	16
	verb #	2	8	8	8	8	8	8
	dir.	3	−	−	−	+	+	+
	dist.	4	5	3	2	1	2	3
	VCs	5	0	0	0	0	0	0
verb cont. −1	word	6	,	,	,	,	,	,
verb chunk	VP type	7	MD	MD	MD	MD	MD	MD
	verb	8	join	join	join	join	join	join
focus cont. −2	word	9	–	Vinken	,	,	join	board
focus cont. −1	word	10	–	,	years	join	board	director
	chunk	11	–	–	NP	VP	NP	PNP
focus chunk	prep./VP type	12	–	–	–	–	as	–
	word	13	Vinken	years	old	board	director	29
	PoS	14	NNP	NNS	JJ	NN	NN	CD
	chunk	15	NP	NP	ADJP	NP	PNP	NP
focus cont. +1	chunk	16	–	ADJP	–	PNP	NP	-
intervening, numeric	commas	17	2	1	1	0	0	0
	CCs	18	0	0	0	0	0	0
	SBARs	19	0	0	0	0	0	0
	NCs	20	1	0	0	0	1	1
	”	21	0	0	0	0	0	0
	colons	22	0	0	0	0	0	0
	“	23	0	0	0	0	0	0
	class		NP-SBJ	–	–	NP-OBJ	PP-CLR	NP-TMP

Table 5.15: The instances for the first sentence of the Wall Street Journal Corpus (cf. Figure 5.2) in the new feature representation.

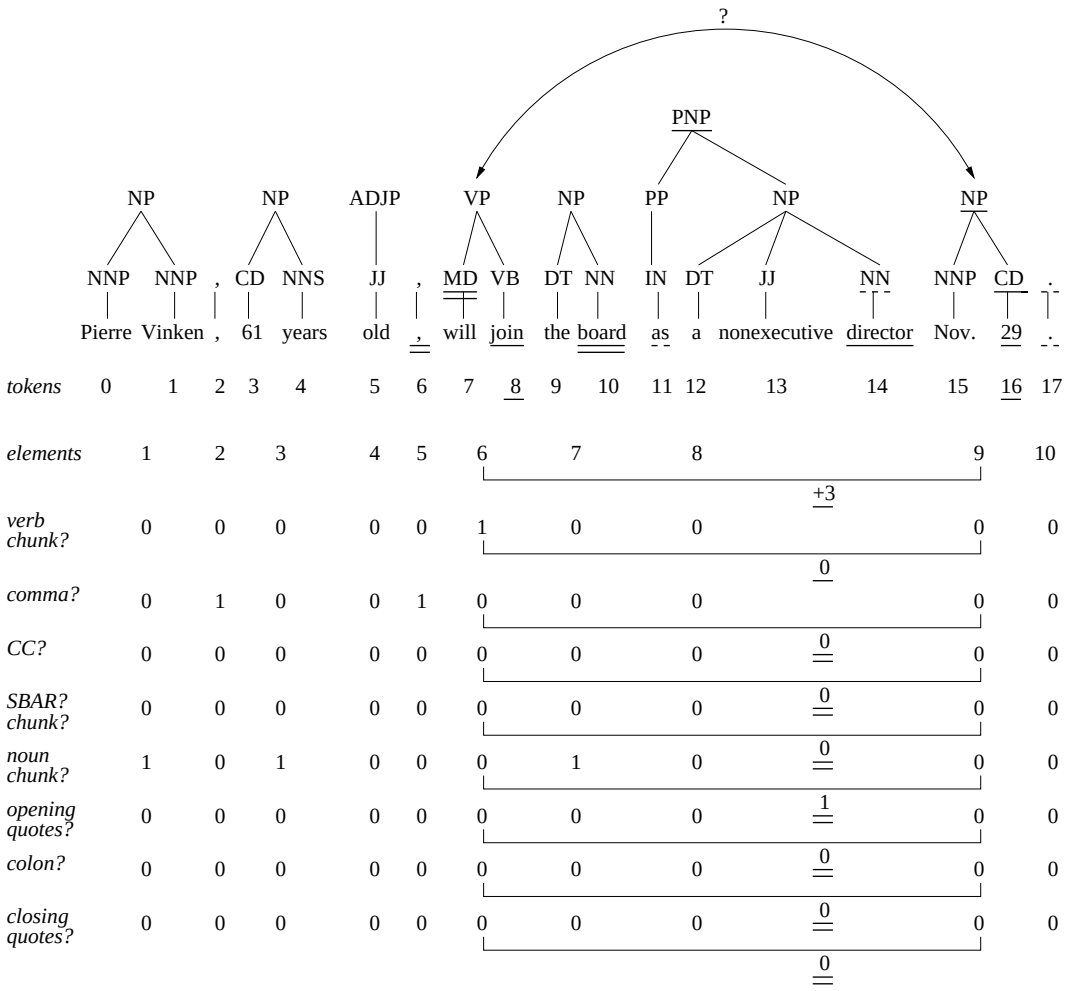

Figure 5.2: Our example sentence from Figure 3.18. The single underlined parts are used as feature values of the instance that corresponds to the decision about the relation (marked by the question mark) between the verb and the "Nov. 29" chunk in the new feature representation. The doubly underlined parts are new. The dashed underlined parts were used in the preliminary representation but have been discarded in the new one.

which takes time. In addition, storing the distribution takes up much memory, especially for the many-valued word features. As words have a Zipfian distribution, many values are very rare. We know that for rare values MVDM value computation is based on little evidence and is therefore not reliable. In order to save run time and memory and maybe even exclude unreliable values, we replace all feature values that appear at most 1, 5 or 10 times in the training material by the new value HAPAX. All values in test material that do not occur as such in training material (i.e. they have been replaced by HAPAX or did not occur at all) are also replaced by HAPAX. To give an idea of the scope of the change: with threshold 5, about 35% of the test instances contain at least one HAPAX value. In

HAPAX threshold	MB	h:m	precision	recall	F_β
0	119	6:49	86.87	79.24	*82.88*
1	116	5:18	86.64	79.65	83.00
5	108	3:34	86.45	80.01	**83.10**
10	104	3:14	86.12	79.99	82.94

Table 5.16: Replacing infrequent feature values by the special symbol HAPAX (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance; new features: verb context word-1, focus context word-2, 7 intervening numeric, VP types)

Class values	MB	h:m	precision	recall	F_β
all	108	3:34	86.45	80.01	*83.10*
frequency > 5	108	3:27	86.53	79.94	83.10

Table 5.17: Replacing infrequent class values by the "no relation" class (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance; new features: verb context word-1, focus context word-2, 7 intervening numeric, VP types; HAPAX threshold=5)

one test fold picked at random, HAPAX values occur in 11 of the 21 non-administrative features.

Table 5.16 shows the results of HAPAX replacement: all three thresholds increase performance and decrease memory and run time requirements.[4] Performance improvement with threshold 5 is even statistically significant (from 82.88 to 83.10, $p < 0.05, t = 1.993$). The improvement is probably due to the elimination of unreliable MVDM values. We will adopt the new HAPAX representation with threshold 5 in subsequent experiments unless otherwise noted.

5.8 Fewer class values

As we already noted in Section 3.2.1, the frequency distribution of classes is very unbalanced. For example, 100 classes occur only once in our restricted cross validation material (1/0 restriction on verb chunks). This means that such a class either occurs only in the training material, in which case it does not contribute anything useful to testing performance, or it only occurs in the test material, in which case the learner has no chance to predict it correctly. Of the 305 classes exemplified in our restricted cross validation material, only 145 were predicted by the learner with the best performing parameter setting and

[4] Reading in the training and test material, counting values, and deciding which values to replace and actually replacing them takes time. This is why the experiment with threshold zero takes longer than the otherwise identical experiment in the last row of Table 5.14.

feature representation (see third row of Table 5.16). The more classes we have, the more memory and time is needed for storing the class distribution and for computing MVDM. We therefore replaced all classes that occur at most 5 times in the training material by the default "no relation" class. This change affects about 0.1% of the instances. Note that the gold standard material against which performance is measured is not changed. Table 5.17 shows that the replacement speeds up classification slightly and does not change the memory requirement. Precision increases slightly and recall decreases slightly. F_β stays the same.

5.9 Discussion

5.9.1 Fewer features

It is quite difficult to say why a certain feature is not useful. For the PoS features a likely reason is that their information is already implicitly contained in the word feature (through the value differences, cf. Section 4.3.7.2). A similar argument is presented in Van den Bosch and Buchholz (2002) in the context of chunking and function tagging.

5.9.1.1 The left context word: subcategorization, PP attachment and control

It is common in parsing to base a decision on local context. However the precise definition of local context varies. In addition, the merit of a local context feature also depends on the parser architecture. Both Ratnaparkhi (1997) and Charniak (2000) use a symmetric left/right context. In our experiment, an asymmetric context (more information from the left than from the right) proved best.

Ratnaparkhi (1997) uses the constituent label (or PoS if a constituent has not been constructed yet) and the headword of two elements to the left and two to the right as conditioning information in his Maximum Entropy parser. In his system the headword of a PP is the preposition, so he conditions on information that is similar to our prepositional features. In our experiments the prepositional features of the context elements do not contribute to performance.

Charniak (2000) conditions generation of a constituent on the category labels (our chunk type features) of up to three previously generated siblings but not on their headwords. In our experiments the word feature of the left context proves useful. To find out why this is the case we perform further analyses by breaking down performance by types of instances. Types are defined by the values of selected features. For each type we compute the number of instances that are classified correctly without and with the left context feature. The difference between the two numbers is the absolute improvement. By dividing both numbers through the total number of instances of this type, we get two accuracy figures. These can be used to compute the relative change in error rate. Suppose accuracy

instance type				absolute	without	with	error rate
dist.	context	focus	class	improv.	focus context	word−1	change
+1	VP	NP	NP-PRD	509	77.04	96.75	−85.83
+1	VP	NP	NP-OBJ	358	94.81	96.80	−38.41
−1	VP	NP	-	247	74.44	89.09	−57.31
+2	NP	PNP	-	183	91.83	94.77	−35.95
−1	VP	NP	NP-SBJ	131	89.85	94.45	−45.33
+1	VP	NP	-	114	83.77	86.19	−14.94
+1	VP	PNP	PP-DIR	78	57.21	66.83	−22.48
−2	VP	NP	NP-SBJ	75	62.53	83.20	−55.15
+1	VP	PNP	PP-PRD	64	38.58	62.55	−39.02
+1	VP	PNP	PP-LOC-PRD	53	34.67	61.31	−40.77
total				2539	90.13	90.83	−7.12

Table 5.18: Absolute improvement in number of instances, accuracy, and error rate change on most improved instance types without and with the head word of the immediate left context of the focus. (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$)

changes from 85% to 90%. This means the error rate drops from 15% to 10%. These are one third *fewer errors*, so the relative error rate change is −33%. If on the other hand accuracy changes from 50% to 55%, the relative error rate change is only −10%.

Table 5.18 shows part of the breakdown for instance types consisting of the distance, the chunk type of the left context and the focus, and the instance's class. The following cases can be distinguished:

- Subcategorization: The types "+1 VP NP *class*" and "+1 VP PNP *class*" are typical constellations for subcategorization information. The verb whose relation we want to determine directly precedes the focus NP or PNP. Note that the identity of the verb is known even if the left context word feature is not present, as it is also represented in the special verb feature. However the weight of the verb feature is lower than the weight of the left context word (cf. Figure 4.5), so it does not have as much influence in classification.

- PP attachment: The type "+2 NP PNP −" represents the classical PP attachment environment: the PP can either attach to the NP directly in front or to the VP two positions in front. Knowing the identity of the NP's head[5] helps the algorithm to decide whether it is likely that the PP attaches to the NP. In that case the relation to the verb would be "−", i.e. no (direct) relation. Typical nouns in this constellation include "damage (to)", "access (to)" and "look (at)", i.e. nouns that subcategorize for a PP.

[5]the N_1 in the PP attachment literature, cf. Section 2.4.1.2

... which will allow several signals to travel along a single optical line ...
... the company doesn't currently have any plans to sell additional newspapers ...

```
(VP (VB allow)                                    (VP (VB have)
  (S                                                (NP (DT any) (NNS plans)
    (NP-SBJ (JJ several) (NNS signals) )              (S
    (VP (TO to)                                       (NP-SBJ (-NONE- *) )
      (VP (VB travel)                                 (VP (TO to)
        (PP-DIR (IN along)                              (VP (VB sell)
          ... ))))))                                      ...)))))
```

Figure 5.3: Two sentence fragments which contain the sequence verb, noun chunk and infinitival verb chunk in two different structures: object control (left) and infinitive under NP (right).

- Control: The types "−1 VP NP −" and "−1 VP NP NP-SBJ" are exemplified by the two sentence fragments in Figure 5.3. To determine whether the noun chunk is the subject of the second verb (left) or is not dependent on it at all (right; in fact the verb depends on the noun chunk) the algorithm needs to know whether the first verb is an (object) control verb (left) or not (right).

5.9.1.2 The verb feature: implicit subcategorization information

As we saw in the previous section, the left context word already encodes subcategorization information in many cases. Still the verb feature improves performance. To study its contribution, we separately computed F_β without and with the verb feature for all relations which occur more than 500 times and sorted results by relative error rate change. Table 5.19 shows that, as predicted by subcategorization theories, error on typical complement relations like objects and predicatives drops while error on typical adjunct relations like PP-PRP[6] rises slightly (as the verb constitutes "noise" for them).

This observation allows us to divide the table into three parts: complement relations, problematic cases for a binary complement/adjunct distinction, and adjunct relations. We will investigate some interesting points in the three parts in more depth in the following paragraphs.

The relations NP-EXT (extensional NP, e.g. "rose 2%") and PP-DIR (directional PP) which are normally not considered subcategorized functions show up in the complement part of the table. There are two explanations for this. Either they *are* subcategorized (contrary to common theory) or there is some other lexical property of the verb, for example its meaning, that licenses these relations. Given our data, it is impossible to tell the difference.

[6]PP of purpose or reason

relation	frequency	without verb	with verb	error rate change
VP/S/SBAR-OBJ	3100	67.71	78.40	−33.11
VP/S-TPC	2027	3.09	21.56	−19.05
VP/S-OBJ	1469	39.24	48.90	−15.90
NP-PRD	3387	85.58	87.54	−13.64
PRT	1405	99.71	99.75	−12.59
NP-OBJ	20898	92.55	93.41	−11.58
PP-DIR	2826	74.71	77.24	−10.03
NP-EXT	1109	90.28	91.18	−9.29
SBAR-OBJ	1954	87.49	88.62	−9.07
ADJP-PRD	3473	94.63	95.01	−7.23
PP-LGS	1365	92.35	92.88	−6.97
PP-CLR	6145	67.72	68.96	−3.81
NP-SBJ	38100	90.87	91.21	−3.80
VP-OBJ	737	30.73	32.93	−3.17
ADVP-MNR	712	72.34	73.06	−2.62
PP-TMP	3656	69.22	69.85	−2.03
VP/S-PRP	684	29.64	30.39	−1.06
PP-LOC	3232	55.87	56.20	−0.74
PP	4233	43.80	44.17	−0.65
SBAR-ADV	1443	57.61	57.84	−0.53
PP-MNR	746	41.28	41.49	−0.35
SBAR-TMP	625	62.37	62.47	−0.27
ADVP	3196	74.20	74.14	0.22
VP/S	523	2.20	2.19	0.00
ADVP-TMP	1863	74.41	74.07	1.35
NP-TMP	1724	73.93	73.57	1.37
PP-PRP	518	26.89	25.82	1.46
VP/S-ADV	1150	51.75	50.91	1.73

Table 5.19: F_β and error rate change on most frequent relations without and with the verb feature (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$)

Logical subjects of passives (NP-LGS), "closely related" PPs (PP-CLR) and subjects (of actives) are considered by some theories to be subcategorized, and by some others not. In any case F_β is already reasonable or good without the verb feature. For the active and passive subjects it is clear which information achieves this: the configuration for the actives, and the *by*-preposition for the passives. For the PP-CLR this is less clear. Part of the performance can be explained by the fact that the verb is sometimes contained in the left context. We therefore looked specifically at instances where this is not the case (i.e. with distance greater than one). The following patterns emerged from the analysis:

- typical "financialspeak" PPs like "(quoted) at x yen", "(closed) at \$ x" or "(sell/buy) for x million". The combination of the preposition and the head noun is already sufficient information.
- "(attributed) (something) to factors": As the semantics of the head noun rule out a directional or dative interpretation of the PP, PP-CLR is the most likely alternative.
- "(prevent) (somebody) from doing (something)": Again for this combination the most likely class is PP-CLR.

For many of the relations in the adjunct part of Table 5.19 the error rate actually drops when adding the verb. This cannot be explained by theories of subcategorization. We therefore had a closer look at the verbs and relations for which the improvement occurred, to see what other factors could explain the effect.

- *write, says* and *adds* and ADVP-MNR: Apparently *if* there is some adverb following a verb of saying, it is most likely that it is an adverb of manner. This might be a combination of the facts that locations are rarely relevant for verbs of saying, that temporal expressions are less common with the present tense and that things can be said in many different ways.
- *been* and PP-TMP: The frequent co-occurrence of "been" with a temporal expression is clearly due to the English tense system. In fact it is more the presence of a temporal expression like "since 1985" that triggers the present perfect form of "be".
- *met with* or *bribed* and VP/S-PRP, e.g. "bribed officials to win ... account" These actions are not performed by coincidence. People need strong reasons to bribe someone for example. Also one can normally not guess precisely for what reason someone attempts bribery. Therefore it is natural to mention the reason for the bribery together with the fact.
- *fell (75 cents) in (over-the-counter) trading* (PP-LOC): This seems to be a recurrent expression that is usually annotated with the PP-LOC relation.
- *said* and PP: This connection stems from sentences like "For the quarter, Textron said aerospace revenue declined 9.8%" in which the PP is annotated to have an unspecified relation to "said".

- *behave* is the classical example from the subcategorization literature of a verb that needs an expression of manner. In our data it occurs in the form "behaved like calves".

5.9.2 Combining and splitting features

Our experiments with combined and split features and also with fewer features illustrate two interesting properties of (our implementation of) MBL. First it is not necessarily the case that a representation with fewer features is faster. Apparently a linguistically plausible division of information over more than one feature can speed up classification. Second the algorithm can cope very well with feature redundancy. None of the (partially) redundant features or feature representations caused a significant decrease in performance.

5.9.3 More features

See Section 5.9.3.3 for a discussion of the "PoS of the verb" feature. The other three groups of new features are discussed in the following three paragraphs.

5.9.3.1 Context window around focus and verb chunk

There is another source of (partial) redundancy in our representation, which we already saw in Section 5.1. If the verb chunk is directly in front of the focus chunk, its head appears in the verb feature as well as in the left focus context word feature. When using a larger left focus context or when also using context of the verb chunk, the effect grows, as more instances will contain duplicated values. We could prevent this redundancy by not representing the verb chunk. However as Table 5.1 showed this would hurt performance.

The need to represent the verb as an independent feature instead of just as a context element has also been noted for other approaches. Figure 5.4 (Ratnaparkhi, 1998, p.66) shows a typical PP attachment decision in Ratnaparkhi's shift-reduce-style Maximum Entropy parser (cf. Section 2.4.2.6). It takes the form of a BUILD operation on the first NP: if BUILD decides "Join VP", the PP is later (once it is built) free to attach under the VP, too. If on the other hand BUILD decides "Start NP", the PP must attach under the NP. Collins (1999, p.224) gives an example in which the verb falls outside the context in approaches similar to Ratnaparkhi's. It is reproduced here with Ratnaparkhi-style annotations in Figure 5.5. The left context of the decision bearing (second) NP includes the (first) preposition and the (first) NP, but not the verb. In head-driven generative approaches like Collins (1997) or Charniak (1997) (cf. Section 2.4.2.4 and 2.4.2.5) the verb is guaranteed to be represented because it is a head.

In summary this case constitutes another example of the fact that (partial) redundancy need not be disadvantageous.

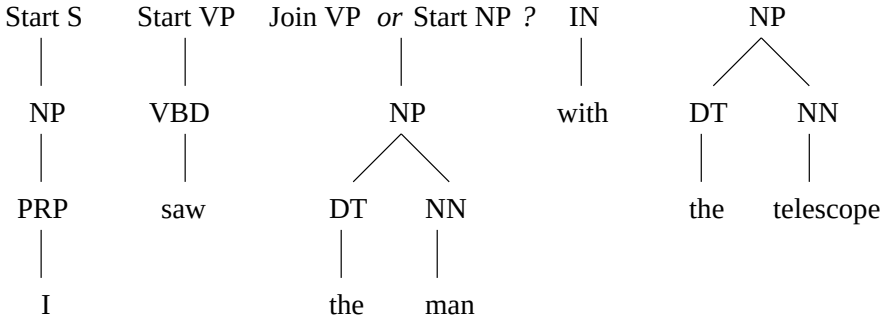

Figure 5.4: Example of a PP attachment decision in Ratnaparkhi's representation.

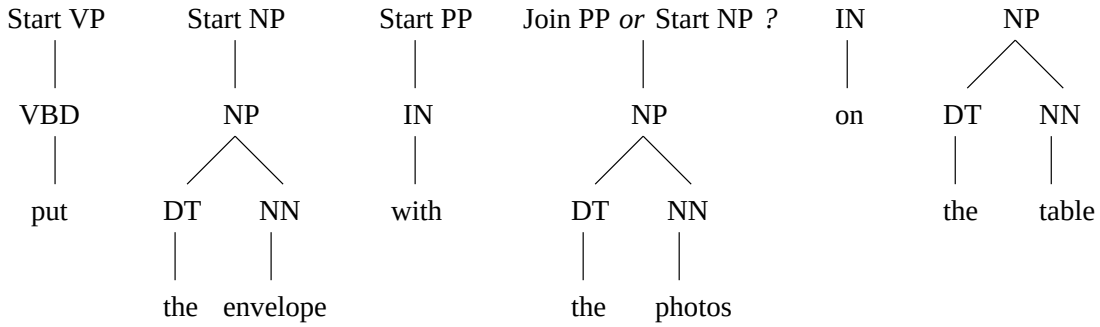

Figure 5.5: Example in which the (important) verb information falls outside the context scope of two elements to the left.

5.9.3.2 Global features of the front, back, and intervening material

Addition of the "back material" decreases performance significantly whereas the front material does not. This fits with previous results that the left context is more informative than the right context and with general psycholinguistic intuitions that, as processing a sentence is an incremental process, it is logical that the part that has already been heard/read has more influence on the GR analysis of the focus than the part that has not been heard/read yet.

The intervening material represents most of the relevant syntactic structure that determines whether the focus attaches to the verb, and, for the relations that are expressed configurationally in English, what type of relation holds between the two. The representation of the intervening material as a sequence of PoS resembles the MBSL approach (cf. Section 2.4.1.1). However MBSL implements a much more sophisticated similarity metric between sequences. In our case MVDM enables the algorithm to compute a sort of task-specific similarity over the symbolic values. 94% of the intervening chunk sequence types occur only with the "no relation" class, so their pairwise value difference is zero. As an example for the value differences between values that do occur with relations, Table 5.20 shows the intervening chunk sequences that are most similar to the empty sequence (i.e. verb and focus are adjacent). We can see for example that a single intervening ADVP,

value difference	sequence	frequency
0.485	ADVP	3837
0.586	DT NP	4
0.666	“	654
0.701	”	211
0.703	, ADVP ,	139
0.714	, PNP , NP ,	9
0.732	PNP , PNP PNP ,	10
0.746	VBP	5
0.772	, PNP ,	281
0.826	, NP CC NP	72
0.827	” PNP	28
0.834	, NP , NP ,	78
0.851	PNP , PNP ,	65
0.866	PNP “	40
0.871	PNP ADVP	246
0.874	, PP PNP ,	16
0.874	VP NP CC	308
0.883	PNP	9829

Table 5.20: Some intervening chunk sequences (and their frequencies). The first column shows the value difference between this sequence and the empty sequence (i.e. verb and focus are adjacent).

whether or not separated by commas, does not have much influence on relations. Likewise quotes can intervene between verbs and for example their subjects and objects. The sequence ", NP CC NP" stems from the treatment of coordinated subjects. In the two sentences "John laughed" and "John, Peter and Mary laughed" there is a subject relation between "John" and "laughed" (in addition to the subject relations "Peter laughed" and "Mary laughed").

Although MVDM obviously performs well, it is clear that the information in the sequences in not optimally exploited. To see this, suppose the algorithm encounters a sequence (say ", ADVP NP ADJP ,") in the test data that it has never "seen" in the training data. Then no MVDM values can be computed between this sequence and the other values of the same feature in training data when searching for the Nearest Neighbors and all training values will be taken to be equally (dis)similar to the test value. True sequence matching on the other hand might tell the algorithm that ", ADVP NP ADJP ," is much closer to, say ", NP ADJP ," than to "SBAR VP", especially if the MVDM matrix contains many pairs that differ only in the presence or absence of "ADVP" and have a low MVDM value difference.

Clark (2002) describes a method for representing an unbounded sequence of symbols by a finite number of numeric features. First he trains a stochastic transducer, or Pair Hidden Markov Model, on the training sequences. Then he computes the expected number of times each transition parameter and each output parameter of the model would be used for generating a test sequence. These numbers are then taken as feature values to represent the test sequence. Our approach for converting a symbolic sequence to a set of numeric features is much simpler. It performs at least as well as the symbolic representation. For our above example it can capture the relative similarity intuition. However it cannot represent the distinction between for example ", NP" and "NP ,".

The decrease in performance when using binary instead of numeric features is mainly due to the comma feature. This effect is not surprising, as one comma between the verb and the focus usually indicates that they are in separate clauses (and thus cannot have a relation) whereas two commas merely indicate a comma separated element/clause between them (and thus a relation is possible). The importance of the comma feature is also confirmed by the fact that it is the numeric intervening feature that increases performance most (from 80.29 to 80.92) when added alone.

We have seen the influence of CC in the coordinated subject example. SBAR marks the start of a subclause which often means that verb and focus are in different clauses and therefore cannot have a relation (the same argument was already used for the intervening VCs feature). Even if they could be in the same clause it is rather unlikely that a relation to the right spans a constituent that includes a subclause, as such a constituent is "heavy" and therefore either extraposition, an alternating subcategorization frame or an alternative position for an adjunct would normally be preferred. In the following examples, the second sentence of each group is awkward, as a relation spans a heavy constituent:

(1) I gave the man a book.
?I gave the man that I met when I was looking for a new appartment last year a book.
I gave a book to the man that I met when I was looking for a new appartment last year.

(2) I saw the man yesterday.
?I saw the man that I met when I was looking for a new appartment last year yesterday.
Yesterday I saw the man that I met when I was looking for a new appartment last year.

A model like that of Charniak (1997) could not capture the distinction between the first two sentences of each example triple, as the intervening material is one NP in both cases.

Apart from the comma, CC and SBAR features, we also found information about intervening NPs, begin and end quotes and colons to be useful. It is interesting to note that the last four features did not improve performance significantly when added in isolation but did when added in combination with the other features. This might be due to the fact that these features contain only part of a useful piece of information. For example in the coordinated subject case, information is distributed among the comma, the CC and the NP feature.

5.9.3.3 Global features of the rest of the focus and verb chunk

To find out why the rest of the focus chunk (as PoS) is useful we break down results according to the most frequent types of relations. Table 5.21 shows that the biggest error rate reduction is achieved for the VP/S/SBAR-OBJ relation (complementizerless clausal object, e.g. of "say"). Further analysis (not shown here) revealed that other VP relations improve too. This suggests that the rest of the focus chunk is useful mainly because it implicitly encodes the VP type of the focus. This was confirmed by the experiments in Section 5.2.

There is also a slight error reduction for subjects. As subjects are the most frequent relation even small improvements have influence on overall performance. The improvement especially occurs if the headword of the focus chunk is a proper noun. In these cases the "rest of the focus chunk" enables the algorithm to make the distinction between temporal expressions and persons. For the example in Table 4.14 on page 99, the new feature would increase the distance for the focus chunks "Barbara May", "last week" and "Mr. Peters" because they have a non-empty rest, and would therefore make the learner more "sure" of its decision (larger majority for NP-TMP).

One low frequency relation for which we noted an improvement through the rest of the focus chunk is NP-CLR, which is mainly used for fixed expressions. These often contain a

relation	frequency	without rest of focus chunk	with rest of focus chunk	error rate change
NP-SBJ	38100	90.40	90.55	−1.46
NP-OBJ	20898	93.11	93.01	+1.45
PP-CLR	6145	69.96	69.29	+2.25
PP	4233	46.25	46.32	−0.13
PP-TMP	3656	72.47	72.52	−0.16
ADJP-PRD	3473	95.12	95.12	+0.02
NP-PRD	3387	87.88	87.86	+0.20
PP-LOC	3232	57.60	57.02	+1.36
ADVP	3196	74.01	74.33	−1.25
VP/S/SBAR-OBJ	3100	80.36	83.68	−16.90
NP-CLR	91	7.69	10.61	−3.17

Table 5.21: F_β and error rate change on the ten most frequent relations (and NP-CLR) when adding the rest of the focus chunk (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 features, split distance)

singular noun without determiner, like "keep track of" or "take advantage of". Thus the information that the rest of the focus chunk is empty is useful.

The experiments in Section 5.2 seem to suggest that the VP type feature values and the rest of the focus chunk feature largely encode the same information. However from the construction of the VP type values we know that, depending on whether the semantic head of the VP chunk is also its syntactic head, the VP type comes from the semantic head or from the "rest". This means that the focus VP type actually combines information from the focus PoS feature and the rest of the focus chunk. Similarly the VP type of the verb chunk combines information from the verb PoS feature and from the rest of the verb chunk which up to now we tested only separately. Table 5.14 shows that, indeed, adding the latter two performs roughly the same as adding the former (80.91 versus 81.04) and that adding all three features does not perform significantly better (81.22, $t = 1.576$) than just adding the verb VP type alone, while requiring more memory and run time. This is another case of distributed information. Neither of the latter two features significantly increases performance by itself but combined they do.

5.9.4 More feature values

The change to represent multi-word prepositions as such instead of just by their last word actually amounts to using the "rest of the PP chunk". It did not result in a significant performance increase and we therefore did not use these new feature values in subsequent experiments. However only 0.26% of the instances are actually affected by the change. For these, accuracy increases from 71.5% to 85.3%. So clearly the change had the effect we

New features	MB	h:m	precision	recall	F_β
VP type for focus	77	2:37	85.30	76.31	80.55
& VP type for verb	82	2:57	85.68	76.87	81.04
& verb PoS & rest of verb chunk (PoS)	87	2:56	85.59	76.71	80.91
& VP type for verb & verb PoS & rest of verb chunk (PoS)	90	3:13	85.88	77.04	81.22

Table 5.22: The VP type and the combination of PoS and "rest of" feature of the verb chunk encode similar information (1/0 restriction on intervening verb chunks, IB1-IG, Gain Ratio, MVDM, $k = 9$, Exponential Decay with $\alpha = 30$, ignoring 5 old features, split distance)

expected it to have. This shows that overall performance might not be the best way to judge the merit of a rather subtle change in the feature representation.

The VP type information of the focus chunk enables the algorithm to distinguish for example between finite and non-finite clauses. Consequently the error rate on for example VP/S/SBAR-OBJ drops by 16.6%. As we mentioned in Section 2.3.1, the treebank's decision to use the same syntactic label for finite and non-finite clauses was based more on practical considerations (fewer labels by collapsing lexically recoverable ones) than on a conviction that the two behave syntactically similarly. Different types of VP chunks have also been used by e.g. Carroll and Rooth (1998a).

The VP type of the verb chunk enables the algorithm to learn subject-verb agreement as it encodes information about the grammatical person. Error rate on NP-SBJ drops by 10.1%.

5.9.5 Fewer feature values

In most statistical parsing work, some treatment of low frequency events is necessary for the success of the parser. Often some smoothing method is applied to the observed frequencies. Ratnaparkhi (1997) constructs (binary) features for his Maximum Entropy parser only for events that occur at least five times. Collins (1999, p. 186) replaces words that occur less than five times by the token "UNKNOWN". This has the additional advantage of enabling the algorithm to learn if low frequency events differ systematically from higher frequency ones. Our HAPAX encoding is comparable.

There are more sophisticated representations than the simple HAPAX replacement. Eisner (1996a) and Van den Bosch and Buchholz (2002) use an encoding for low-frequency words that contains an approximation of morphological information (e.g. capitalization, suffix). The Memory-Based Tagger of Daelemans et al. (1996) encodes all of this information in separate features that are only used for unknown words. It would certainly be interesting to apply a similar encoding to our data. Note however that we applied the HAPAX encoding to all features, not only to the word-valued ones.

5.9.6 Fewer class values

It might seem as if prediction of the low frequency classes by the learner is so bad that we may as well not predict them at all. This is not true in general however. That we do not see a statistically significant effect of their deletion is merely due to the classes' low frequency. The rare class WHADVP/SBAR-PUT (*wh*-clause as locative complement of *put*), which appears only three times in our material, e.g. in "I suggest that the Wall Street Journal (...) should *put* its money *where* its mouth is", has a precision of 100% and a recall of 66.7% ($F_\beta = 80$) if we do not delete low frequency classes. The predicative clause of purpose and reason (SBAR-PRP-PRD) which occurs 13 times, e.g. in "That *'s because* they only drop "mere names," says Mr. Churchill." has a precision, recall and F_β of 84.6.

As we cannot know in advance whether anyone is interested in these classes, it would be unwise to prevent our learner from predicting them just because their frequency is low, as Table 5.17 also shows that having many rare classes does not *hurt* the performance of our learner in any way. From a practical point of view, deletion of infrequent classes of which we know that we will not need them in our application is a sensible way to increase speed. Classes with a frequency below eleven are seldomly predicted by the learner.

5.10 Summary

In this chapter, we improved upon the preliminary feature representation. Performance rose from 80.09 to 83.10, mostly through the addition of ten new features and eight new feature values. The new features (values) contain information about the local left context of the verb and the focus chunk, about certain chunks and punctuation marks in between the verb and the focus chunk, and about the syntactic head of the verb chunk or the focus chunk if the latter is verbal. Their values are words, integer numbers and PoS tags, respectively.

Memory requirements stayed practically constant (107 versus 108 MB) and run time even decreased (from 4:48 to 3:27 hours) because apart from adding new features (values) we also discarded five old features and all feature values with a frequency less than or equal to five. Speed also increased by a separate representation of direction and distance and through the seven new numeric features. This demonstrates the surprising fact that using more features can be faster. Other interesting properties of the algorithm that were exemplified in this chapter are its ability to cope with redundant or sparse data and with information that is distributed over several features (the numeric ones, or the rest of the focus/verb and its PoS). We again saw the power of the MVDM metric by its ability to exploit the information in the symbolic representation of the intervening material and of the rest of the focus chunk. However it again proved to slow down classification.

In the case of the intervening material, we found a much faster representation that per-

formed as well as the symbolic representation by a straightforward transformation to numeric features that did not use linguistic knowledge. In the case of the symbolic features "rest of the focus/verb chunk" (plus PoS) and the faster variant VP type, the connection is not that straightforward. We need linguistic knowledge to determine that the VP type of "to laugh" and of "not to laugh" is the same (*to*-infinitive) but for "wants to laugh" it is different (third person singular present).

We are not only interested in what information is useful for finding grammatical relations but also why. To answer this question we performed extensive analyses in which we compared classification with and without a certain feature (or set of feature values). We employed two kinds of analyses. In one, classification accuracy is broken down by types of instances, in the other, F_β is broken down by types of relations. Both analyses allow us to better understand where the improvement occurs. We can then relate patterns of improvement to linguistic phenomena like subcategorization, PP attachment, control, coordination, fixed expressions, finite versus infinite clauses, and agreement. This analysis is not only interesting in the context of machine learning of grammatical relations. We showed how the comparison between classification with and without the verb feature allows us to sort grammatical relations from typical complements to typical adjuncts. This provides empirical evidence for the theory that for example the extensional NP and directional PP relations are dependent on lexical properties of the verb. We could further identify individual cases in which the addition of the verb does not influence performance although in general the relation has been classified as subcategorized, or on the contrary, addition improves performance although in general the relation has been classified as adjunct. Examples like "been" with a temporal expression suggest that the algorithm does not only learn subcategorization but also other factors (here: tense) that influence co-occurrence. We believe that analyses like these can prove fruitful for (psycho)linguistic research into subcategorization and other lexical properties. It is clear that the data that resulted from our experiments contains many more interesting patterns. However a complete analysis is beyond the scope of this thesis. The discovery of these patterns is left for future research.

Chapter 6

Integration into the Memory-Based Shallow Parser and comparisons

The two previous chapters described how we improved performance by optimizing the algorithmic settings and the feature representation on the task of finding grammatical relations, using the task definition and the data as introduced in Chapter 3. In this chapter we use the optimal settings and feature representation but apply it to slightly different tasks and data in Section 6.1.[1] These experiments pave the way for a comparison of MBSP to related work in Section 6.2.

6.1 Additional tasks and data

In contrast to the previous chapters, the experiments in this section reflect more realistic conditions for the relation finder. In particular we will also find non-local dependencies (Section 6.1.1), use a smaller set of coarser defined relations (Section 6.1.2), work on other data than the Wall Street Journal Corpus (Section 6.1.3) and, perhaps most importantly, work on data that was tagged and chunked automatically instead of using the hand-corrected treebank annotation (Section 6.1.4).

6.1.1 Non-local dependencies

In Section 3.1.2.6 we explained how non-local dependencies are represented in the Penn Treebank. Up to now we excluded these relations from our experiments. They are however important in applications like question answering or subcategorization extraction where we want to find the object of a verb, for example, no matter whether it stands in a local

[1] There is no guarantee that the setting and representation that proved to work best for the task and data of the previous chapters is also the best one for the other tasks and data. However it is infeasible to optimize everything again.

```
(S
 (NP-SBJ (DT This) )
 (VP (VBZ is)
   (NP-PRD (DT an) (JJ old) (NN story) )))
```

```
(S
  (NP-SBJ-1 (PRP He) )
  (VP (MD will)
    (VP (VB be)
      (VP (VBN paid)
      (NP (-NONE- *-1) )))))
```

```
(NP
  (NP (DT the) (JJ Japanese) (NNS companies) )
  (SBAR
    (WHNP-1 (WDT that) )
    (S
      (NP-SBJ (-NONE- *T*-1) )
      (VP (VBP compete)
        (PP-CLR (IN with)
          (NP (PRP it) ))))))
```

```
(NP-SBJ
  (NP (DT The) (NN plant) )
  (, ,)
  (SBAR
    (WHNP-1 (WDT which) )
    (S
      (NP-SBJ-4 (-NONE- *T*-1) )
      (VP (VBZ is)
        (VP (VBN owned)
          (NP (-NONE- *-4) )
          (PP (IN by)
            (NP-LGS ... )))))))
```

Figure 6.1: Four combinations of local and non-local relations: "**This is** an old story": only a local relation (here: surface subject) between the bold face words. "**He** will be **paid**": a local and a non-local relation (surface subject and deep object). "the Japanese companies **that compete** with it": only one non-local relation (surface subject). "The plant, **which** is **owned** by ...": two non-local relations (surface subject and deep object).

or a non-local relation to the verb. This section reports on experiments that include the non-local relations. There can be at most one local but several non-local relations between a verb and some other word. We can distinguish four combinations, which are exemplified in Figure 6.1. The corresponding instances in Table 6.1 contrast the class representation we used up to now with the one used in this section. For distinction, non-local relations are prefixed by "T-" (trace). The table also shows the frequency of each of the four combinations of local and non-local relations. We see that non-local relations are much less frequent than local ones. Still 10.8% of the relations are affected by the class change so its influence is not negligible. In the above examples, passive and *wh*-movement is the source of the non-local dependency. Another source is topicalization, which is exemplified in Figure 6.2.

A first experiment including the non-local relations yields disappointing results. F_β drops to 80.02, 3.08 points less than the 83.10 we had reached at the end of the previous chapter. The first and second row in Table 6.2 show that this decrease is entirely due to the performance on the non-local relations themselves whereas performance on the local relations is not affected significantly by the addition. A quick analysis of the errors involving non-local relations showed that the learner is especially bad at making the distinction between pure surface subjects, and those that are also deep objects. This is not surprising, as the feature representation lacks the information whether the verb chunk is passive or not. In a second

features				class		frequency of case	
dir.	dist.	focus	verb	old	new	absolute	%
–	1	This	is	NP-SBJ	NP-SBJ	114,701	89.2
–	1	he	paid	NP-SBJ	NP-SBJ;T-NP-OBJ	5,135	4.0
–	1	that	compete	–	T-NP-SBJ	8,226	6.4
–	1	which	owned	–	T-NP-OBJ;T-NP-SBJ	484	0.4

Table 6.1: Partial instances for the four combinations of local and non-local relations from Figure 6.1 with old and new class values. The last two columns show the absolute and relative frequency of each case in the unrestricted cross validation material.

```
( (SINV (`` ``)
    (S-TPC-1
      (SBAR-ADV (IN Unless)
        (S
          (NP-SBJ (PRP he) )
          (VP (VBZ changes) )))
      (, ,)
      (NP-SBJ (PRP they) )
      (VP (VBP lose) ))
    (, ,) ('' '')
    (VP (VBD said)
      (S (-NONE- *T*-1) ))
    (NP-SBJ (DT a) (JJ Democratic) (NN leadership) (NN aide) )
    (. .) ))

( (SINV
    (S-TPC-1
      (NP-SBJ (DT The) (NN perception) )
      (VP (VBZ lingers) ))
    (, ,)
    (VP (VBZ says)
      (SBAR (-NONE- 0)
        (S (-NONE- *T*-1) )))
    (NP-SBJ
      (NP (DT an) (NN official) )
      (PP-LOC (IN at)
        (NP (DT a) (JJ major) (JJ industrial) (NN company) )))
    (. .) ))
```

Figure 6.2: The difference between direct and indirect speech is expressed through the non-local relations (`S-OBJ` or `S/SBAR-OBJ`). The local relation is the same (`VP/S-TPC`: topicalized sentence) in both cases: "Unless he changes, they **lose**," **said** a Democratic leadership aide. The perception **lingers**, **says** an official at a major industrial company.

experiment we added this information in the form of additional feature values of the VP type feature of the verb chunk. If the words in the verb chunk and their PoS match

non-loc.	pas-sive		MB	h:m	local			non-local			both		
					prec.	rec.	F_β	prec.	rec.	F_β	prec.	rec.	F_β
no	no		108	3:34	86.45	80.01	83.10						
yes	no		109	3:50	86.20	80.11	83.04	76.08	46.93	58.04	83.67	76.67	80.02
no	yes		108	3:15	86.48	80.02	83.13						
yes	yes		109	3:55	86.37	80.19	83.17	80.92	57.68	67.34	85.08	77.95	81.36
yes	yes	*	108	3:27							85.12	77.97	81.39

Table 6.2: Performance with and without the non-local relations, with and without the passive feature values of the VP type feature, on local and non-local relations and overall. Parameter settings and rest of feature representation same as for third row of Table 5.16. *= same representation for local and non-local relations

local relation	freq.	prec.	rec.	F_β	non-local relation	freq.	prec.	rec.	F_β
NP-SBJ	35141	92.52	93.29	92.90	T-NP-SBJ	6015	82.86	60.85	70.17
NP-SBJ; T-NP-OBJ	2948	90.41	56.31	69.40	T-NP-OBJ; T-NP-SBJ	482	89.33	50.41	64.45
NP-OBJ	20898	93.32	95.17	94.24	T-NP-OBJ	514	74.15	63.62	68.48
PP-CLR	6143	70.00	71.25	70.62	T-PP-CLR	17	70.00	41.18	51.85
PP	4227	58.20	40.38	47.68	T-PP	34	27.27	8.82	13.33
PP-TMP	3656	77.63	67.86	72.42	T-PP-TMP	22	50.00	13.64	21.43
					T-ADVP-TMP	661	91.31	93.80	92.54
ADJP-PRD	3472	93.68	97.75	95.67	T-ADJP-PRD	19	75.00	31.58	44.44
NP-PRD	3384	89.75	91.37	90.55	T-NP-PRD	40	33.33	10.00	15.38
PP-LOC	3230	65.62	54.61	59.61	T-PP-LOC	74	55.05	81.08	65.57
					T-ADVP-LOC	163	80.84	82.82	81.82
ADVP	3196	76.16	74.66	75.40	T-ADVP	20	0.00	0.00	0.00
VP/S/SBAR -OBJ	3100	94.17	83.87	88.72	VP/S-TPC; T-S/SBAR-OBJ	873	60.96	24.86	35.31

Table 6.3: Performance on ten most frequent local relations and non-local counterparts (using passive feature).

a regular expression, the string "`-pas`" is concatenated to the original VP type. This additional information indeed increases overall performance to 81.39 and performance on the non-local relations by nearly 10 points from 58.04 to 67.34. The passive feature values do not have any significant effect on the local relations (first and third row of Table 6.2).

If an application does not need to distinguish between local and non-local relations we can represent both by the same class instead of using the "T-" prefix for the latter. This simplification does not have any significant effect on performance but classification is slightly faster (fifth row of Table 6.2).

Table 6.3 compares performance on local and non-local counterparts of the ten most frequent relations. Some local relations have two counterparts, e.g. the locative PP in "he met her *in New York*" has a non-local variant as a PP ("the city *in which* he met her") or as an ADVP ("the place *where* he met her"). The function of the latter is clearly marked by the lexical item *where*, so precision and recall on this relation are high. Performance on

	MB	h:m	prec.	rec.	F_β
with 1/0 restriction	109	3:55	85.08	77.95	81.36
without restriction	178	8:29	85.13	77.81	81.31

Table 6.4: Performance with and without the 1/0 restriction on intervening verb chunks.

the PP case is lower but still higher than for local PP-LOCs. This is because most cases of preposition plus "which" are T-PP-LOC whereas there are many more frequent alternatives for the general preposition-plus-noun case. In general, performance on non-local relations is much lower than on the local ones. This can be explained by the much lower frequency of the former and the fact that in *wh*-constructions, the preceding head noun, and not the *wh*-word, contains most of the semantic information.[2] Most of the reasonably frequent non-local relations also have reasonable performance. An exception are topicalized sentences. In this case the learner often fails to retrieve indirect speech or mistakes it for direct speech (see also Figure 6.2). Topicalized direct speech (VP/S-TPC;T-S-OBJ) is more frequent (1130) and the learner achieves higher performance for it (F_β=63.30). Unless stated otherwise, experiments in the remainder of this thesis include the passive feature and the non-local relations (in the T-prefixed representation).

In Section 4.2 we introduced the preprocessor, which was implemented as a 1/0 restriction on intervening verb chunks in all experiments up to now. In order to investigate its influence on performance with the new feature representation and the non-local classes, we repeated the last experiment without the restriction. Table 6.4 shows that both set-ups yield basically the same performance but that the restricted version is more than twice as fast and takes less memory. It will therefore be the preferred version for practical applications. On theoretical grounds, however, it is interesting to note that the method is, in principle, able to find very distant relations.[3]

6.1.2 Application-specific classes

Up to now the set of GRs that we used was the maximal set extractable from the Penn Treebank. Our reasoning was that different applications would need different subsets of these relations but that we cannot know in advance which relations might or might not be needed and we therefore wanted our results to be as general as possible. For a specific

[2]In fact one might want the grammatical relation to hold between the head noun and the verb. This could be achieved by more manipulations of the annotations derived from the treebank.

[3]Here are two examples of distant relations that the unrestricted version found: "**If** Sen. Bradley would permit a vote on capital gains, though, it would pass, Christmas retail sales would be strong instead of burdened by a falling stock market, the 1990 economy would be robust, and the revenue gains at every level of government, including New Jersey's, **would be** surprisingly high." (SBAR-ADV); "There **are** about $10 million of 7% bonds priced at 99 1/4 to yield 7.081% in 2004; about $15 million of 7% bonds priced at 98 1/2 to yield 7.145% in 2008; about $88.35 million of 7% bonds priced at 97 1/4 to yield 7.227% in 2018; and **about $15 million** of 6 3/4% bonds priced to yield 7.15% in 2019." (NP-PRD).

Task	# cl.	preprocessing					postprocessing		
		MB	h:m	prec.	rec.	F_β	prec.	rec.	F_β
unlabeled depend.	1	102	1:46	89.32	85.55	87.39	93.19	86.52	89.73
subject/object	2	103	1:55	94.65	91.38	92.99	94.59	93.92	94.25
QA classes	9	105	2:06	87.08	80.74	83.79	88.51	81.10	84.64
C&B's GRs	13	105	2:06	89.00	83.05	85.92	90.23	82.67	86.28

Table 6.5: Results with preprocessing and postprocessing for task-specific class prediction. The second column shows how many non-default classes there are for each task. For postprocessing MB is 109 and h:m is 3:55. Run time excludes the preprocessing or postprocessing step.

application, we would then need a mapping from our treebank classes to the application-specific classes. There are two ways to realize this mapping:

- by preprocessing i.e. mapping the classes in the training material. The application-specific classifier then directly predicts the new classes.
- by postprocessing i.e. mapping the classes after classification but before using them in the application.

To find out what consequences on performance, speed and memory the two alternatives have, we conducted preprocessing and postprocessing experiments with four different mappings. The first mapping corresponds to the task of predicting unlabeled dependencies (attachment): all classes except pure non-local relations and "–" (no relation) are mapped to, say, `REL`. The second task is to extract only NP subjects and direct objects. Thus all classes except local `NP-SBJ` and `NP-OBJ` are mapped to "–". The third mapping was used for the Question Answering system that will be described in Chapter 7. It does not distinguish between different types of objects, between syntactic categories, between local and non-local relations and between PP complements and adjuncts. Finally we test the mapping from our GRs to the ones of Carroll, Minnen, and Briscoe (1998), henceforth C&B. It will be explained in more detail in Section 6.2.2.[4] The most important points are that their GRs do not distinguish between semantic adjunct types (`-TMP`, `-LOC` etc.), between most syntactic categories (`NP`, `ADJP`, `ADVP` etc.) and between local and non-local relations. Very roughly the QA mapping keeps mostly adjunct distinctions whereas the C&B mapping keeps mostly complement distinctions. In all four cases, the gold standard against which the results are evaluated is mapped, too. This is why F_β values are higher than in the previous experiment (81.36).

The results demonstrate several points. First, unsurprisingly, the task gets easier with fewer classes. Second, subjects and objects are easier to find than most other relations. Third, postprocessing consistently yields better performance than preprocessing but is much slower. This means that even if we need the learner to perform only a simple(r) task

[4] The auxiliary and determiner relations only hold within chunks so they do not occur in this experiment.

it is still better for performance to train it on a more difficult task.[5] If we have several applications with different target classes they can use the same learner and just apply different mappings afterwards if memory but not speed is a concern. On the other hand tailoring the learner to the specific classes needed is a way to speed up classification.

To find out why preprocessing performs worse we analyze the data of the "C&B's GRs" experiment in more detail. A breakdown of performance by relation shows that precision on adjuncts decreases when switching from postprocessing to preprocessing but recall increases more, so F_β increases too. On complements however either the opposite happens (precision higher, recall lower, F_β lower) or precision as well as recall are lower. Thus on adjuncts the learner becomes less "over-cautious". This is the effect of merging several different semantic types of adjuncts into one general adjunct class. It then becomes more likely for this new class to get the majority among the Nearest Neighbors, i.e. to get predicted. Take the sentence "... the Secret Connection attache case which can surreptitiously record conversations for nine hours at a stretch." Without preprocessing, the NN distribution for the pair *record/at stretch* is {`-: 0.0323755, PP-CLR: 0.00800639, PP-TMP: 0.0217656, PP-LOC: 0.00826969, PP-MNR: 0.00730642`}, so "–" (no relation) has the majority (which is correct because "at a stretch" attaches to "hours"). After preprocessing the distribution becomes {`-: 0.422741, ncmod: 0.712311, iobj: 0.140134`} so "ncmod" (non-clausal modifier) gets predicted. In this case this is incorrect but often it is right.

As an example of why performance on complements drops, take the phrase "which traded 11 million shares". Without preprocessing, the NN distribution for the pair *traded/shares* is {`NP-OBJ: 0.185321`} i.e. "shares" unanimously gets classified as direct object. After preprocessing, the distribution becomes {`ncmod: 0.996092, dobj: 0.787955`}, so "shares" is incorrectly classified as an adjunct. The NNs in the postprocessing case are phrases like "who sold shares", "which bought machines" etc. while in the preprocessing case they are "which soared 9%", "Saab ended talks", "stock plunged HAPAX" etc. which look much less related. This is partially due to a lower relative importance of the word-valued features with preprocessing and partially to different MVDM values. Apparently, having to distinguish between semantic types of adjuncts helps the algorithm to better estimate the value of lexical information.

6.1.3 Influence of text type

We started our experiments with data from the WSJ Corpus because that was the largest corpus with hand-corrected parse trees featuring information on different kinds of complements and adjuncts. Later however the parsed Brown Corpus became available as part

[5] It might still be the case that we could find a parameter setting and feature representation that is better suited for the simple task and yields even better performance than the "original task and postprocessing" set-up. However this would involve extensive optimization.

	# words	# sentences	av. sentence length	# genres
WSJ	1,173,766	49,208	23.9	1
Brown	459,148	24,243	18.9	8

Table 6.6: Characteristics of the WSJ and the Brown Corpus.

↓ train \ test →	WSJ	Brown
WSJ	80.03	74.23
Brown	73.10	78.87

↓ train \ test →	WSJ	Brown
WSJ	17.7	29.9
Brown	37.5	18.3

Table 6.7: Left: F_β on different training and test corpora. Right: Percentage of test instances whose focus word has been replaced by HAPAX. Experiments are without the `-CLR` function tag.

of the third release of the Penn Treebank. The characteristics of the two corpora are compared in Table 6.6.

The annotation scheme in both corpora is nearly the same.[6] We can therefore test what influence the text type and differences in text type between training and test material have on relation finding performance by training on either WSJ or Brown and testing on either WSJ or Brown. We use the complete Penn Treebank Brown Corpus (459,148 tokens) and files 0001–0999 and 2000–2035 of the WSJ Corpus (459,136 tokens). The WSJ part deliberately excludes the files that we used for the optimization experiments in the previous chapters (1000–1999). We randomize both corpora at the sentence level and then split them into equally sized training and test partitions. The results are shown in Table 6.7 (left part).

The WSJ/WSJ performance is lower than in previous experiments, probably because we use only half the amount of training material. The Brown corpus seems to be slightly more difficult given the lower performance when training and testing on the same source.[7] Unsurprisingly training and testing on a different corpus results in a lower performance. Interestingly training on Brown and testing on WSJ performs worse than the other way round. All these results appear to be related to the percentage of low frequency words in the test material (see right part of Table 6.7). When training on Brown and testing on WSJ, more than a third of the focus words have not been seen sufficiently often. Nearly half of these HAPAX words are common nouns (only less than a quarter are proper names). For example "%" occurs more than 1100 times in the WSJ data but only 5 times in the Brown part. Although it is typical for financial texts, we normally would not consider it specialized vocabulary. Another example is "million" which does not occur in the Brown data at all (only "millions" does).

[6] The `-CLR` function tag is no longer used in the Brown Corpus. In the WSJ files, we therefore replace it by `-OBJ` if it is the only tag on NPs, ADJPs, etc. (cf. p. 59) and delete it otherwise. The `-IMP` tag (for imperatives) exists only in the Brown Corpus. We therefore replace it by `-OBJ` if it is the only tag on S and SINV and delete it otherwise for the experiments in this section.

[7] There might also be a small effect of the fact that the parameters and features are optimized for WSJ, not for Brown.

context−5		..	context−1		focus		context+1		context+2		context+3		
word	PoS	..	word	PoS	word	PoS	word	PoS	word	PoS	word	PoS	class
−	−	..	−	−	Pie.	NNP	Vin.	NNP	,	,	61	CD	I-NP
−	−	..	Pie.	NNP	Vin.	NNP	,	,	61	CD	yea.	NNS	I-NP
−	−	..	Vin.	NNP	,	,	61	CD	yea.	NNS	old	JJ	O
−	−	..	,	,	61	CD	yea.	NNS	old	JJ	,	,	I-NP
−	−	..	61	CD	yea.	NNS	old	JJ	,	,	will	MD	I-NP

Table 6.8: The first chunking instances for the sentence "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.". Features: word and PoS of focus token (word or punctuation), of five context tokens to the left and of three context tokens to the right. Parameters settings: IB1-IG, Gain Ratio, MVDM for PoS features and overlap for word features, $k = 3$, majority voting.

Gildea (2001) performed related experiments with the WSJ and Brown corpus on the full parsing task (without function tags) using Collins's Model 1 parser. However he uses a larger training set from the WSJ than from Brown (950,000 versus 413,000 words) and does not explore training on Brown with testing on WSJ. Yet his results confirm that the Brown corpus is slightly more difficult and that training on a different corpus decreases performance by several percent.

6.1.4 Using a real tagger and chunker

Up to now we used the tags and chunks from the treebank and concentrated on improving the performance of the relation finder. For comparisons to other systems and for real applications however we need to use an automatic tagger, chunker and PNP finder. In this section we describe how these modules are implemented, what their influence on relation finding performance is and how they can best be combined with the relation finder. The Memory-Based Tagger (MBT) software is described in Daelemans et al. (1996). It is based on TiMBL (version 4.2 at the time of writing) but in addition it offers automatic conversion from sentences to instances and back, a sophisticated treatment of unknown words based on word-internal features (capitalization, suffix letters, etc.) and the use of the predicted class of a word as a feature for the classification of the nearby following words. Memory-based chunking has been studied by Veenstra (1998), Tjong Kim Sang and Veenstra (1999), Veenstra and Van den Bosch (2000) and others. We adopt the setting and instance representation as proposed by Veenstra and Van den Bosch (2000). Some sample instances are shown in Table 6.8.

We also introduce a novel module that tags and chunks at the same time. This is achieved by simply letting MBT predict tags that are a concatenation of a PoS tag and an IOB tag. In this notation our example sentence looks like:

dist.	# PPs	# ,s	prep.	context-1			focus		context+1			class
				head	PoS	chunk	head	PoS	head	PoS	chunk	
1	0	0	as	as	IN	PP	dir.	NN	29	CD	NP	+
2	0	0	as	dir.	NN	NP	29	CD	.	.	–	–

Table 6.9: The two PNP finding instances for our example sentence. Features: distance (in chunks) between preposition and NP, number of intervening PPs, number of intervening commas, preposition, headword, head's PoS and chunk type of one context chunk to the left, of focus chunk, and of one context chunk to the right. The chunk type of the focus is ignored as it is NP by definition. Instances are only created for distance values between 1 and 5 (similar to the search space restrictions in Section 4.2). Parameter settings: IB1-IG, Gain Ratio, overlap, $k = 1$, majority voting.

training			testing			relation finder performance				
tags	chunks	PNPs	tags	chunks	PNPs	MB	h:m	precision	recall	F_β
tb	tb	tb	tb	tb	tb	109	3:55	85.08	77.95	81.36
tb	tb	tb	tb	tb	Timbl	109	3:56	84.54	77.43	80.83
tb	tb	Timbl	tb	tb	Timbl	110	3:57	84.88	77.24	80.88
tb	tb	tb	tb	tb	regexp	109	4:01	83.57	77.42	80.38
tb	tb	regexp	tb	tb	regexp	112	4:03	84.77	77.14	80.78
tb	tb	tb	MBT		Timbl	109	4:01	75.24	67.90	71.38
MBT		Timbl	MBT		Timbl	111	4:24	79.96	66.47	72.59
tb	tb	tb	MBT		regexp	109	4:07	74.27	67.83	70.91
MBT		regexp	MBT		regexp	112	4:32	80.04	66.41	72.59

Table 6.10: Influence of automatically tagged and chunked data on relation finding performance. "tb" refers to treebank tags and/or chunks and "regexp" to the regular expression.

(1) Pierre/**NNP**-*I-NP* Vinken/**NNP**-*I-NP* ,/**,**-*O* 61/**CD**-*I-NP* years/**NNS**-*I-NP* old/**JJ**-*I-ADJP* ,/**,**-*O* will/**MD**-*I-VP* join/**VB**-*I-VP* the/**DT**-*I-NP* board/**NN**-*I-NP* as/**IN**-*I-PP* a/**DT**-*I-NP* nonexecutive/**JJ**-*I-NP* director/**NN**-*I-NP* Nov./**NNP**-*B-NP* 29/**CD**-*I-NP* ./**.**-*O*

The PNP finder instances are shown in Table 6.9. The PNP finder seems to be the simplest module of MBSP so we also try whether it can be replaced by a simple regular expression that joins all PP and NP chunks that match the pattern "[$_{PP}$ ...] [$_{NP}$ ...]" or "[$_{PP}$ ...] [$_{NP}$ ...] CC [$_{NP}$ ...]" or "[$_{PP}$ ...] [$_{NP}$ ...] , [$_{NP}$ ...] CC [$_{NP}$ ...]" (CC is a coordinating conjunction).

We train the tagger, chunker and PNP finder on sections 00–09 of the WSJ corpus and apply them to our cross validation material (from sections 10–19). For the relation finder we try two different set-ups: training on treebank data (as before) or training on output from the lower modules. Results are shown in Table 6.10. The first row shows our previous set-up: training and testing on treebank tags and chunks. In the second and third row we use TiMBL to predict PNPs (its F_β is 96.63). It does not seem to matter much whether the relation finder is trained on treebank or automatically predicted PNPs (80.83 versus 80.88).

It does matter for the regular expression PNP finder however (which achieves an F_β of only 92.92). If the relation finder can adapt to the imperfect output of the regular expression it performs nearly as well as with the TiMBL PNP finder (80.78 versus 80.88). Otherwise performance is significantly worse (80.38, $p < 0.001$, $t = 4.462$). A similar effect holds for the combined tagger/chunker (which achieves 92.98% accuracy on the combined tags and an F_β of 91.33 on chunks alone). Relation finder performance is better when trained on predicted tags/chunks and PNP chunks than when trained on treebank material (72.59 versus 71.38, respectively 72.59 versus 70.91 with the regular expression PNP finder).

6.1.5 Summary

The previous sections demonstrated two points. First, performance on a given task does not need to suffer and might even improve if the learner is trained to perform a more difficult task. We saw this effect when adding the non-local relations to the local ones and when training on the full task (with postprocessing) instead of on the application-specific ones. Second, performance is better if training and test material are as similar as possible. We saw this effect with the WSJ/Brown experiments and with automatically tagged and chunked versus treebank material.

6.2 Comparisons

As we saw in Sections 2.4.3 and 2.4.4, there are a few other systems that extract general grammatical relations (i.e. more than subjects and objects). One of these systems, using Transformation-Based Learning (TBL), is described in Ferro, Vilain, and Yeh (1999). Yeh (2000a) compares this system to the relation finder of Buchholz, Veenstra, and Daelemans (1999) on the very small, manually chunked training and test set of Ferro, Vilain, and Yeh (1999). In that comparison the memory-based relation finder does not perform very well: F_β is 60 whereas the TBL system scores 71. Part of this difference is due to extra information (passive and VP type) in the TBL system which was not incorporated into the MBL system at that time but some part of the difference must also be attributed to the different learners. Note however that MVDM and larger values of k which proved advantageous in our current system were not used back then. However it is unsure whether they would have had a positive effect given the small training set (3299 words). Yeh (2000a) notes that if he "had been trying to compare the two systems on a large annotated training set, the MBL system would do better by default just because the TBL system would take too long to process a large training set." By contrast the two systems with which we compare the Memory-Based Shallow Parser in this section can either handle large training sets (MBSL) or do not need to be trained at all (Carroll and Briscoe). All TiMBL experiments reported in this section were performed with TiMBL version 4.2.

6.2.1 Memory-Based Sequence Learning

MBSL (see Argamon, Dagan, and Krymolowski (1998), Dagan and Krymolowski (2002) and Sections 2.4.1.1 and 2.4.1.3) has also been applied to the task of finding unlabeled dependencies (Krymolowski and Dagan, 2002). Krymolowski and Dagan's definition of dependencies encompasses each local head-dependent relation that is listed in the intermediate format (cf. Section 3.1.2) excluding relations that involve punctuation. In cases of multiple heads (due to coordination), only the first head is considered. A target PoS sequence extends between the head and the dependent. In our Memory-Based Shallow Parser we distinguish between relations within chunks (e.g. in the NP chunk "Pierre Vinken" the relation from "Pierre" to "Vinken") and relations between chunks. The former are found by the chunker, the latter by the relation finder. In MBSL there is no such distinction. Instead it is possible to extract e.g. only dependents of verbs, only right dependents of nouns, or only dependents which are nouns. In the most general setting, which we use here, dependencies between all types of words and in both directions are extracted. In Krymolowski and Dagan (2002) this is denoted as `*←*, *→*`, where the asterisk matches any PoS and the arrow points to the head.[8]

As data sets we take the by now standard parsing division of WSJ sections 02–21 for training and 23 for testing. We use the treebank tags, in order to abstract away from the influence of any particular PoS tagger. We automatically chunk the text and find PNPs as outlined in Section 6.1.4. Finally we apply the relation finder to the combined output of chunker and PNP finder. F_β of the chunker, PNP finder and relation finder on the test set are 94.46, 93.07 and 77.56 respectively.

We record an unlabeled dependency for

- each chunk head and some other word from the same chunk, e.g. for the chunk "a nonexecutive director" we would record dependencies `nonexecutive→director` and `a→director`.
- each preposition and the head of the NP chunk that were joined by the PNP finder, e.g. `as←director`.
- each verb chunk head and focus chunk head that were predicted to have a *local* relation by the relation finder, e.g. `Vinken→join`.
- the words "not" and "n't" if they occur outside any chunk to the immediately preceding word. This simple heuristic catches most of the cases like `was←n't` and is necessary because the relation finder will not predict relations for chunkless words.

[8] MBSL settings are: a threshold of $\theta_T = 0.5$, left and right context of 2 words, at most one embedded instance in a tile, with a recursion level $r = 1$ and a maximal tile length of 5. Many thanks to Yuval Krymolowski for running the experiment.

Method	VB*←*, *→VB*			*←*, *→*		
	precision	recall	F_β	precision	recall	F_β
MBSL	87.4	73.5	79.9	85.8	81.7	83.7
MBSP	91.8	85.4	88.5	93.6	76.7	84.3

Table 6.11: Performance of Memory-Based Sequence Learning and the Memory-Based Shallow Parser on section 23 when evaluating unlabeled relations to verbs only, and to all words.

span	#	MBSL			MBSP		
		precision	recall	F_β	precision	recall	F_β
2	6581	92.2	95.4	93.8	96.3	95.7	96.0
3	3443	85.9	87.6	86.8	93.9	92.2	93.1
4	1958	84.8	76.9	80.7	92.1	87.7	89.9
5	1062	79.8	57.0	66.5	88.2	78.8	83.2
6	655	75.6	44.1	55.7	84.2	71.6	77.3
7	499	76.5	30.0	43.1	85.0	69.3	76.3
8	381	74.7	22.5	34.6	82.0	64.5	72.2
9	286	60.2	14.3	23.1	81.5	57.3	67.3
10	255	69.4	9.80	17.1	81.5	60.7	69.6

Table 6.12: Breakdown of performance by number of words spanned when evaluating unlabeled relations to verbs only. 2 means the relation connects two adjacent words. The second column shows how many instances of a given length there are. Relations spanning more than ten words are not shown (1265 instances).

As MBSP finds only relations to verb chunks, we first compared performance of MBSP and MBSL on a subset of all relations. This subset is denoted as VB*←*, *→VB*, i.e. relations to words tagged as verbs. Results are shown in the left half of Table 6.11.[9] We see that MBSP performs much better. This might be explained by the following facts:

- MBSP uses lexical information whereas MBSL is based on PoS only.
- TiMBL gives different weights to different features whereas all the elements of a sequence have equal influence in MBSL.
- MBSP splits the task into chunking and relation finding and lets the relation finder work on the simplified output of the chunker (chunks reduced to headword, head's PoS and chunk type). This reduction steps allows it to find relations over longer distances. MBSL takes only limited advantage of previously found structure as there is a limit on the maximal number of embedded instances that can be used.

A breakdown of performance by the number of words a relation spans (see Table 6.12) confirms the last point: the performance difference between MBSL and MBSP is much

[9]Figures are computed by the evaluation module integrated into MBSL.

bigger on longer relations. To get an idea of how far we are still away from the task of finding *all* dependencies, we also evaluate the systems' output on the complete set of relations `*←*, *→*`. Note that this set includes relations that our system does not even attempt to find, e.g. `Vinken←old` and `years→old` in our example sentence, or PPs attached to NPs, or VPs attached to SBARs (complementizers). This explains the much lower recall (see right half of Table 6.11). Precision is higher than for verbs because the complete set contains more "easy" cases (e.g. inside NPs). Thanks to the high precision overall performance of MBSP on this set is even higher than MBSL's.

6.2.2 Carroll and Briscoe

The GR extraction system of Carroll, Minnen, and Briscoe (1998) was already described in Section 2.4.3.2. The results reported in this section are based on the latest version, which also uses a list of phrasal verbs as additional lexical information.[10] C&B's test set consists of 500 sentences from the SUSANNE corpus (Sampson, 1995), which is a subset of the Brown corpus.[11] The sentences were chosen randomly from among the sentences for which the parser produces at least one complete parse (about 80%). GRs were first annotated automatically, then corrected manually. 236 of the test sentences also occur in the Brown corpus part of the Penn Treebank.[12] Apart from these sentences we take the complete Brown and WSJ treebank material for training (73,215 sentences, 1,628,672 tokens). We tag and chunk the test set with a combined tagger/chunker.[13] We then perform tenfold cross validation on the tagger/chunker and PNP finder training sets. The output of these experiments provides realistic training material for the PNP and relation finder. As the Brown corpus annotation does not use the `-CLR` function tag, we first train a relation finder on all material with this tag deleted and classify the test data. We then retrain the

[10] Many thanks to John Carroll and Ted Briscoe for sending the system's output on the test set and for explaining the details.

[11] Raw and annotated files, a description of the annotation, and evaluation software are available at `http://www.cogs.susx.ac.uk/lab/nlp/carroll/greval.html`.

[12] The complete part-of-speech tagged Brown corpus can be found on the Treebank CD-ROM. However only the genres F, G, K, L, M, N, P and R are also parsed. Genres A, G, J and N are represented in the SUSANNE corpus. Test sentences are from genres A, G and J only. This means that we only have parse tree annotations for the sentences of the G genre (belles lettres, biography, memoirs). The process of locating the test sentences in the treebank uncovered some differences between the two annotations which might influence performance slightly. For example the Treebank files sometimes contain quotation marks, hyphens or duplicate question marks or semi-colons that are not present in the test set. The test set sometimes contains tags in angle brackets (e.g. <bminhd>) that are not present in the Treebank. Treebank and test set encode formulas/symbols differently. Some phrases occur in the test set completely in capital letters whereas only the first letters are capitalized in the Treebank. Some phrases, which seem to be minor headlines, are not clearly delimited from the following sentence in the tagged Treebank files whereas they do not occur in the test set. Some words differ in the two corpora (*infield* versus *infielder*, *3-to-o* versus *3 to 0*, *Hovdingar* versus *h<ouml>vdingar*, 12 versus XII).

[13] See Section 6.1.4. Settings: `-p ddwfa -P pdFasss -O"-a0 -mM -w1 -k3 -vS"`

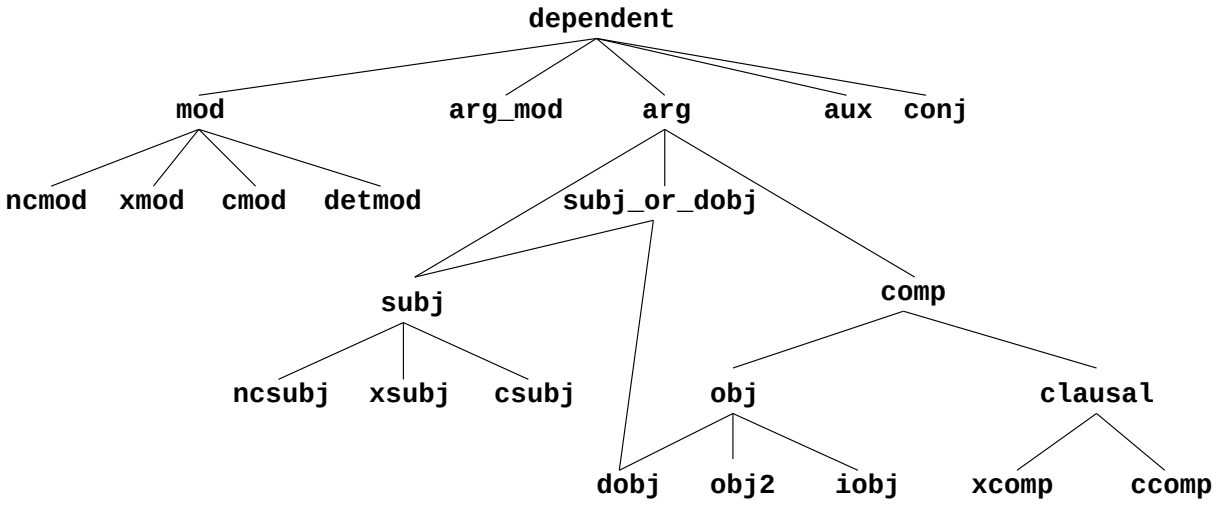

Figure 6.3: The hierarchy of grammatical relations used by Carroll and Briscoe

relation finder on the WSJ material only and reclassify the instances that were classified as PP (with or without function tags) in the first pass.

The most difficult part of the comparison is the mapping from our relations to the ones used by C&B. They use 15 basic GRs, which are organized in a subsumption hierarchy (see Figure 6.3). The GRs are explained in Carroll et al. (1997),[14] here we give only a summary. Modifiers (`mod`), i.e. adjuncts, are differentiated into non-clausal (`ncmod`), open (`xmod`) and closed (`cmod`) modifiers. Closed modifiers contain their own subject (e.g. "because he laughed") whereas the subject of open modifiers is unexpressed and can be controlled (e.g. "while laughing"). The same distinction holds for subjects (`subj`) and complements (`comp`). Object (`obj`) subsumes direct object (`dobj`), second NP object (`obj2`) and indirect object (introduced by a preposition; `iobj`). `Arg_mod` is used for the *by*-phrase expressing the logical subject in passives.

A GR annotation includes at least the name of the relation and the lemmatized head and dependent. The `subj, arg_mod` and `dobj` relations also specify the deep relation if appropriate (`obj` and `subj` respectively with passives, or `iobj` after dative shift) in the `initial_gr` slot. The `mod, arg_mod, iobj` and `clausal` relations also specify the preposition or complementizer which is the syntactic head of the dependent, if present, in the `type` slot. Carroll and Briscoe (2001) introduce three more relations. `Aux` is the relation between an auxiliary verb and the following verb, `detmod` holds between a determiner and the head noun and `conj` describes the relation between two conjuncts in coordination (e.g. `conj(and,Peter,Mary)` for "Peter and Mary").

Like MBSL, Carroll and Briscoe do not differentiate between intra-chunk and inter-chunk relations so again we have to construct relations from the output of the chunker, the PNP finder and the relation finder. Chunks are mapped as follows:

[14] An updated version is available at `http://www.cogs.susx.ac.uk/lab/nlp/carroll/greval.html`

- Anything tagged `DT, WDT, PRP$` or `WP$` inside a chunk is taken to have a `detmod` relation to the chunk's head.
- Any word form of *be, do* and *have* inside a chunk is taken to have an `aux` relation to the following verb in the same chunk (if one exists).
- If there is a sequence of two main verbs in a chunk and the second is a present participle or a `to`-infinitive, an `xcomp` relation is predicted between them with the first one being the head. Any predicted subject relation is copied from the dependent to the head. For the sentence "[Peter] [wants to leave]" for example, our relation finder would predict a relation `ncsubj(leave,Peter,_)`. The copy rule then adds `ncsubj(want,Peter,_)`. Any predicted preceding modifier relation is moved from the dependent to the head.
- Anything tagged `RB*` (adverb) inside a chunk is taken to have a `ncmod` relation to the following word in the same chunk (if one exists).
- If there is a word tagged `POS` (possessive marker) in a chunk a `ncmod` relation is predicted between the chunk's head and the word preceding the `POS` (e.g. "[Peter] ['s big house]" gives rise to `ncmod(poss,house,Peter)`).
- If there is a coordinating conjunction (CC) in a chunk, a `conj` relation is predicted between the preceding word (if any) and the closest following word (if any) within the same chunk that has the same major part-of-speech. Any relation that one of the conjuncts has is also copied to the other one. For the sentence "[Peter and Mary] [leave]" for example, our relation finder would predict a relation `ncsubj(leave,Mary,_)`. The copy rule then adds `ncsubj(leave,Peter,_)`.
- Anything else in a chunk is taken to have a `ncmod` relation to the chunk's head.

Relations are mapped according to the following rules:

- Predicative XPs and non-finite VPs[15] map to open dependents, finite VPs and SBARs to closed dependents and all other categories to non-clausal dependents.
- Anything with the `-SBJ` function tag maps to a relation under `subj`. If there is also a non-local `-OBJ` relation (due to a passive) the `initial_gr` is `obj`.
- Anything with the `-OBJ` or `-PRD` function tag maps to a relation under `comp`.[16]
- NPs with the `-OBJ` or `-CLR` (closely related) function tag map to `dobj`. A second `NP-OBJ` is mapped to `obj2`.

[15] The (non-)finiteness can be derived from the VP-type feature.

[16] There is one exception for the relation `VP-OBJ` which mostly encodes VP coordination and thus maps to `conj`.

	precision	recall	F_β
C&B	76.4	77.4	76.9
MBSP	81.8	66.0	73.0

Table 6.13: Overall performance of Carroll and Briscoe's GR extraction and of the Memory-Based Shallow Parser on Carroll and Briscoe's test set. All differences are significant.

- PPs with the `-CLR, -DTV,` or `-PUT` function tags map to `iobj`. The semantic head fills the `head` slot while the syntactic head fills the `type` slot.
- `PP-LGS` maps to `arg_mod` with `initial_gr subj`.
- Anything else maps to a relation under `mod`.
- Local and non-local relations map to the same GR (except for passives).

A special conversion problem occurs with subclauses with complementizers, and prepositions whose dependent is not an NP (e.g. *by laughing* or *until after midnight*). Our relation finder determines only the relation between the matrix verb and the complementizer/preposition but not between the complementizer/preposition and its dependent. In C&B's annotation however the relation is between the verb and the dependent. We therefore take the closest following NP or PNP (for prepositions) or VP (for prepositions and complementizers) that does not have a relation to any head as the dependent. If such an element does not exist the dependent is left unspecified.

In C&B's annotation, relations hold between lemmas and also the `type` slot is frequently filled by a lemma. Some lemmas may be multi-word (e.g. `according_to`). As we do not have a lemmatizer we map word forms to lemmas after parsing, using the lemmatized version of the test text, which is available together with the "raw" version. Once our output is in the required format we use C&B's evaluation software. In general a relation in the parser output and in the gold standard match if they have the same head and dependent and if the relations are identical or at most one subsumption level apart (e.g. a predicted `mod` matches with a gold standard `ncmod, xmod` or `cmod`). If present the `initial_gr` and `type` slot also have to match.

Overall results of the comparison are shown in Table 6.13. Following Carroll, Minnen, and Briscoe (1998), we computed significance with a paired t-test.[17] As with the general task in the MBSL comparison, MBSP has lower recall ($p < 0.001$, $t = 13.8263$) because it misses many relations to non-verbs, but higher precision ($p < 0.001$, $t = 5.50214$). However in this case overall performance of MBSP is lower than C&B's system ($p < 0.001$, $t = 5.93221$). The performance of our system on C&B's test set depends on many factors: the annotation scheme of the treebank, annotation errors in the treebank, our definition of heads, the performance of the memory-based modules and the mapping from their output

[17] But see Charniak (1995) and Yeh (2000b) for discussions of problems.

Relation	# occ.	C&B			MBSP			Treebank		
		prec.	rec.	F_β	prec.	rec.	F_β	prec.	rec.	F_β
dependent	2920	76.4	**77.5**	**76.9**	**83.2**	65.3	73.1	84.5	70.7	77.0
mod	1691	74.4	**77.4**	**75.9**	**80.8**	62.2	70.3	79.9	68.7	73.9
ncmod	1004	73.0	**72.4**	**72.7**	70.9	52.0	60.0	69.8	60.2	64.7
xmod	56	72.7	**57.1**	**64.0**	66.6	7.1	12.9	64.7	19.6	30.1
cmod	94	62.3	**51.0**	**56.1**	61.1	23.4	33.8	67.3	32.9	44.2
detmod	526	95.5	94.3	94.9	96.3	94.4	95.3	98.4	96.0	97.2
arg_mod	13	100.0	61.5	76.1	100.0	61.5	76.1	100.0	100.0	100.0
arg	969	78.5	**77.6**	**78.0**	84.4	70.0	76.5	90.4	74.3	81.5
subj	483	81.8	**82.4**	**82.1**	87.1	72.8	79.3	92.2	83.4	87.6
ncsubj	477	83.4	**82.6**	**83.0**	87.4	73.1	79.6	92.5	83.6	87.8
xsubj	4	100.0	75.0	85.7	100.0	50.0	66.6	100.0	50.0	66.6
csubj	1	0.0	0.0	0.0	0.0	0.0	0.0	33.3	100.0	50.0
comp	486	75.0	**72.8**	73.9	81.7	67.2	73.8	88.3	65.2	75.0
obj	286	75.1	68.5	71.6	**84.5**	65.0	73.5	91.8	*55.2*	*69.0*
dobj	179	84.8	81.5	83.1	89.6	77.6	83.2	91.5	84.9	88.1
obj2	8	58.3	87.5	70.0	66.6	50.0	57.1	100.0	50.0	66.6
iobj	99	55.8	43.4	48.8	72.8	43.4	54.4	100.0	*2.0*	*3.9*
clausal	200	74.8	**79.0**	76.8	78.3	70.5	74.2	85.0	79.5	82.1
xcomp	172	81.1	80.2	80.7	81.7	72.6	76.9	90.3	81.4	85.6
ccomp	28	83.3	**71.4**	**76.9**	59.2	57.1	58.1	59.3	67.8	63.3
aux	160	88.6	83.1	85.8	95.1	85.6	90.1	97.2	86.8	91.7
conj	87	69.7	**68.9**	**69.3**	**90.9**	34.4	50.0	82.0	36.7	50.7

Table 6.14: Number of occurrences (# occ.) of each GR in the 236 test sentences for which we have parse trees from the Penn Treebank. Performance of Carroll and Briscoe's GR extraction (C&B), of the Memory-Based Shallow Parser (MBSP), and of direct mapping from the treebank on these sentences. Figures in bold face mark the significantly better result of the comparison between MBSP and C&B; figures in italics mark cases where the treebank mapping performs significantly worse than MBSP.

to the test set relations. To get a better idea of how many of the errors of the system are due to the modules, we also compute our system's performance on only those 236 sentences from the test set for which we also have parse tree annotations from the Penn Treebank.[18] Next instead of mapping the predicted chunks and relations of verbs, we map those that we extracted directly from the treebank (as if for making training material).

Table 6.14 shows a breakdown of performance by relation. Figures for relations higher in the hierarchy sum up performance on all the relations they subsume. Thus performance on `dependent` is overall performance. For both systems it is very similar to performance on the whole test set, so these 236 sentences seem to be representative. On some relations MBSP performs (insignificantly) better than C&B's system. These include `detmod` and `aux`, two relations that only occur within chunks. This is in line with Li and Roth (2001) who showed that a specialized chunker is better at finding chunks than a full parser. MBSP is also (insignificantly) better on objects, especially indirect objects. The reason is

[18] On these 236 sentences F_β of the relation finder when comparing to the treebank annotations is 73.47.

probably that MBSP uses information about the identity of the preposition. Performance on clausal complements, especially `ccomp`, is lower. This might be due to the imperfect way of determining the dependent of complementizers. The largest performance difference occurs with clausal modifiers. Few of these are recalled by MBSP. However this is for the most part not the fault of the memory-based modules as the low result of the treebank-based predictions shows. Many missing `cmod` are reduced relative clauses. As they attach to nouns they are not recalled. In addition the treebank does not annotate explicitly that the noun is the subject or object of the reduced relative clause (cf. Section 3.1.2.6), which results in more missed relations. The treebank's low recall on `iobj` is caused by the missing `-CLR` tag in the Brown corpus (cf. Section 6.1.3). Thus all prepositional objects get mapped to `ncmod`, which also decreases precision there. In general these results show that the mapping from one annotation scheme to another is full of caveats. Given this large handicap the performance of MBSP does not look too bad, especially if one considers the fact that the test sentences were deliberately chosen to be *in coverage* for C&B's system. On a fully random test set their system's performance would presumably be lower (depending on how successful extraction of GRs from partial parses is).

Riezler et al. (2002) describe a combination of an LFG grammar, a full parser for producing possible parses and a discriminative estimation technique for ranking those. The system has been tested on 700 WSJ sentences annotated with f-structures (cf. Section 2.1.3) and also on C&B's test set. Crouch et al. (2002) describe in detail the problems encountered during mapping from f-structures to C&B's annotation. Riezler et al. (2002) report an F_β of 74.0 (based on labeled precision and recall) on C&B's test set.

6.3 Summary

This chapter showed how the relation finder that we developed in the previous chapters fits into the bigger picture of the Memory-Based Shallow Parser, how it can best be integrated and how MBSP compares to other approaches for finding GRs.

Chapter 7

Application: question answering

In this chapter we show how the Memory-Based Shallow Parser can be applied to the task of open-domain question answering (QA). Our goal is not to implement a fully-fledged QA system but to investigate how the relation finder (and the other modules of MBSP) can contribute to a real-world application. In the general QA task a system is given a natural language question and has to return a natural language answer. For limited domains, the task might be approached by mapping the question to a database query that extracts the necessary information from a database which models the information in this domain. Query results can then be presented using predefined patterns, and domain knowledge can guide the mapping. For open-domain QA, by contrast, exhaustive domain knowledge or databases are not available, and answer patterns cannot be predefined. Instead the answer is directly extracted from natural language texts. The set of all texts from which the system can extract answers is called the *document collection*.

Moldovan et al. (2000) define five classes of difficulty of QA on document collections (see Table 7.1). The difficulty does not only depend on the question but also on how the answer is contained in the document collection. Our QA system can only answer class 1 questions. Many current QA systems include strategies for answering class 2 questions, e.g. by using WordNet (Miller et al., 1990), see (Harabagiu et al., 2001), and some attempts have been made at tackling class 3 (see the "list task" below).

In general the larger the document collection, the higher the chance that it contains answers of the class 1 kind. The largest document collection that exists is probably the World Wide Web (WWW). We implemented a prototype of a QA system that tries to answer questions on the basis of the WWW. This system is called Shapaqa (shallow parsing for question answering). There are two variants of Shapaqa using two different WWW search engines. We also took part in the QA track of TREC-10 (Text REtrieval Conference). The set-up of these yearly QA competitions is described in more detail in Section 7.1. TREC QA tracks use a fixed document collection and have specific requirements on the format of answers. We therefore implemented another version of Shapaqa that fulfils these constraints. This version also has two variants. Both versions and all theirs variants

Class	KB	Reasoning	NLP/Indexing	Examples and Comments
1	dictionaries	simple heuristics, pattern matching	complex noun, apposition, simple semantics, keyword indexing	Q33: *What is the largest city in Germany?* A: ... *Berlin, the largest city in Germany* ... Answer is: simple datum or list of items found verbatim in a sentence or paragraph.
2	ontologies	low level	verb nominalization, semantics, coherence, discourse	Q198: *How did Socrates die?* A: ... *Socrates poisoned himself* ... Answer is contained in multiple sentences, scattered throughout a document.
3	very large knowledge-base	medium level	advanced NLP, semantic indexing	Q: *What are the arguments for and against prayer in school?* Answer across several texts.
4	domain KA and classification, HPKB	high level		*Q: Should Fed raise interest rates at their next meeting?* Answer across large number of documents, domain specific knowledge acquired automatically.
5	world knowledge	very high level, special purpose		*Q: What should be the US foreign policy in the Balkans now?* Answer is a solution to a complex, possibly developing scenario.

Table 7.1: The taxonomy of QA systems reproduced from Moldovan et al. (2000). KB: knowledgebase, KA: knowledge acquisition, HPKB: high performance knowledge bases.

share a common core, which is introduced in Section 7.2. Section 7.3 describes the online system and Section 7.4 the TREC system. Most of the content of these sections has been published before in Buchholz and Daelemans (2001b) and Buchholz (2002). Section 7.5 discusses related research and Section 7.6 summarizes our approach and gives an outlook on future research.

7.1 Text REtrieval Conference QA tracks

The TREC QA tracks have given a boost to QA research and the development of actual QA systems. Many groups working in the field have taken part and documented their approaches in the proceedings.[1] The tracks have also influenced QA evaluation methods and the definition of the task. As we largely adopt their task definition and evaluation method for our system, we describe the tracks briefly in this section. The QA track of TREC started in 1999 with TREC-8 and has been carried out each year since. The basic set-up is as follows:

[1] `http://trec.nist.gov/pubs.html`

894 How far is it from Denver to Aspen?
895 What county is Modesto, California in?
896 Who was Galileo?
897 What is an atom?
898 When did Hawaii become a state?
899 How tall is the Sears Building?
900 George Bush purchased a small interest in which baseball team?
901 What is Australia's national flower?
902 Why does the moon turn orange?
903 What is autism?

Figure 7.1: Some questions with original numbers from the TREC-10 QA track.

- Systems receive a list of several hundred natural language questions. The questions are fact-based, thus excluding questions such as "Who is the best actor in the world?". Each question consists of just one sentence. Most of them are *wh*-questions. Occasionally they are not formulated as questions (e.g. "Name a food high in zinc"). Some examples from TREC-10 are shown in Figure 7.1.

- There is a fixed document collection, mostly consisting of newspaper/newswire text.

- Systems have to answer the questions completely automatically. Using external resources such as dictionaries, ontologies or, indeed, the WWW is allowed.

- For each question a system can submit up to five ranked answer strings. Together with each string it must specify which document of the collection the answer is based upon.

- The assessors judge each answer in the context of the associated document. This means that even if some document states wrong information answering a question and a system extracts this answer from this document, the answer will be judged correct.

- If the first correct answer is at rank x the system receives $1/x$ points for this question. If the system found no correct answer it receives zero points. The total score of a system is the average over the scores for all questions. This evaluation metric is called Mean Reciprocal Rank (MRR).

Some aspects of the QA track have changed over the years. In TREC-8 some questions came from the logs of the FAQ Finder system (Burke et al., 1997) but most were created explicitly for the track by either the assessors or the participants. This resulted sometimes in unnatural formulations and often in questions that used the same wordings as the answer in the document, which is unrealistic. In later tracks, questions were either directly taken

from logs or, if a question in the log was not grammatical, formulated without looking at the documents.

In TREC-8 and 9 there was a 50-byte and a 250-byte task. The questions were the same but in the latter task answer strings could be longer. An answer string did not actually have to *be* the answer, it was only required to *contain* an answer.[2] In TREC-10 only the 50-byte task was kept. In TREC-11 answer strings are required to be the *exact answer*, i.e. nothing more or less than the answer.

From TREC-9 onwards an answer that was actually correct but not supported by the associated document was judged "unsupported".[3] This distinction yields two scores: a *strict* one in which unsupported is considered incorrect and a *lenient* one in which unsupported is considered correct.

In TREC-8 and TREC-9 the organizers guaranteed that the document collection contained at least one at most 50-byte answer to each question. From TREC-10 onwards this condition was dropped. Systems could then also state that there was *no answer* in the collection. If there was indeed no answer,[4] the system got points for this reply. Some questions in TREC-9 were variants of other questions.

In TREC-10 there was an additional "list task". Questions explicitly asked for a number of items, e.g. "Name 20 countries that produce coffee". Systems had to return an unranked list of the specified number of answers. Typically the items had to be extracted from different sentences or different documents. This is a first step towards general class 3 QA. Submitted answer lists are evaluated using item precision and recall. This means that systems have to identify actual answer items, not just strings containing answers, because they have to decide whether two sentences contain the same item or two different items. Otherwise the same item would be listed several times and others not at all, which would harm recall. Table 7.2 summarizes the QA tracks.

The questions and answers in the TREC QA tracks represent only part of the general QA problem. In particular questions as well as answers are short (one sentence; 50 bytes). One of the effects of this restriction is that questions often ask for entities (persons, locations, dates, measures, etc.) instead of for reasons, procedures etc. Linguistically entities are mostly expressed by phrases, and often by single chunks whereas the other categories are mostly expressed by clauses, sentences or paragraphs. This means that the QA task as currently defined in TREC is well suited for our system, which is based on chunks.

[2] If an answer string contains more than one entity that could potentially be an answer without any indication of which one should be taken it is judged incorrect.

[3] For example if the question is "Who is the 16th president of the United States?" and a system answers "Lincoln" but refers to a document that only mentions Lincoln but not the fact that he was president or which president he was, the answer is judged unsupported (Voorhees, 2001).

[4] At least neither the assessors nor the systems could find one.

	TREC-8	TREC-9	TREC-10	TREC-11
year	1999	2000	2001	2002
# questions	200	500 +193 variants	500	500
excluded	2	11	8	
sources of questions	FAQ Finder, assessors, participants	Encarta, Excite	MSNSearch, AskJeeves	MSNSearch, AskJeeves
# of documents	528,000	979,000	979,000	1,033,461
MB	1904	3033	3033	3130
answer length	50 or 250 byte	50 or 250 byte	50 byte	unrestricted
tasks	main	main; variants	main; list; context	main; list
new aspects	—	unsupported	"no answer"	exact answers
# participants	20	28	36	34
best MRR	0.66	0.58	0.68	
average MRR	0.29	0.22	0.25	

Table 7.2: Overview of the TREC QA tracks. TREC-11 evaluation is still going on; therefore not all facts are known. Best and average MRR are for the 50-byte task. Average is based on the best run from each group.

7.2 The core system

Most QA systems, including Shapaqa, implement at least the following steps:

1. Analyzing the question: what information is given and what information is asked for?
2. Information Retrieval: making a first selection of text fragments from the document collection on the basis of the output of the question analysis. Depending on the system, text fragments can be whole documents, paragraphs, sentences or other units.
3. Analyzing the fragments and extracting potential answers.
4. Sorting answers and making a final selection (if necessary).
5. Converting final answer(s) into the desired output format.

The part of Shapaqa that is shared by both versions and their variants implements the third and fourth step. This core is described in this section. The implementation of the other steps differs per version and variant and is therefore explained in later, specific sections. The core assumes that the question is already processed into an internal representation. Let us take the question "When was the telephone invented?" as our run-on

example. In the internal representation it looks like this: `VERB="invented" OBJ="the telephone" TMP="?"`. This specifies the main verb of the sentence, all its dependent phrases and the underlying grammatical relation between the verb and the phrase. It also specifies whether a dependent is *given* or asked for. The latter case is indicated by the question mark. As only the main verb is represented separately, an internally more complex question such as "Who was the first man to walk on the moon?" is represented as: `SBJ="?" VERB="was" OBJ="the first man to walk on the moon"`. In the internal representation, Shapaqa distinguishes between the relations `SBJ` (underlying NP subject), `OBJ` (underlying NP object), `TMP` (temporal), `LOC` (locative and directional), `PRP` (purpose and reason), `MNR` (manner), `EXT` (extension) and `OTH` (other relation between a PP and a verb). The last relation is parameterized for the preposition. In answer sentences there is also the relation `LGS` (underlying subject of passives) and the relations `SBJ` and `OBJ` refer to the surface subject and object. These relations are based on the Penn Treebank function tags.

In addition to the question in internal representation, the core also needs a list of sentences or partial sentences that possibly contain answers to the question. This list is provided by the second step. For each sentence from the list, the core applies the modules of the Memory-Based Shallow Parser (tagger, chunker, PNP finder, relation finder) and performs certain *tests*:

1. Are all the *given* phrases literally contained in the (partial) answer sentence?
2. Is the last word of each *given* phrase also the last word (i.e. head) of an appropriate chunk in the answer sentence? "Appropriate" chunk here means that the `VERB` must be the head of a verb chunk and the others must be the head of a non-verb chunk.
3. Does the first chunk of each *given* phrase have a relation to the `VERB` chunk in the sentence that matches the relation in the question? In active sentences relations match if they are identical (for `OTH` the preposition has to be identical, too), in passives `LGS` matches with `SBJ` in the question and `SBJ` matches with `OBJ`. If the verb is a form of "to be" `SBJ` also matches with `OBJ` and vice versa in order to match "Who is X?" with "X is the president of Y".
4. Is there a chunk in the sentence that has the same relation to the `VERB` chunk as the "?" in the internal question representation?

The precise order of the application of modules and tests varies per version of Shapaqa. Let us illustrate the tests with some sentences that possibly contain answers to our example question about the telephone.[5]

(1) The importance of *the telephone* network as a critical factor in the success of fax cannot be overstated.

[5]Note that the sentences were automatically parsed and the annotation contains errors. Integers on chunk labels indicate relations, e.g. `NP-SBJ-1` is the subject of `VP-1`.

(2) Alexander Bain *invented* the fax machine in ...

(3) [$_{NP}$ Touch-Map Systems] [$_{VP}$ *invented*] [$_{NP}$ *the telephone* dealer locator] [$_{P}$ over] [$_{NP}$ seventeen years] [$_{ADVP}$ ago] .

(4) [$_{VP-1}$ *Invented*] {$_{PNP-OTH-1}$ [$_{PP}$ at] [$_{NP}$ almost the same time] } {$_{PNP}$ [$_{PP}$ as] [$_{NP}$ *the telephone*] } {$_{PNP}$ [$_{PP}$ to] [$_{NP}$ speed data analysis] } {$_{PNP}$ [$_{PP}$ for] [$_{NP}$ the 1880] } U.S. [$_{NP}$ Census] , [$_{NP-SBJ-2}$ the tabulating machine] [$_{VP-2}$ was] [$_{NP-OBJ-2}$ an electromechanical device] [$_{NP}$ that] ...

(5) [$_{NP-SBJ-2}$ One year] [$_{SBAR}$ after] [$_{NP-SBJ-1}$ *the telephone*] [$_{VP-1}$ was *invented*] , [$_{NP-SBJ-2}$ its usage] [$_{VP-2}$ was taxed] .

(6) [$_{NP-SBJ-1}$ *The telephone*] [$_{VP-1}$ was *invented*] {$_{PNP-LGS-1}$ [$_{PP}$ by] [$_{NP}$ Alexander Graham Bell] } {$_{PNP-TMP-1}$ [$_{PP}$ in] [$_{NP}$ 1876] } .

Sentences (1) and (2), which are adjacent in the original document, already fail the first test as they do not contain all phrases. Sentence (3) fails the second test as *telephone* is not the head of a chunk. Sentence (4) fails the third test as there is no underlying object relation between *the telephone* and *invented*. Sentence (5) fails the fourth test as no chunk has a TMP relation to *invented*. Finally sentence (6) passes all tests. The chunk *in 1876* is the one whose relation matches that of the question mark phrase in the internal question representation. We will call it a *key chunk*. Its semantic head *1876* constitutes a *keyword* answer. Every time Shapaqa finds a key chunk in the list of potential answer sentences, it increments a *frequency* counter for the keyword and stores the key chunk and the sentence as *evidence* for the keyword answer. After all potential answer sentences have been processed by the core of Shapaqa, keyword answers are sorted by their associated frequency. The top of the sorted list then contains the keywords for which most evidence has been found. A special module demotes semantically empty keywords[6] to the bottom of the list. The output of Shapaqa's core is the sorted list of keywords, with each keyword associated with its frequency and linked to its evidence. Sorting by frequency usually demotes answers based on parser errors and on wrong information in documents and is a first step towards the construction of complex answers, i.e. answers that consist of several simple answers, for example: "Who was President of Costa Rica in 1994?" — "Calderón was president until 8th May, after that Figueres became president." (Buchholz and Daelemans, 2001a). At the moment both simple answers will appear as separate keywords in the list.

[6]one, I, you, he, she, it, we, they, me, him, her, us, them, this, that, these, those, here, now, then, there, who, whom, which, what, where, when, why

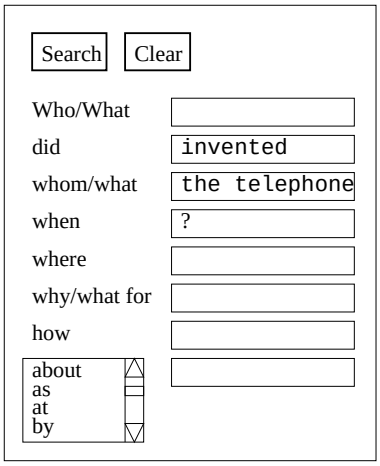

Figure 7.2: The HTML input form of the online version of Shapaqa.

7.2.1 Implementation

The implementation of the Memory-Based Shallow Parser that forms part of Shapaqa consists of four instances of memory-based learners running in server mode. This means that the learner is kept in RAM of the computer it runs on and constantly listens to one of the computer's ports. If an instance is sent to the port the learner classifies it and sends the classification back to the port. In addition to Shapaqa, an online MBSP demo[7] and an offline version of MBSP use the same servers through these port connections.

The PoS tagger software is a forerunner of MBT (Daelemans et al., 1996). The tagger was trained on the PoS tagged version of the Penn Treebank II WSJ Corpus.[8] The chunker, PNP finder and relation finder were trained on the parsed version of the Penn Treebank II WSJ Corpus (1,173,766 words, 49,208 sentences), converted into chunks and relations with an earlier version of `chunklink.pl` than that described in this thesis. All three use IGTree, with Gain Ratio feature weighting. Chunker instances look like shown in Table 6.8 on page 143. PNP finder instances are similar to the ones in Table 6.9 but use a context of two chunks to the left and no feature "number of intervening PP chunks". Relation finder instances are similar to the ones in Table 3.1 on page 72 (our preliminary feature selection) but use no feature "number of intervening VP chunks".

7.3 The online system

In the current version of the online prototype,[9] question analysis is left to the user. Phrases can be entered into HTML text boxes; entering a verb is obligatory. Figure 7.2 shows

[7] `http://ilk.kub.nl` under "Demos"
[8] Settings: IGTree, Gain Ratio, `-p ddfa, -P pdFasss.`
[9] `http://ilk.kub.nl/shapaqa/`

the relevant part of the interface with our example question entered. The lowest left box contains a list of prepositions from which the user can choose (for the OTHer relation). Once the user presses the "submit" button, the internal question representation is transformed into a url-encoded query[10] and sent to an internet search engine. As different search engines have different query syntax and return results in different formats the interface to each search engine has to be implemented explicitly. At the time that the prototype was designed interfaces to Google[11] and AltaVista[12] were implemented. Both engines have their own advantages and disadvantages.

In contrast to probably all other engines, Google does not only return URLs but also text snippets that normally contain the query words. This means that Shapaqa does not need to retrieve the original pages (which might take quite long) but can work on the text snippets instead. However the text snippets have a fixed length. If the query words do not occur close to each other in the document, Google creates a text snippet that contains (partial) sentences from different places in the original document, separated by ellipsis dots. Also if sentences are too long to fit into the text snippet, Google truncates them and marks this by dots. Figure 7.3 shows the top text snippets for the query on the telephone. The fourth one does not contain the word "invented". The fifth and sixth are composed of non-adjacent sentences. Shapaqa cuts the text snippets into (partial) sentences using a simple tokenizer and hands them to the core component together with the question in the internal representation. It also stores the original URL for each sentence. The ability of Shapaqa to work with the text snippets crucially depends on the fact that MBSP performs only local decisions and does not try to find a global parse (which probably does not exist for a partial sentence).

AltaVista does not return text snippets but lists of URLs of documents containing the query words. However it offers the possibility to restrict search to documents in which the query words occur NEAR to each other (defined as "within 10 words of each other").[13] They might still occur in different, but adjacent, sentences but chances of this are much lower than without the NEAR operator. This means that fewer documents need to be processed. Shapaqa retrieves the documents one by one, splits them into sentences and hands the sentences to the core component for processing. Again it stores the original URL for each sentence.

Even with the text snippet short-cut or the NEAR restriction many sentences have to be checked while the user is waiting for results. We therefore tried to implement Shapaqa in a

[10] The words in a phrase are required to be found in this exact sequence by enclosing them in quotation marks. Plus signs prevent Google from ignoring stop words (as we do not know exactly what Google considers a stop word we use plus signs on all words). The query is then `"+the +telephone" +invented`. Url-encoding means replacing spaces and other non-alphanumeric characters by special codes. The full URL is then something like `http://www.google.com/search?as_qt=w&as_eqt=w&as_eq=&as_dt=i&sitesearch=&lr=lang_en&start=0&num=10&btnG=Google+Search&as_lq=&as_q=%22%2Bthe+%2Btelephone%22+%2Binvented`

[11] `http://www.google.com`

[12] `http://www.altavista.com`

[13] see `http://help.altavista.com/adv_search/syntax`

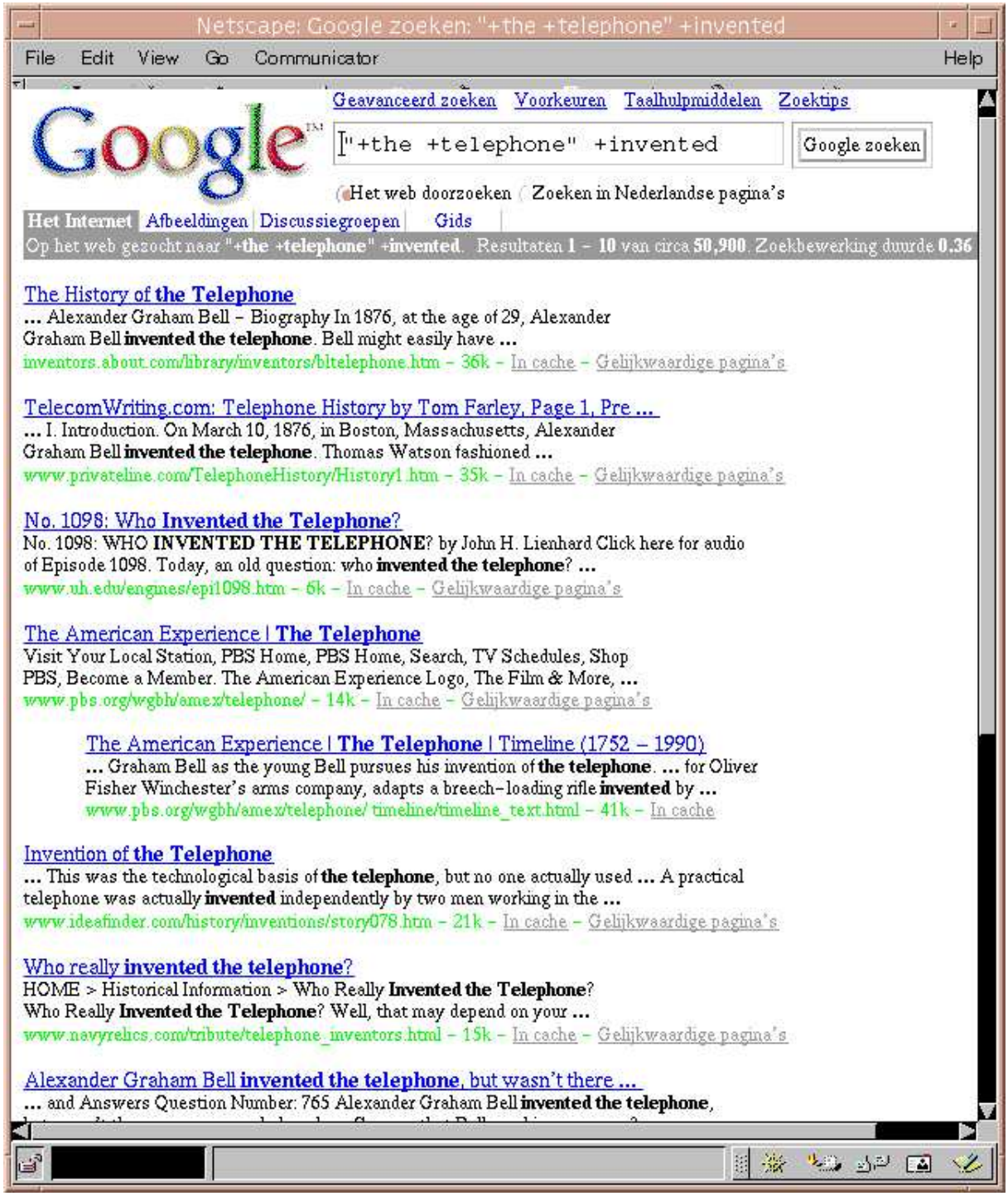

Figure 7.3: Google's top results for the query `"+the +telephone" +invented`

way that avoids unnecessary computation as much as possible. In particular we try to avoid unnecessary parsing steps. This is done by interleaving URL (i.e. text snippet/document)

processing with the tests explained earlier, in a way which we call *parsing on demand*, as depicted in Figure 7.4. During sentence tokenization the program already records which words from the given phrases it has encountered in the current sentence. When the tokenizer decides that it has reached the end of a sentence, Shapaqa applies the first test (all phrases in sentence?). If the test fails, the program does not process the sentence any further. Likewise it applies the second test directly after chunking, and if the test fails it does not call the relation finder for this sentence. Otherwise the relation finder has to classify one instance for each (pair of the verb chunk and a) given phrase. Thus even if the sentence contains several verbs, only dependents of the relevant verb are checked, and among them only those that are mentioned in the question. After each instance classification Shapaqa applies the third test (same relation?) and if the test fails it does not process the sentence any further. If the program extracts an answer from the sentence, it also does not process the rest of the text snippet/document any further as it is rather unlikely that it contains another different answer. The possibility to perform *parsing on demand* depends crucially on the architecture of MBSP in which all modules work in sequence and produce a deterministic output and on the implementation of the relation finder which classifies instances independently of each other. More unnecessary processing could be avoided if there were a search engine that implemented text snippets *and* the `NEAR` operator, or ideally an operator for restricting query words to appear in the same sentence.

Shapaqa's users can specify how many URLs (text snippets/documents) it must search. They can also indicate what kind of output they wish to see. In the default setting Shapaqa lists each URL as it processes it. It also outputs pairs of adjacent sentences that fail the first test or sentences that fail any other test together with a short failure message. This output is colored light grey. Successful sentences are printed in black with the given phrases and the key chunk highlighted in color. The idea of this presentation is to help users to focus on the successes while still allowing them to check the failures, but no further research was done into this or alternative presentations. After all URLs have been processed, keyword answers are sorted by frequency as described before and a summary is printed listing in order each keyword answer, its frequency and the evidence sentences together with their source URL. An example is reproduced in Figure 7.5, p. 178.

The online version of Shapaqa is implemented in PHP (version 4.1.2) and runs on our group's web server. At the time of writing the variant using Google is no longer available as Google does not allow unauthorized automatic querying.[14] However, a new variant using Yahoo![15] has been added since Yahoo! also displays text snippets.

7.3.1 Evaluation

The evaluation reported in this section was published earlier in Buchholz and Daelemans (2001b) and performed even earlier. Next to the version of Shapaqa described above, and

[14] `http://www.google.com/terms_of_service.html`
[15] `http://www.yahoo.com/`

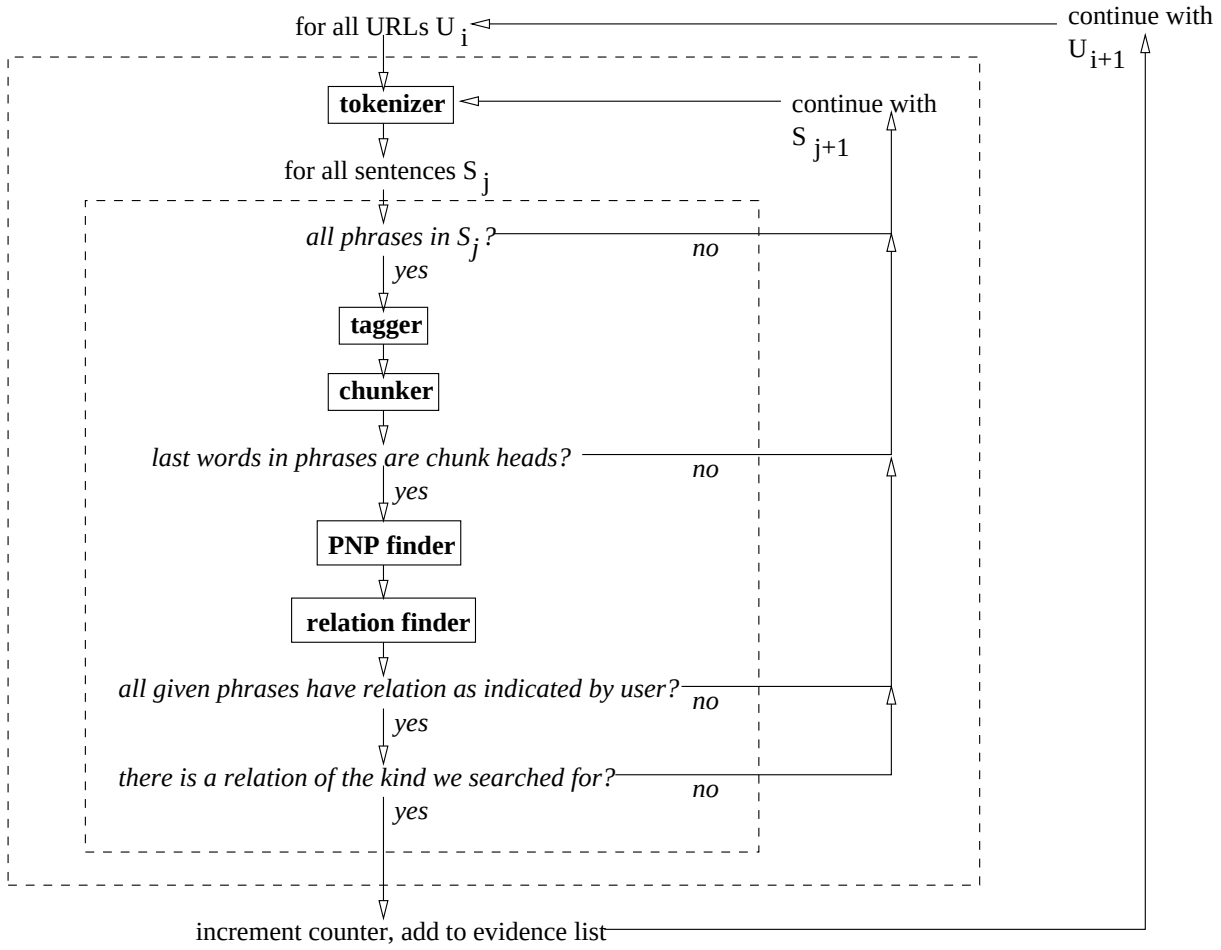

Figure 7.4: Implementation of *parsing on demand* in the online version of Shapaqa.

referred to here as Shapaqa GR, the paper also reports on another version that is less relevant for this thesis as it does not use the relation finder. Instead of assigning classes to pairs of chunks (verb and something else), it assigns classes to single chunks. We refer to it as Shapaqa CT (chunk type). Shapaqa CT does not distinguish the relational classes `SBJ, OBJ` and `LGS` but groups all of them under the class "`NP`". This class also contains any other NP chunk that does not fall under `TMP, LOC, PRP, MNR` or `EXT`, including NPs that do not depend on any verb, such as the "61 years" chunk in "Pierre Vinken, 61 years old, ...". Likewise temporal, locative etc. chunks that do not depend on a verb are also classified as `TMP, LOC`, etc. by Shapaqa CT. In Shapaqa GR chunks that do not depend on the verb have the "no relation" class.

For evaluation, we used the 200 questions from the TREC-8 QA track (Voorhees and Tice, 2000), see also Section 7.1. The first step was to manually convert the natural language questions into Shapaqa's internal question representation. While some questions have only one, very obvious representation (like our old telephone example), others have several. Thus in these cases, results may depend on the particular representation. We tried to

	Shapaqa GR	Shapaqa CT	SENT	Google
MRR	0.28	0.34	0.32	0.30
questions with answers	72	101	114	198
"precision" over answered questions	0.76	0.68	0.56	0.31
"precision" over 72 questions	0.76	0.73	0.56	0.47

Table 7.3: Results over the test questions: MRR, number of questions for which at least one answer is proposed, "precision" over all questions answered by a system, and over those answered among the 72 questions that were answered by Shapaqa GR.

choose a representation that we thought would be used by the average user (given the constraints of the HTML form). The following rules were used:

- Enter in active form, with the main verb as the only verb.
- Skip parts which, when left out, do not change the meaning, like "What is the population of Ulan Bator, *capital of Mongolia*?". This was necessary due to the sometimes unnatural phrasing of the TREC-8 questions.
- Skip verb particles such as "up" in "Who came up with the name, El Nino?".
- Represent questions with "What is the name of/Name the/How many/Which/What X" as if they were simple "who/what" questions (60 cases).[16]
- Questions with "How far/many times" etc. could not be represented, so Shapaqa did not score any points for them (12 cases).

We let Shapaqa answer the questions, and took the first evidence sentence of each of its top five keyword answers for judging. Two human judges then read the answers from top to bottom until they found a correct answer to the original question.[17] The rank of the correct answer was noted and the overall MRR computed. To put the results into perspective, it is necessary to compare them to other methods of finding answers on the internet. One such method is Google, which performs keyword search and returns text snippets. We entered all of the words in the internal representation of a test question as keywords into Google, and took the top five text snippets for judging. We also evaluated a variant of Shapaqa using only the most basic kind of NLP: the sentence tokenizer and the first test. If a sentence contained all of the given phrases, it was returned as an answer (this method is henceforth called SENT). Again, top five answers were judged. The results are shown in the first row of Table 7.3: Shapaqa CT performs better than SENT, and SENT is still better than Google. Shapaqa GR performs worst. However, MRR values do not differ dramatically.

The picture changes if we look at the "precision" of the systems: the total points received divided by the number of questions for which the system proposed at least one answer hypothesis. Table 7.3 shows that the higher the level of NLP used, the fewer questions a system tries to answer, but for these few, "precision" is higher. This is even true if we

[16] Respectively "when" and "where" for "in which year" etc. and "in what city".

[17] Judging largely followed the TREC QA guidelines (Voorhees and Tice, 2000).

compare precision of systems on only those 72 questions that Shapaqa GR tried to answer. This observation led us to the idea of a combined system: if Shapaqa GR returned any answers, we took these answers as those of the combined system. If not, and if Shapaqa CT returned answers, we took those, and so on down to plain Google. This combined system achieves an MRR of 0.46, which is substantially better than any of the individual systems. We conclude that this back-off architecture is an easy and successful way to combine approaches with different degrees of NLP and different "precision" values. Note however that only the two Shapaqa approaches identify the actual answer in the sentence (the key chunk) and therefore allow highlighting and frequency counts. Although the difference in precision between the GR and CT versions is not big, there are examples where the former is clearly useful. For the question "Who killed Lee Harvey Oswald?", GR put the correct answer ("Ruby") on top, while CT found "Kennedy" most frequently (and "JFK" third) as it cannot make the difference between subjects and objects, i.e. killers and victims.

7.4 The TREC-10 system

Our TREC-10 system does not have its own Information Retrieval engine. It uses the top ranked 1000 documents per question from the list provided by NIST, the TREC organizers. Following the insights from our evaluation of the online system we use Shapaqa together with a simple back-off method (henceforth called "baseline"). Shapaqa and the baseline component are integrated as follows: if Shapaqa returns at least one answer string, its top answer string is taken as the top ranked answer string of the combined system. All the remaining ranks are filled by the top answer strings returned by the baseline component. We describe the two components separately in the following two sections. The overall architecture is shown in Figure 7.6, p. 179.

7.4.1 Baseline component

The baseline component tokenizes and PoS tags the question. Then it extracts all words that the tagger marked as not occurring in its training material or that it tagged as noun, non-modal verb, adjective, particle, number, or foreign word.[18] A special case are non-subject questions with a form of the auxiliary *do* and following infinitive, e.g. "When *did* Elvis Presley *die*?". English grammar tells us that in a declarative sentence (i.e. the answer) it will be the main verb that carries the inflection: "Elvis Presley *died* in 1977". Therefore if the question contains "did" or "does", the baseline component replaces all following infinitival verb keywords by their past tense or third person singular present tense form, respectively. This rule applies to 22 and 14 questions, respectively, and uses the CELEX lexical database (Baayen, Piepenbrock, and van Rijn, 1993).

[18] Except for the words "much", "many", "name" and forms of "be", "have" and "do".

After keyword extraction, the baseline component tokenizes all top 1000 documents for each question, and extracts all sentences which contain at least one keyword for that question. This yields 1,233,692 sentences. For two questions no sentences could be extracted.[19] The component records the document ID and the number and position of the keywords together with each extracted sentence and sorts the sentences for each question according to the number of keywords in them.[20]

It returns the top sentences of the sorted list as answers. To trim them to 50 bytes we compute that byte position in the sentence that is at the center of all the keyword positions, and then take 50 bytes around it, possibly shifting the 50 byte window as far as necessary to the right or left for it not to extend beyond the sentence boundary.

As an example, consider the question "When was President Kennedy shot?". Keywords are *President, Kennedy* and *shot*. 5928 sentences are extracted. The topmost chosen answer sentence and its 50-byte string are "In 1963, he was riding in the motorcade wi*th President Kennedy when Kennedy was fatally shot* by Lee Harvey Oswald in Dallas" in which the answer falls outside the chosen 50 bytes.

On the non-variant questions of the TREC-9 data (cf. Section 7.1), taking the full top five sentences of the sorted list as answers yielded an MRR of 0.321 using the automatic evaluation.[21] After the sentences were trimmed to 50 bytes with the above method, MRR dropped to 0.125. This is hardly surprising, given that the method is completely insensitive to the type of answer the question is asking for (it returns the same 50 byte window, whether the question word is "who", "where", or "when" etc.) and that the complete answer sentences were on average 293 bytes long.

7.4.2 Shapaqa

The Shapaqa component processes the PoS tagged question further with the help of the other modules of MBSP. The relation finder does not work well on questions, especially in the first part, which shows the characteristic question syntax. This is probably due to the small number of direct questions in the WSJ training material. Therefore we developed a set of hand-made regular expressions and substitutions on the basis of the TREC-9 data and applied these to the question after parsing to fix the most common errors. For our previous example, the result would be "$[_{ADVP-TMP-1}$ When] was $[_{NP-SBJ-1}$ President Kennedy] $[_{VP-1}$ shot] ?".

[19] "What is pilates?" and "What is dianetics?". Keyword matching is sensitive to capitalization and both terms appear in the documents only with capital letters.

[20] Multiple occurrences of the same keyword are only counted once.

[21] During manual judgement of the track submissions, the TREC organizers created a file of regular expressions as a service to the community. The file can be used together with a scoring program to (roughly) score the output of a system. If a regular expression for a question (number) matches the output for that question, that output is scored correct.

Next Shapaqa transforms the parsed question into its internal representation. It rejoins prepositions stranded at the end of the question with their NP and converts passive to active. It then takes the head (i.e. the last word) of the first verb chunk as the `VERB` (auxiliaries in inverted questions should not be a verb chunk of their own). Finally it stores all the chunks that are dependent on the `VERB` together with their relation and concatenates each "verb-independent" chunk to the closest verb-dependent chunk to the left because it probably belongs to the same phrase. In this way it splits up the question into phrases that all have some relation to the central verb. It also replaces the phrase that contains the *wh*-word by the question mark. Our example would then look like: `VERB="shot" OBJ="President Kennedy" TMP="?"`.

Shapaqa treats questions asking for "Which/What X" as if they were simple "Who/What" questions and the phrase "In which state" as a simple "Where" because the relation finder assigned it a locative relation. In total, Shapaqa could map 60 "Which/What X" questions. It treats questions starting with "Name a X" as if they read "What is a X?" (which is an ad-hoc solution that did not work: both questions were incorrectly answered). In addition it performs some other minor simplifications during mapping. In total, it could convert 416 of 500 questions to its internal representation.

Next the Shapaqa component takes all sentences that contain all of the baseline component's keywords (in total 44,753 sentences) and lets them be parsed by an offline version of MBSP (using the same servers as the online Shapaqa version). It then applies the four tests to the parsed sentences and collects frequencies of keyword answers.

We used two variants of Shapaqa in our system, called Shapaqa-TREC and Shapaqa-WWW here. Shapaqa-TREC extracts answers directly from the TREC document collection in the way described above. One of the answer chunks whose head has the highest frequency is taken as the answer of the Shapaqa component. If necessary the answer is trimmed to 50 bytes by cutting off the end. For our example, the answer was extracted from the sentence: "President Kennedy was shot *on Nov. 22, 1963*.". The headword was found three times as head of an answer.

Shapaqa-WWW uses the online version of Shapaqa described earlier. Google returned results for 380 of the 416 questions that could be converted to Shapaqa's internal representation. In these Shapaqa found answers to 283 questions (7936 answers in total). For some questions, the only answers found are semantically empty ones like "he/who/somebody" etc., which are discarded. This leaves us with answers to 265 questions, which is slighty more than half of all TREC-10 questions. As explained above, answers are sorted according to the frequency of their headword. The most frequent headword is taken to be the preliminary answer. For our example, the headword *1963* was found twelve times.

To turn a preliminary answer from the WWW into a valid TREC answer string, we have to find a document in the TREC collection that contains the headword. To increase the chance that the document actually supports the answer (as is necessary to be judged correct under the strict evaluation), we look for the headword in the sorted sentences extracted for the question, starting with the ones that contain most of the baseline component's

keywords. Headwords could be found in these sentences for 226 questions. After a sentence is found, a 50 byte piece of it centered around the headword (unless shifted to meet sentence boundaries) is extracted as Shapaqa-WWW's answer string. For our example, answer sentence and string are: "Ruby shot Oswald to death with the .38-caliber Colt Cobra revolver in the basement of Dallas *City Jail on Nov. 24, 1963, two days after Presi*dent Kennedy was assassinated.", which unfortunately makes the answer invalid.

7.4.3 Runs: WWW+ and WWW−

We submitted two runs for the TREC QA track main task. Run WWW+ uses Shapaqa-WWW's answer (if present) on the first rank, Shapaqa-TREC's answer (if present) on the next highest rank, and the baseline component's answer on all other ranks. If not enough answers can be found to fill the five ranks, NIL (meaning that no answer exists in the document collection) is added.[22] WWW+ received a MRR of 0.210 (0.234 lenient). Run WWW− uses only Shapaqa-TREC (for the first rank) and the baseline component.[23] Its MRR is 0.122 (0.128 lenient).

After submitting the runs, we unfortunately noticed a bug in Shapaqa-TREC. The special rule that SBJ and OBJ of "to be" match was omitted in this implementation.[24] This means that for example for the frequent question type "What is a Y?" which asks for a definition or description, only sentences matched which read "X is a Y" and not those that read "A Y is X" (which is the way to formulate a definition). This led to many erroneous answers like "What is a prism?" — "Serbian agression", extracted from a sentence that reads "Serbian aggression is a prism through which we can see all sides of Europe".

As 254 of the questions that could be mapped to Shapaqa's internal representation have a form where the special matching rule could have been applied, the omission might have influenced Shapaqa-TREC's performance significantly. To find out how serious this effect is, we reran the WWW− run after fixing the bug and studied the differences. It turned out that only 50 questions were affected by the bug fix, so we compared these manually. In most of the cases, the new answer was as bad as the old one, so the score did not change. We estimate that the new version's MRR would be about .004 higher than the original one's, which seems negligible.

Table 7.4 displays the distribution of the rank of the first correct answer in both runs and shows that WWW+ has twice as many correct answers at the first rank than WWW−. Table 7.5 shows how often each of the three components contributed to the five answers for the WWW+ run before and after the bug fix. We see that even after the bug fix, Shapaqa-WWW contributes three times as many answers as Shapaqa-TREC (159+67 vs.

[22] Due to a bug, the NIL answer was not added at the lowest rank, as intended, but at the highest.

[23] In the original paper, WWW+ is called TilburgILKs and WWW− TilburgILK.

[24] Shapaqa-WWW is implemented in PHP and runs on our webserver, whereas Shapaqa-TREC is in Perl and runs on a machine whose hard disk can hold all the TREC documents for the QA track.

rank of first correct	1	2	3	4	5	none
WWW−	43	15	15	7	14	398
WWW+	87	19	9	7	11	359

Table 7.4: Distribution of rank of first correct answer in both submitted runs (strict evaluation)

components used	WWW+	after bug fix
none (only answer NIL given)	3	3
only baseline	267	263
Shapaqa-TREC and baseline	4	8
Shapaqa-WWW and baseline	195	159
Shapaqa-WWW, Shapaqa-TREC and baseline	31	67

Table 7.5: Number of questions for which different components contributed to the five answer strings.

8+67). This is probably due to the much larger document collection that Shapaqa-WWW works on.

Table 7.6 shows how the assessors judged each component's answer strings. We compute the *precision*[25] of a component as the percentage of correct answers among all the answers it contributed. We see that Shapaqa-WWW has a higher precision than Shapaqa-TREC, especially after the bug fix. Both versions of Shapaqa have a much higher precision than the simple baseline component. Precision of the baseline component's answer is slightly higher for its top ranked answer than for the lower ones, but in general there does not seem to be a clear correlation between its answers' ranks and their reliability. Shapaqa-WWW has more unsupported answers than the other components. This is clearly due to the mapping from WWW-answers to TREC documents.

Table 7.7 gives a breakdown of Shapaqa-WWW's precision on different types of questions. The first column shows the question type according to the module that maps questions to Shapaqa's internal representation. The second column indicates how many questions of this type could successfully be converted to Shapaqa's internal representation. The third column shows for how many questions Shapaqa-WWW returned an answer string. The fourth and fifth columns give the precision on these strings (strict and lenient). "When"-questions do best, followed by those with the *wh*-word inside a prepositional phrase.

Shapaqa relies on high-level NLP (chunking, grammatical relations) for finding answers. If it finds more than one answer however, it uses the frequencies of the answers' headwords to choose among the answers. The idea is that frequency correlates roughly with reliability of the answer. However, this can only work if the document collection from which answers are extracted is large enough. On average, Shapaqa-TREC (after the bug fix) finds 2.55 different answers (where "different" means having a different headword) for the questions

[25]Note that this is slightly different from the "precision" in Section 7.3.1 because there the rank of the correct answer entered into the formula.

judgment	baseline 1st	2nd	3rd	S-TREC	S-TREC*	S-WWW
incorrect	462	473	469	27	63	144
correct	34	22	26	8	12	71
unsupported	1	1	1	0	0	11
prec. strict	6.8	4.4	5.2	22.9	16.0	31.4
prec. lenient	7.0	4.6	5.4	22.9	16.0	36.3

Table 7.6: Judgments (under strict evaluation) and precision of answers (strict and lenient) of each component. Baseline 1st, 2nd and 3rd means the baseline's first, second and third answer. S-TREC is Shapaqa-TREC, S-TREC* is Shapaqa-TREC after the bug fix, S-WWW is Shapaqa-WWW.

Wh-type	# of questions	# of q. answered	prec. strict	prec. lenient
who	44	21	57.1	57.1
what	237	136	25.0	30.1
which X	5	2	0	0
what X	55	27	18.5	25.9
where	24	12	25.0	25.0
when	24	18	66.7	72.2
why	2	0	0	0
how	7	1	0	0
PP	16	9	55.6	66.7
Name a	2	0	0	0
Total	416	226	31.4	36.3

Table 7.7: Precision of Shapaqa-WWW on different *wh*-types.

it finds answers for at all. The average frequency of the most frequent answer for each question is 1.44, but the distribution is highly skewed in that most top answers have only frequency one, and only one top answer has a frequency higher than ten. By contrast, Shapaqa-WWW finds 17.0 different preliminary answers per question, and the average frequency of the most frequent answers is 26.7.

To further study the correlation between frequency and reliability, we divided the 265 questions for which Shapaqa-WWW found an preliminary answer into a high-frequency and a low-frequency group according to the frequency of the top answer (greater, or less or equal to 7). The precision of Shapaqa-WWW's answers to questions in the high-frequency group is 36.1% (39.3%) whereas it is only 26.0% (32.7%) for the low-frequency ones.

This shows that answers which we find often are more reliable than those with little evidence. As Shapaqa-WWW searches on a much larger document collection than Shapaqa-TREC, it can take advantage of this fact. This effect even holds despite the very simplistic way of mapping the WWW answers back to the TREC collection in order to comply with the TREC guidelines.

7.4.4 Error analysis

In this section, we study the effect of several design decisions we made about the document selection, the transformation from natural language questions to Shapaqa's internal representation and the mapping from WWW preliminary answers back to the TREC collection.

Our system does not search the whole document collection for answers but only the top 1000 documents per question (as provided by NIST). For 14 questions, some other systems found an answer in documents not in the top 1000, so some minor improvement should be possible through a better IR component.

Several things can go wrong during the transformation from natural language questions to Shapaqa's internal representation. First, if no central verb can be found, the transformation is not possible. This happened to 14 questions. In three of these cases there really was no verb (e.g. "How many liters in a gallon?"), so the system would have needed a special rule to insert the verb "are". In one case the main verb "was" was not analyzed as a verb chunk. In the other ten cases, the main verb is analyzed as a noun. This problem is probably due to too few questions in the training material of the tagger.

Second, two chunks are assigned the same relation and there is no coordinating conjunction between them. This happened to 22 questions. Sometimes the chunks really have the same relation (e.g. "*In Poland*, *where* do most people live?"), sometimes the analysis is due to the relation finder's failure to distinguish between different object relations ("*What* do you call *a newborn kangaroo*?"), but most of the time the analysis is just plain wrong. Again, more questions in the training data could improve performance.

Third, a chunk has a direct relation to the central verb that does not fit any of the predefined categories (28 cases). These are mostly nouns mistagged as adverbs which then give rise to adverbial chunks being neither locative nor temporal, manner or purpose/reason. In some cases they are adjectival complements of "to be". The system needs to be extended to deal with these categories.

Fourth, questions with "How many/How much" or "How X" where X is some adjective cannot be converted (12 cases). Finally, there are some rare cases, like no relation between the *wh*-phrase and the central verb or failure to recognize the question phrase as such.

Shapaqa treats "which/what X" questions like simple "who/what" questions. To see in how far this influences performance, we manually checked the top frequency answers found on the WWW for 10 of these questions. In three cases the topmost answer looks okay. In two other cases at least the second answer is correct (e.g. "What river in the US is known as the Big Muddy?" — "The river"; "The Missouri"). For the remaining five questions, the missing constraint leads to wrong results ("Which president was unmarried?" — "The mother"). So, ideally, we would want to use the extra information. However, we would then need a component that can e.g. decide whether something is a "mountain range in North America". Until that time, our solution is an approximation.

We conducted a similar survey to find out how harmful our approach of ignoring the dif-

ference between *who* and *what* is. A manual check on Shapaqa-WWW's answers to five questions of each type suggests that this simplification does not introduce many errors. This makes sense, given that questions like "Who/What discovered radium?" are very unlikely to have any "what" answers, and questions like "Who/What is Australia's national flower?" are unlikely to have any "who" answers. The only counter-example is "Who developed the Macintosh computer?", for which the assessors did not accept "Apple Computer Inc.".

Clearly, the forced mapping from preliminary WWW answers to TREC documents is far from ideal. Sometimes Shapaqa-WWW finds the correct answer on the WWW but cannot map it. One reason is that no answer exists in the collection. This happened with "What is Australia's national flower?" — "The Golden Wattle". Alternatively, the answer that came up highest from the WWW does not exist in the collection but others do. This happened with "What is a shaman?" and the answer "a healer" (other systems found answers like "tribal magician" or "a kind of priest").

A problem for both versions of Shapaqa are question-answer-pairs like "What is mold?" — "Mold is a problem". This answers the question in letter but not in spirit. There is probably a limited number of abstract nouns that can occur in this construction (another one is "solution") and explicitly excluding the most common ones might be an opportunistic solution. A more principled approach would probably need semantic knowledge.

"What is a panic disorder?" — "A panic disorder is a type of generalized anxiety disorder." Although this answer as a whole is okay, Shapaqa identifies "type" as the head of the (predicative) object, so it is this word that gets looked for in the TREC document sentences, which might or might not work. As in the previous case, there are probably only a limited number of nouns that cause this problem but a general solution needs semantic knowledge.

"What is epilepsy?" — "Epilepsy is a common neurological disorder." This answer is correct and the head is correctly identified, too. However, as the head is rather unspecific, the answer string that is finally extracted from the TREC documents ("from an inner-ear disorder that causes vertigo") makes the answer invalid. In contrast to the previous cases, this problem is entirely due to the forced mapping from WWW answers to TREC documents, so its solution is not of general interest. One might try to find a match not only for the headword but also for the other words in the chunk.

7.5 Related research

A good introduction to the history and state-of-the-art of QA and related fields is Hirschman and Gaizauskas (2001). Many recent system descriptions are contained in the TREC proceedings. Here we will focus on systems that share one of the key techniques with Shapaqa: grammatical relations, frequencies, use of the WWW, and machine learning.

Many TREC QA systems employ Named Entity (NE) tagging, which is sometimes also

referred to as shallow parsing. However, that is a different use of the term than in this thesis. NEs are mostly chunks, e.g. [$_{PERSON}$ `Pierre Vinken ]` or [$_{DATE}$ `Nov. 29 ]`, but many chunks are not NEs, e.g. [$_{NP}$ `a woman ],` [$_{VP}$ `will laugh ]`, or [$_{ADVP}$ `later ]`. To the best of our knowledge, no other QA system follows our approach of finding GRs directly after chunking. Oard et al. (2000), Harabagiu et al. (2002), Elworthy (2001), Scott and Gaizauskas (2001), Litkowski (2002), Hovy et al. (2002) and some previous submissions by the same groups employ a full parser to analyze questions and potential answer sentences. The first two then use only unlabelled dependencies derived from the parses to constrain answers whereas the others use named GRs. These include at least subject and object, but sometimes also time and location, or apposition (which we did not use). In general, if a relation in the question matches a relation in the potential answer sentence, this answer receives extra credit by the answer selection function. This is different from our approach in which we require *all* relations to match.

Not many TREC-8 systems use the frequency of an answer as a criterion for answer selection. Two exceptions are Singhal et al. (2000) and Prager et al. (2000). In the latter, frequency is only one of five features for answer selection. Besides our own, two other TREC-10 systems used the WWW for QA. Brill et al. (2002) count frequencies of n-grams in Google's text snippets, apply hand-crafted NE filters and map the best answers back to the TREC collection. The last step is similar to our mapping. Clarke et al. (2002) add documents returned by Google or AltaVista to the TREC collection. Although the system must not extract the final five answer strings from these documents, the documents influence answer selection by increasing the frequency with which certain answers occur.

There are a few online QA systems. START[26] (Katz, 1997) parses information from many domains into its knowledge bases and then answers questions from these. Its answers are often excerpts from authority sites such as online encyclopedias. Kwok, Etzioni, and Weld (2001) describe an online open-domain QA system that uses, amongst other components, Charniak's parser (Charniak, 2000) and the Link parser (Grinberg, Lafferty, and Sleator, 1995). However, the system is no longer available on the web. According to Radev et al. (2002), it worked by running on 100 workstations in parallel in order to be able to parse in reasonable time. Radev et al. (2002) describe an online QA system called NSIR.[27] It submits queries to the search engines AlltheWeb, Northern Light and Google, downloads the top x documents (as specified by the user) and chunks them. Chunks constitute potential answers and are ranked by computing the probability of a chunk's signature being of the expected answer type. The signature is the sequence of PoS tags in the chunk. This information corresponds to what we have dubbed the "rest of the chunk" plus the chunk's head's PoS. Ionaut[28] (Abney, Collins, and Singhal, 2000) works on a local cache

[26] `http://www.ai.mit.edu/projects/infolab/`

[27] There is a link to a demo on `http://perun.si.umich.edu/clair/home/nsir.htm`, but it did not work at the time of writing.

[28] `http://www.ionaut.com:8400/`

of web files, so it avoids the slow document downloading phase. It finds and ranks NEs. LCC has an online demo[29] of the system described by Harabagiu et al. (2002).

Machine learning does not play a crucial role in most QA systems. Only sometimes are off-the-shelf statistical PoS taggers, parsers or NE finders used. Exceptions are Hovy et al. (2002) who use a decision tree parser as one important component and Ittycheriah, Franz, and Roukos (2002) who train a Maximum Entropy model on answer scoring.

7.6 Summary and future research

This chapter demonstrated how MBSP can be used to find short answers to fact-based questions in large document collections. The local, independent way of classification by the memory-based modules allows parsing of partial sentences and efficient parsing on demand. We showed how components with varying levels of NLP can easily be integrated by using the lower level modules as back-off. Results are promising but leave much room for improvement. Part of the improvement can be achieved by using better classifiers. This thesis showed how performance of the relation finder can be increased by changing parameter settings and feature representations. However it still needs to be investigated what the best choice is given the tight speed and memory constraints for online applications.

Other possible improvements of Shapaqa fall outside the scope of this thesis. For example some question types are still missing. "How many/how much" questions should be relatively easy to incorporate as the answer is usually a number within a chunk. "Which/What X" questions call for using a resource such as WordNet to check whether a potential answer is of semantic category X. "How far/long/hot/..." questions also need semantic knowledge to relate adjectives and measure phrases (3 km, 5 days, 35 degrees). Questions with the verb "to be" might profit from a special "is-relation" recognizer, as these relations are often expressed by other means than a verb (e.g. "Pierre Vinken, director of Elsevier, ..."). Morphological and diathesis knowledge is needed for pairs such as "invented the telephone/invention of the telephone". The latter implies finding grammatical relations of nouns, for which we lack adequate training material at the moment.

Our vision for the future is a system that is completely trainable. It would have to learn the mapping from a natural language question to a search engine query, the linguistic annotation for questions and potential answer sentences, and the decision about whether and where an annotated sentence contains an answer to an annotated question.

[29] `http://www2.languagecomputer.com/demos/question_answering/index.html`

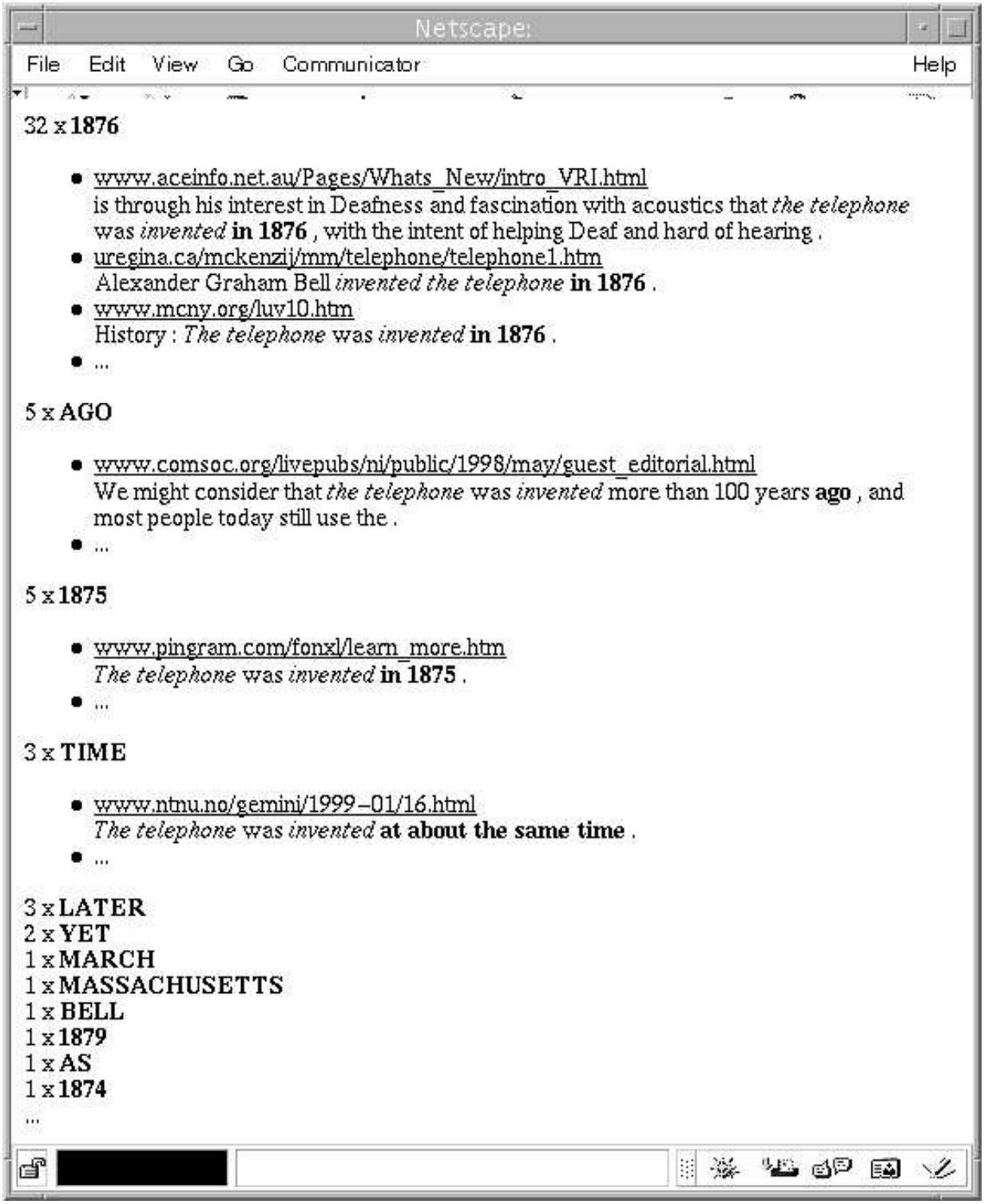

Figure 7.5: Parts of the output summary of the online version of Shapaqa (using Google) for the question "When was the telephone invented?". In this figure, colors have been replaced by italic font (verb, given phrases) and bold face font (key chunk).

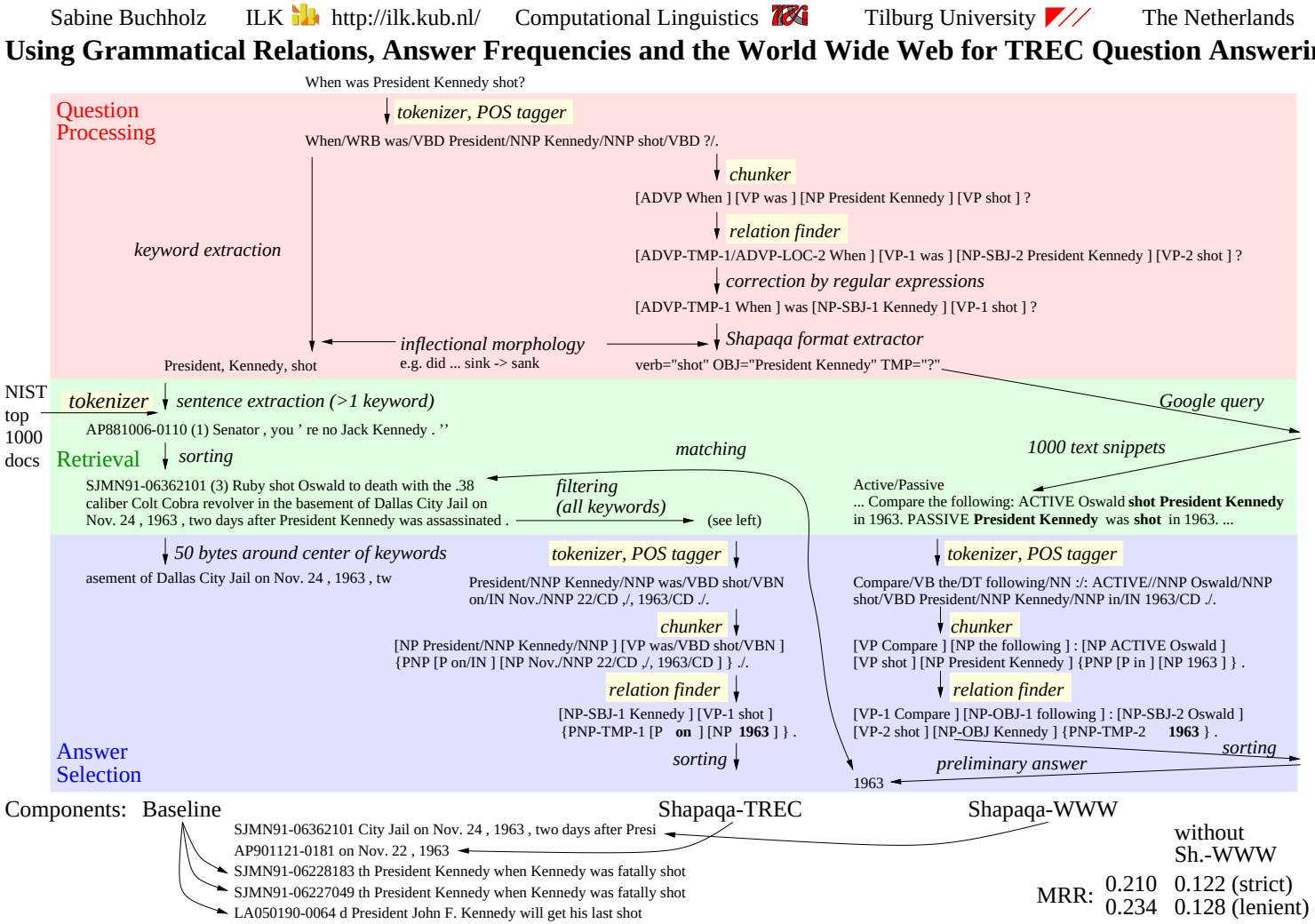

Figure 7.6: The poster for TREC-10 shows how our system generates answers for one example question. The question is at the top, the five ranked answers at the bottom. The three middle parts are question processing, information retrieval and answer selection. Left is the baseline component, in the middle Shapaqa-TREC, right Shapaqa-WWW. Strings like `SJMN91-06362101` are document IDs.

Chapter 8

Conclusions

In this thesis we used machine learning as a tool to gain insight into the linguistic phenomenon of grammatical relations. In an extensive literature study we first identified diverse sources of information that are relevant for the recognition of GRs. In systematic learning experiments we then assessed the importance of these sources quantitatively. Through detailed error analyses we investigated their function qualitatively. In the following two sections we discuss the answers to the two research questions formulated in Section 1.3:

- **What information is useful for performing the task?**
- **How can this information best be used by a Memory-Based Learner?**

8.1 Information for the task

In Chapter 2 we reviewed work that relates to GRs and especially focussed on what information it uses. The following list is based on that overview and summarizes for each type of information whether we also used it in the experiments reported in Chapters 4 to 6 and if so, whether it was useful.

Semantic heads Most current parsers use lexical heads. However most do not distinguish between syntactic and semantic heads so that the lexical head of some constituents is its semantic head (e.g. NPs, ADJPs) and the head of others is the syntactic head (e.g. PPs). Only Black et al. (1992) always use two heads. Both heads are known to be useful for at least PP attachment. Charniak (1995) and Blaheta and Charniak (2000) therefore make special provisions to include both in this case. HPSG uses only one head but it also has a mechanism (viz. unification) to ensure that relevant information from the alternative head is passed on to the parent. In our instances the word features contain mostly semantic heads (exceptions are punctuation, SBAR chunks, and PP chunks that have not been

joined into PNPs). These features proved to be very useful, e.g. ignoring the head of the focus chunk causes the second largest drop in performance (Figure 5.1, p. 104) among the 15 initial features. The verb chunk head and several left context words also contain useful information (Sections 5.9.1.2, 5.9.1.1, and 5.9.3.1).

Syntactic heads We use a separate feature for the syntactic head of PNPs (i.e. the preposition). The prepositional feature of the focus has the second highest Gain Ratio weight (Figure 4.5, p. 94) of the 15 initial features and performance decreases when it is ignored. The prepositional features of the context, however, seem to be irrelevant. In later experiments we added the VP type to the prepositional feature. This information corresponds to the syntactic head of a verbal chunk (*to* or inflection) and also proved very useful (Section 5.9.4).

Non-heads Obviously the notion of non-head depends on the definition of head. If e.g. the semantic head is taken as *the* lexical head then the syntactic head would also count as a non-head. In the experiments in which we added the "rest of the verb/focus chunk" we showed that most of the improvement was caused by information about the syntactic head of verb chunks. In the DP analysis, the determiner is the syntactic head of an NP. Black et al. (1992) also use the determiner as the secondary head of an NP. Ferro, Vilain, and Yeh (1999) allow conditions on non-heads for only a limited number of words, including some determiners. In our experiments we noted a minor improvement for fixed expressions (`NP-CLR`) through information about the *absence* of a determiner (and other non-heads), see Section 5.9.3.3. In summary we note that there is information in non-(semantic)-heads. However it mostly stems from words that are syntactic heads. As the syntactically relevant information about a word should be encoded in its PoS tag, it is sufficient to represent non-heads by their PoS.[1] Using words instead decreased performance in our experiments (Table 5.12). The exception to this rule are prepositions, which also carry semantic information or for which a specific word form can be required by subcategorization. A last point on non-heads concerns constructions for which it is unclear what the head is. A well-known example are multi-word prepositions such as "such as". In these cases it seems more appropriate to represent all participating words than to arbitrarily choose one as head. This is why some parsers include a preprocessing/tokenization step that joins these words into one token (e.g. Briscoe and Carroll (1997)). In our experiments the effect of joining multi-word prepositions was not significant.

The above analyses have interesting implications for other parsers. Lexicalized probabilistic parsers along the lines of Charniak (2000), Collins (1997) and others might consider using two heads instead of one (similar to Black et al. (1992)). To avoid too sparse data the syntactic head can often be replaced by its PoS. It remains to be explored what the overall effect on performance is in the face of the trade-off between more useful information and

[1] However this presupposes that *all* syntactically relevant information is encoded in the tag. As we saw in the discussion of the Penn Treebank tag set (Section 2.3.1) this is not always the case.

sparser data. It should be noted that the approach will still fail to account for multi-word prepositions or determiners in PPs.

DOP parsers might consider restricting the lexicalization of larger subtrees to those words that can be the syntactic or semantic head of higher constituents. The precise conditions still need to be determined. They can probably be derived from linguistic work on feature propagation but they could also be found empirically by deleting all words in certain conditions (e.g. adjectives in NP in XP) and testing whether performance changes. Again this general approach will fail for multi-word prepositions and other idioms.

Context All other parsers discussed in Chapter 2 use information from the context in a symmetrical way. The chunkers and bottom-up parsers typically use two or three left and right context elements. The generative parsers either generate all children in one step or condition on a fixed number of previously generated children, regardless of the direction. To the best of our knowledge none of the authors gives any motivation for this symmetry. In our experiments it proved best to use more information from the left context than from the right (Tables 5.1, 5.6 and 5.7).

PoS and PoS hierarchies Practically all parsers use PoS at some or all stages of the parsing process. In the previous sections we already discussed the use of PoS of non-heads. The parsers of Charniak (2000) and Collins (1997) use only the PoS of heads at higher levels. Our own results (Table 5.1) and those of Van den Bosch and Buchholz (2002) suggest that PoS information is not that important in itself. Charniak (2000) also notes that half of its function is as a back-off for unknown words. However, attenuation provides an alternative for this (Eisner, 1996a; Van den Bosch and Buchholz, 2002). Eisner (1996a) uses manually reduced versions of the PoS for back-off. Jackendoff (1977) describes how syntactic categories can be decomposed into syntactic features. Some rules can then apply to all categories with a certain feature value. We reach similar effects automatically through MVDM (see Section 4.3.7.2).

Chunks and chunk types Many parsers use chunking as an intermediate step to parsing. Chunks serve three purposes for our relation finder. Firstly, they reduce the workload: there is one instance per pair of chunks instead of one per pair of words. Secondly, they provide a level of abstraction: for example, we count distance in chunks instead of in words. Buchholz, Veenstra, and Daelemans (1999) show how much various types of chunks decrease the number of instances and the average distance between the verb and the focus and thereby increase performance. Thirdly, chunk types are used as additional information. The chunk type of the focus has the highest Gain Ratio weight of the 15 initial features and its omission decreases performance (not dramatically though).

Higher level constituents Most of the full parsers decribed in Section 2.4.2 use previously generated/built higher level constituents to guide subsequent parse decisions. As we do not build these structures, we cannot use this information source. It might be approximated by the sequence of intervening chunks: for example a sequence of "NP, NP ADJP," represents, with a certain probability, an NP or, with some other probability, some other constituent.

Linear order (direction, distance, adjacency) Linear order is such a basic concept that one might overlook its importance as an information source. Many parsers encode it directly in the rules or (sub)trees that form their basic building blocks. In parsers based on Markov grammars the direction is encoded as an extra conditional feature (Collins, 1996) or two separate left and right probabilities are used (Collins, 1997; Charniak, 2000). In our experiments, the combination of direction and distance was the only feature that achieved any serious performance when used in isolation (see Table 3.2). It also had the largest negative influence when ignored. The linear order of the intervening chunks is captured only in their symbolic representation, not in the numeric one. An approximation of distance is encoded in both representations.

Semantic types Ratnaparkhi, Reynar, and Roukos (1994) use the bit string encoding of the clustering of words as additional features for PP attachment. Black et al. (1992) manually assign semantic categories to words and use their bit strings as features in their parser. Charniak (1997) uses word classes that are acquired by clustering as back-off for the word itself. Buchholz, Veenstra, and Daelemans (1999) use separately assigned adverbial functions, and Ferro, Vilain, and Yeh (1999) use Named Entity types and WordNet classes as additional information sources. Also many QA systems use Named Entity types and WordNet classes. Ushioda et al. (1993) use lists of temporal nouns. All these approaches are ways to encode semantic type information. In our experiments, semantic types are implicitly encoded through the Modified Value Difference Metric (see Section 4.3.7.2). Its use for the focus and context words proved very useful (Table 4.6).

Subcategorization Abney (1991), Model 2 and 3 of Collins (1997), Carroll, Minnen, and Briscoe (1998), and Ferro, Vilain, and Yeh (1999) use explicit subcategorization information for parsing. The lexicalized, probabilistic parsers of Section 2.4.2 mostly use implicit subcategorization, as does our relation finder (see Section 5.9.1.2). As we represent the verb in a separate feature we are able to measure the effect of ignoring this information source. We showed what influence the verb has on the performance of the most frequent relations. However, in order to exclude that other lexical properties of the verb such as tense influence the outcome, we would need lemmatization.

Verb chunk properties Grefenstette (1996) first finds verb chunks and then determines their finiteness and voice. Ferro, Vilain, and Yeh (1999) also use this additional informa-

tion. In Kübler and Hinrichs (2001a) the chunker finds generic VP chunks for English but distinguishes between finite and non-finite verbal chunks for German. The SPARKLE project (Carroll et al., 1997) also uses four types of verbal chunks. Our VP types in addition encode agreement and distinguish not only infinitival from participial chunks but also base infinitives from *to*-infinitives. This information proved useful (see Section 5.9.3.3). In Section 6.1.1 we added information on passivity. However this information does not seem to be necessary for surface grammatical relations. Tense is a separate feature in grammar theories like HPSG or LFG. It is not consistently represented in our system. We saw one example for which tense was probably useful (Section 5.9.1.2).

Lemmas Briscoe and Carroll (1997) lemmatize text prior to parsing. Ferro, Vilain, and Yeh (1999) use word stems as additional features. In our system, a task-specific lemmatization is implicitly achieved through the Modified Value Difference Metric for words (see Section 4.3.7.2) but we did not quantify its effect.

Attenuation Eisner (1996a) replaces low frequency words by special symbols encoding capitalization, hyphenation, suffixes, etc. Brent (1991a) also uses capitalization and typical suffixes for subcategorization acquisition from raw text. Collins (1999) replaces low frequency words with one special symbol. The latter approach is the one we followed too and it increased performance slightly.

Punctuation In the Penn Treebank, punctuation is treated like words: it is assigned a PoS and is attached to the tree. Therefore most parsers that are trained on the Penn Treebank do not need to make special provisions for punctuation. However, some use certain punctuation signs for additional features, e.g. Collins (1996) uses commas in the "distance" and Ratnaparkhi (1997) has features for parentheses, commas, and periods. In our experiments, some numeric features for the intervening material explicitly count commas, quotation marks and colons. Of these, commas are most important. Information on parentheses did not prove useful. Punctuation also shows up in the context features but we did not quantify its importance there.

Information on typical tagger and chunker errors Our experiments showed that performance on automatically tagged and chunked text improves if the relation finder is also trained on this kind of text (Section 6.1.4). In contrast to treebank material this text implicitly contains information about typical errors of the tagger and chunker. The Ramshaw and Marcus (1995) data set for NP chunking is tagged with the Brill tagger, so chunkers trained on this set automatically use the "typical error" information source. Probabilistic parsers often integrate the tagger into the system by multiplying the probability of the tree with the probability of the tag sequence. This is another way to use this information source.

8.2 The information and Memory-Based Learning

The previous section showed what information sources we used in our experiments and which of them proved useful. The current section is based on the same experiments but summarizes them under a different aspect: how can this information best be represented for MBL?

MVDM We saw the effect of MVDM for many of the information sources above, not only for words and PoS but also for the distance (see Section 4.3.7.2). Conceptually it enables the algorithm to treat feature values as collections of properties instead of as atomic unrelated symbols. Sections 4.3.7.3, 4.3.7.4, and 5.9.5 suggested that some of the potential disadvantages of the MVDM metric can be compensated for by using a larger value of k and Inverse Distance or Exponential Decay distance weighted class voting and by replacing all low frequency values by one unique value. The latter technique especially increased speed which can be a problem when using MVDM.

Numeric features for sequences Sequence-valued features constitute a particularly interesting application of MVDM. We will discuss sequences in more detail in Section 8.4.1. Here we only note that we presented two alternatives for the sequence-as-symbolic-value representation: a set of numeric features and a set of symbolic feature values. These representations are much faster than the sequence-valued one. The numeric representation even illustrates the surprising fact that in this implementation of MBL the addition of features may speed up classification. Other examples for this effect are the PoS and chunk type features (see Table 5.1).

Redundancy PoS and chunk type features mainly contain redundant information in the sense that their value can often be inferred from the value of the corresponding word feature. This is confirmed by the fact that their omission does not decrease performance. On the other hand their omission does not increase performance either. This shows that IB1-IG is robust against this kind of redundancy. Other cases of redundancy come from the representation of the verb chunk and its context in addition to the focus context, as these might overlap, and from the representation of the preposition of a PNP chunk by a separate feature (Sections 5.9.3.1 and 5.2, respectively). A related phenomenon concerns non-target relation classes. Section 6.1.2 showed that performance on a subset of relations can drop if relations not belonging to this set are not represented in the data.

Distributed information TiMBL's current feature weighting schemes assume feature independence. Redundant and distributed information violate this assumption. In this light, it is interesting to see that the algorithm can make use of distributed information. Examples are the verb/focus PoS and the rest of the verb/focus chunk, which together

encode the VP type (Section 5.9.3.3), or the numeric comma, CC and NP features, which together encode cases of coordinated NPs (Section 5.9.3.2).

8.3 Practical results

Based on the above insights we developed a memory-based grammatical relation finder that operates in the context of the Memory-Based Shallow Parser. It finds grammatical relations to verbs, including

- different kinds of complements as well as the most prominent semantic types of adjuncts,
- local and non-local relations, including control,
- dependents of all syntactic categories,
- dependents at an unrestricted distance,
- relations to multiple heads in cases of coordination.

On this set the relation finder achieves a performance of $F_\beta = 81.36$ when using treebank tags and chunks, and $F_\beta = 72.59$ with automatically tagged and chunked material (tenfold cross validation on 500,000 tokens of WSJ). With 1,600,000 tokens of WSJ and Brown training material it achieves $F_\beta = 73.47$ on Brown test data.[2]

This result is comparable to that of GR extraction methods that are based on full parsers. It should be noted that the task is a difficult one. Trained lexicographers are known to achieve about 91% pairwise agreement on the simpler task of identifying just the complements in a sentence, without determining their type (see Section 2.5.1 and Meyers, Macleod, and Grishman (1996)). We showed that even a lower performing but faster version of this relation finder (using IGTree and fewer features) can be used to enhance the precision of web-based question answering. MBSP's ability to perform local, independent classification made it possible to parse text snippets instead of complete documents and to perform parsing on demand.

8.4 Future Research

There are many directions for future research. An obvious one is to extend the current method to include grammatical relations to non-verbs. However, as discussed in Section 1.1.1, this would require the availability of appropriate training material. Another research direction concerns the best treatment of unknown words through the promising

[2] without the `-CLR` tag

technique of attenuation. Potentially attenuation can not only simulate PoS tags but also Named Entity-like semantic information; e.g. *-man, -son* and *-er* are typical suffixes for persons whereas words ending in *-shire* or *-burg* often describe locations. Two further research directions are discussed below.

8.4.1 Automatic feature construction for the representation of sequences and trees

Our discussion on the best representation for the sequence of intervening chunks and of non-heads showed that although using the whole sequence as one symbolic value together with MVDM gives good results, other representations yield equally good results and are more efficient (faster, less memory). In our case we found the better representations manually. In general one would want to be able to also find them automatically. This is the topic of automatic feature construction. However the search space is huge and it looks quite challenging to automatically construct complex features like:

- the first PoS from the left that is TO or MD or starts with VB (the rule for our VP type feature)
- a form of *be, get* or *become*, possibly followed by words that are not verbs (e.g. adverbs), followed by the head of the chunk which is tagged VBN or VBD (our rule for passive; VBD is for mistagged participles)
- a constituent with two or more VP children, one or more of the constituents comma, CC, CONJP,[3] and nothing else (the rule for coordinated VPs by Charniak (1999))

An alternative to using MVDM for sequences would be to implement a specific metric for sequence-valued features, just as we have a specific metric for numeric features. An obvious candidate is the *edit distance*, also called Levenshtein distance (Levenshtein, 1966), which defines similarity between two sequences as the minimum number of insertions, deletions, substitutions and possibly swaps that are necessary to convert one sequence into the other. There are efficient ways to compute the similarity of a test sequence to a whole collection of training sequences (Sankoff and Kruskal, 1983). It is even possible to associate weights with specific operations, e.g. we could give a low weight to the insertion/deletion of an adverb. Clark (2002) describes an approach to supervised learning of morphology using the Expectation Maximization algorithm (Dempster, Laird, and Rubin, 1977) to train stochastic transducers. He notes that this method is a generalization of edit distance and shows how it can be used as a kernel function for a Memory-Based Learning algorithm. Other kernel functions can compute the similarity between two sequences or trees efficiently, while still implicitly taking into account all subsequences or

[3] multi-word conjunctions like "as well as"

subtrees. They have been used in machine learning algorithms such as the Voted Perceptron (Freund and Schapire, 1999) or Support Vector Machines (SVM), see Cortes and Vapnik (1995), to (re)rank full parses and tag NEs (Collins and Duffy, 2002) and to extract `person-affiliation` and `organization-location` relations from texts annotated with NEs and NPs (Zelenko, Aone, and Richardella, 2002). The latter application suggests that extraction of grammatical relations should be possible, too. Zelenko, Aone, and Richardella (2002) claim that, in contrast to kernel methods, "feature-based representations produce inherently local representations of objects, for it is computationally infeasible to generate features involving long-range dependencies". Our experiments on the representation of the intervening material show that this is not true in general.

8.4.2 Separation of work between the modules

We mentioned in Section 2.3.1 that the tag set of the Penn Treebank, which we have used in MBSP, conflated several tags of other sets. We also saw that some researchers have (re)introduced some lost or new distinctions, especially for auxiliaries (Eisner, 1996a; Charniak, 1999). Similarly the syntactic categories (on which our chunk types are based) conflate some distinctions that are known to be important, and again Collins (1999) for example found it useful to split the S category. Kübler and Hinrichs (2001b) and Carroll and Rooth (1998a) let the chunker already distinguish several types of VPs whereas we encode this information in a separate feature later. When converting the treebank trees to chunks we made certain decisions that followed previous work on chunking, e.g. with respect to possessives, WHXPs and intra-chunk coordination (Section 3.1.2.4). In summary it remains to be seen whether the current set of PoS tags and chunk types used in MBSP is the optimal one. We saw in Section 6.1.4 that the PNP finder can be replaced by a simple regular expression. It might also be possible to integrate PNP chunking into the normal chunker (Grefenstette, 1996; Aït-Mokhtar and Chanod, 1997a; Abney, 1991; Carroll et al., 1997). All these considerations pertain to the question of the best separation of work between the modules. In principle, the more information the lower modules determine and the more they reduce the search space the better for the relation finder. In practice, however, some decisions might be too difficult for the lower modules. It might then be better to postpone these decisions than to introduce too many errors.

8.5 Summary

We use machine learning to investigate what information sources can be used to find grammatical relations to verb chunks in English sentences, and how this information can best be exploited by a Memory-Based Learner. We showed that the distinction into heads and non-heads that is often used in the probabilistic parsing literature obscures the fact that there are syntactic and semantic heads and that both carry valuable information. We also explored information contained in sequences of PoS or chunks. This information

can be used implicitly, by representing the whole sequence, or explicitly, by extracting the most important pieces of information into separate features. With respect to MBL, we showed how the Modified Value Difference Metric implicitly encodes syntactic and semantic knowledge. We also demonstrated how the relation finder enhances precision of an online QA system.

Index

References

Abney, S. 1991. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht, pages 257–278.

Abney, Steven. 1996. Partial parsing via finite-state cascades. In John Carroll, editor, *Workshop on Robust Parsing (ESSLLI'96)*, pages 8–15.

Abney, Steven, Michael Collins, and Amit Singhal. 2000. Answer extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP)*. ACL.

Abney, Steven P. 1987. *The English Noun Phrase in its Sentential Aspect*. Ph.D. thesis, MIT.

Aha, D. W. 1997. Lazy learning: Special issue editorial. *Artificial Intelligence Review*, 11:7–10.

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

Aït-Mokhtar, S. and J.-P. Chanod. 1997a. Incremental finite-state parsing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, pages 72–79.

Aït-Mokhtar, Salah and Jean-Pierre Chanod. 1997b. Subject and object dependency extraction using finite-state transducers. In *Proceedings of ACL'97 Workshop on Information Extraction and the Building of Lexical Semantic Resources for NLP Applications*.

Argamon, S., I. Dagan, and Y. Krymolowski. 1998. A memory-based approach to learning shallow natural language patterns. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 67–73.

Argamon, Shlomo, Ido Dagan, and Yuval Krymolowski. 1999. A memory-based approach to learning shallow natural language patterns. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(3):369–390. Special Issue on Memory-Based Language Processing.

Baayen, R. H., R. Piepenbrock, and H. van Rijn. 1993. *The CELEX lexical data base on CD-ROM.* Linguistic Data Consortium, Philadelphia, PA.

Bies, A., M. Ferguson, K. Katz, and R. MacIntyre, 1995. *Bracketing Guidelines for Treebank II Style, Penn Treebank Project.* University of Pennsylvania, Philadelphia.

Black, E., F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, and S. Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the DARPA Speech and Natural Language Workshop.*

Blaheta, D. and E. Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 234–240.

Bod, R. 1992. A computational model of language performance: Data oriented parsing. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 855–859.

Bod, R. 2001. What is the minimal set of fragments that achieves maximal parse accurary? In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 66–73.

Boguraev, B., E. Briscoe, J. Carroll, D. Carter, and C. Grover. 1987. The derivation of a grammatically-indexed lexicon from the Longman Dictionary of Contemporary English. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 193–200.

Brants, Thorsten and Wojciech Skut. 1998. Automation of treebank annotation. In David M. W. Powers, editor, *Proceedings of the Joint Conference on New Methods in Language Processing and Computational Natural Language Learning workshop (NeMLaP-3/CoNLL98)*, pages 49–57.

Brent, Michael R. 1991a. *Automatic acquisition of subcategorization frames from unrestricted English.* Ph.D. thesis, MIT, Cambridge, MA.

Brent, Michael R. 1991b. Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 209–214.

Brent, Michael R. 1993. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19(2):243–262.

Brent, Michael R. 1994. Surface cues and robust inference as a basis for the early acquisition of subcategorization frames. In L. Gleitman and B. Landau, editors, *The Acquisition of the Lexicon.* MIT Press, Cambridge, MA, pages 433–470.

Brill, E. and P. Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*.

Brill, Eric. 1993. *A Corpus-Based Approach to Language Learning*. Ph.D. thesis, University of Pennsylvania, Department of Computer and Information Science.

Brill, Eric, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2002. Data-intensive question answering. In Voorhees and Harman (Voorhees and Harman, 2002), pages 393–400. Available at http://trec.nist.gov/pubs.html.

Briscoe, Ted and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, pages 356–363.

Briscoe, Ted, John Carroll, Jonny Graham, and Ann Copestake. 2002. Relational evaluation schemes. In *Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems (LREC 2002 Workshop)*.

Buchholz, Sabine. 2002. Using grammatical relations, answer frequencies and the world wide web for TREC question answering. In Voorhees and Harman (Voorhees and Harman, 2002), pages 502–509. Available at http://trec.nist.gov/pubs.html.

Buchholz, Sabine and Walter Daelemans. 2001a. Complex answers: A case study using a WWW question answering system. *Natural Language Engineering*, 7(4):301–323. Special issue on question answering, Guest editors: Lynette Hirschman and Robert Gaizauskas.

Buchholz, Sabine and Walter Daelemans. 2001b. SHAPAQA: Shallow parsing for question answering on the world wide web. In *Proceedings of the EuroConference on Recent Advances in NLP*.

Buchholz, Sabine, Jorn Veenstra, and Walter Daelemans. 1999. Cascaded grammatical relation assignment. In Pascale Fung and Joe Zhou, editors, *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, pages 239–246.

Bunt, Harry and Arthur Horck, editors. 1996. *Discontinuous Constituency*, volume 6 of *Natural Language Processing*. Mouton de Gruyter, Berlin; New York.

Burke, Robin D., Kristian J. Hammond, Vladimir A. Kulyukin, Steven L. Lytinen, Noriko Tomuro, and Scott Schoenberg. 1997. Question answering from frequently-asked question files: Experiences with the faq finder system. Technical Report TR-97-05, The University of Chicago.

Carroll, Glenn and Mats Rooth. 1998a. Valence induction with a head-lexicalized PCFG. In *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing (EMNLP-3)*.

Carroll, Glenn and Mats Rooth. 1998b. Valence induction with a head-lexicalized PCFG - gold. In *Inducing Lexicons with the EM Algorithm*, volume 4 (3) of *AIMS Report*. IMS, Universität Stuttgart.

Carroll, John and Ted Briscoe. 2001. High precision extraction of grammatical relations. In *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT 2001)*.

Carroll, John, Ted Briscoe, Nicoletta Calzolari, Stefano Federici, Simonetta Montemagni, Vito Pirelli, Greg Grefenstette, Antonio Sanfilippo, Glenn Carroll, and Mats Rooth. 1997. SPARKLE work package 1 – specification of phrasal parsing. Technical report, SPARKLE project. Available at http://www.ilc.pi.cnr.it/sparkle/sparkle.html.

Carroll, John, Guido Minnen, and Ted Briscoe. 1998. Can subcategorisation probabilities help a statistical parser? In *Proceedings of the 6th ACL/SIGDAT Workshop on Very Large Corpora*.

Carroll, John, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC)*.

Charniak, Eugene. 1995. Parsing with context-free grammars and word statistics. Technical Report CS-95-28, Brown University.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.

Charniak, Eugene. 1999. A maximum-entropy-inspired parser. Technical Report CS-99-12, Brown University.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139.

Chomsky, Noam. 1970. Remarks on nominalization. In Roderick A. Jacobs and Peter S. Rosenbaum, editors, *Readings in English Transformational Grammar*. Ginn, Waltham, Mass., pages 184–221. Reprinted in 1980 by Georgetown University Press, Washington, D.C.

Chomsky, Noam. 1986. *Barriers*, volume 13 of *Linguistic Inquiry Monographs*. MIT Press, Cambridge, MA.

Clark, Alexander. 2002. Memory-based learning of morphology with stochastic transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 513–520.

Clark, Stephen and Julia Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems (LREC 2002 Workshop)*.

Clarke, C.L.A., G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. 2002. Web reinforced question answering (MultiText experiments for TREC 2001). In Voorhees and Harman (Voorhees and Harman, 2002), pages 673–679. Available at http://trec.nist.gov/pubs.html.

Cohen, P. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press, MA, USA.

Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Collins, Michael and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270.

Collins, M.J and J. Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora*.

Cortes, C. and V. Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

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

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

Crouch, Richard, Ronald M. Kaplan, Tracy H. King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad-coverage stochastic parser. In *Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems (LREC 2002 Workshop)*.

Daelemans, W. 1996. Abstraction considered harmful: lazy learning of language processing. In H. J. van den Herik and A. Weijters, editors, *Proceedings of the Sixth Belgian–Dutch Conference on Machine Learning*, pages 3–12, Maastricht, The Netherlands. MATRIKS.

Daelemans, W., S. Buchholz, and J. Veenstra. 1999. Memory-based shallow parsing. In *Proceedings of the Third Computational Natural Language Learning workshop (CoNLL).*

Daelemans, W. and A. van den Bosch. 1992. Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers and A. Nijholt, editors, *Proceedings of TWLT3: Connectionism and Natural Language Processing*, pages 27–37. Twente University.

Daelemans, W., A. van den Bosch, and A. Weijters. 1997. IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Daelemans, W., A. van den Bosch, and J. Zavrel. 1997. A feature-relevance heuristic for indexing and compressing large case bases. In M. van Someren and G. Widmer, editors, *Poster Papers of the Ninth European Conference on Machine Learning*, pages 29–38, Prague, Czech Republic. University of Economics.

Daelemans, W., J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proceedings of the 4th ACL/SIGDAT Workshop on Very Large Corpora*, pages 14–27.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2000. TiMBL: Tilburg memory based learner, version 3.0, reference guide. ILK Technical Report 00-01, Tilburg University. Available from http://ilk.kub.nl.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2001. TiMBL: Tilburg memory based learner, version 4.0, reference guide. ILK Technical Report 01-04, Tilburg University. Available from http://ilk.kub.nl.

Dagan, Ido and Yuval Krymolowski. 2002. Compositional partial parsing by memory-based sequence learning. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-Oriented Parsing.* Centre for the Study of Language and Information (CSLI), chapter II.6.

Dempster, A.P., N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Dietterich, Thomas G. 1998. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136.

Dudani, S.A. 1976. The distance-weighted k-nearest neighbor rule. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 6, pages 325–327.

Eisner, Jason M. 1996a. An empirical comparison of probability models for dependency grammar. Technical Report IRCS-96-11, Institute for Research in Cognitive Science, University of Pennsylvania.

Eisner, Jason M. 1996b. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

Elworthy, D. 2001. Question answering using a large NLP system. In Voorhees and Harman (Voorhees and Harman, 2001). Available at http://trec.nist.gov/pubs.html.

Ersan, Murat and Eugene Charniak. 1995. A statistical syntactic disambiguation program and what it learns. Technical Report CS-95-29, Brown University.

Ferro, Lisa, Marc Vilain, and Alexander Yeh. 1999. Learning transformation rules to find grammatical relations. In *Proceedings of the Third Computational Natural Language Learning workshop (CoNLL)*, pages 43–52.

Freund, Y. and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press.

Gildea, Daniel. 2001. Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Grefenstette, Gregory. 1996. Light parsing as finite-state filtering. In Wolfgang Wahlster, editor, *Workshop on Extended Finite State Models of Language, ECAI'96, Budapest, Hungary*. John Wiley & Sons, Ltd.

Grinberg, Dennis, John Lafferty, and Daniel Sleator. 1995. A robust parsing algorithm for link grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies (IWPT-4)*.

Grishman, Ralph, Catherine Macleod, and Adam Meyers. 1994. Comlex syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, pages 268–272. New York University.

Harabagiu, S., D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, and P. Morărescu. 2001. FALCON: Boosting knowledge for answer engines. In Voorhees and Harman (Voorhees and Harman, 2001), pages 479–488. Available at http://trec.nist.gov/pubs.html.

Harabagiu, S., D. Moldovan, M. Paşca, M. Surdeanu, R. Mihalcea, R. Gîrju, V. Rus, F. Lăcătuşu, P. Morărescu, and R. Bunescu. 2002. Answering complex, list and context questions with LCC's question-answering server. In Voorhees and Harman (Voorhees and Harman, 2002), pages 355–361. Available at http://trec.nist.gov/pubs.html.

Hemphill, C.T., J.J. Godfrey, and G.R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 96–101.

Hindle, D. and M. Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19:103–120.

Hinrichs, Erhard W., Sandra Kübler, Frank H. Müller, and Tylman Ule. 2002. A hybrid architecture for robust parsing of german. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*.

Hirschman, Lynette and Robert Gaizauskas. 2001. Natural language question answering: the view from here. *Natural Language Engineering*, 7(4):275–300. Special issue on question answering, Guest editors: Lynette Hirschman and Robert Gaizauskas.

Hovy, E., U. Hermjakob, , and C.-Y. Lin. 2002. The use of external knowledge in factoid QA. In Voorhees and Harman (Voorhees and Harman, 2002), pages 644–652. Available at http://trec.nist.gov/pubs.html.

Ittycheriah, Abraham, Martin Franz, and Salim Roukos. 2002. IBM's statistical question answering system – TREC-10. In Voorhees and Harman (Voorhees and Harman, 2002), pages 258–264. Available at http://trec.nist.gov/pubs.html.

Jackendoff, Ray. 1977. *$\bar{X}$ Syntax: A Study of Phrase Structure*. MIT Press, Cambridge, MA.

Jacobs, Joachim. 1994. *Kontra Valenz*, volume 12 of *Fokus*. WVT, Wissenschaftlicher Verlag Trier.

Joshi, Aravind K. 1987. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, pages 87–115.

Kaplan, Ronald M. and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representations. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, pages 173–281.

Katz, Boris. 1997. From sentence processing to information access on the world wide web. In *AAAI Spring Symposium on Natural Language Processing for the World Wide Web*.

Kolodner, J. 1993. *Case-based reasoning*. Morgan Kaufmann, San Mateo, CA.

Krymolowski, Yuval and Ido Dagan. 2000. Incorporating compositional evidence in memory-based partial parsing. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 45–52.

Krymolowski, Yuval and Ido Dagan. 2002. Targeted partial dependency parsing. Available at `http://www.cs.biu.ac.il/~yuvalk/papers/index.html#mbsl`.

Kübler, Sandra and Erhard W. Hinrichs. 2001a. From chunks to function-argument structure: A similarity-based approach. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 338–345. ACL.

Kübler, Sandra and Erhard W. Hinrichs. 2001b. TüSBL: A similarity-based chunk parser for robust syntactic processing. In *Proceedings of the First International Human Language Technology Conference (HLT-2001)*.

Kübler, Sandra and Heike Telljohann. 2002. Towards a dependency-oriented evaluation for partial parsing. In *Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems (LREC 2002 Workshop)*.

Kucera, H. and W.N. Francis. 1967. *Computational analysis of present-day American English*. Brown University Press, RI.

Kupiec, Julian M. 1992. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*, 6:225–242.

Kwok, Cody C.T., Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the web. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*.

Lau, Raymond, Ronald Rosenfeld, and Salim Roukos. 1993. Adaptive language modeling using the maximum entropy principle. In *Proceedings of the ARPA Human Language Technology Workshop*.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics – Doklady*, 10(8):707–710.

Levin, B. 1993. *English Verb Classe and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago IL.

Li, Xin and Dan Roth. 2001. Exploring evidence for shallow parsing. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL)*.

Lin, Dekang. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*.

Litkowski, K.C. 2002. CL Research experiments in TREC-10 question answering. In Voorhees and Harman (Voorhees and Harman, 2002), pages 122–131. Available at http://trec.nist.gov/pubs.html.

Magerman, David M. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283.

Manning, Christopher D. 1993. Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 235–242.

Marcus, M., B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 110–115.

Mel'čuk, Igor A. 1988. *Dependency syntax : theory and practice.* SUNY Series in Linguistics. State University of New York Press, Albany.

Merlo, P., M. Crocker, and C. Berthouzoz. 1997. Attaching multiple prepositional phrases: Generalized backed-off estimation. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-2).*

Meyers, Adam, Catherine Macleod, and Ralph Grishman. 1994. Standardization of the complement adjunct distinction. Proteus Project Memorandum 64, Computer Science Department, New York University.

Meyers, Adam, Catherine Macleod, and Ralph Grishman. 1996. Stardardization of the complement/adjunct distinction. In *Proceedings of EURALEX'96, Göteborg University, Sweden.*

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244. Special issue.

Moldovan, Dan, Sanda Harabagiu, Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju, and Vasile Rus. 2000. Lasso: A tool for surfing the answer net. In Voorhees and Harman (Voorhees and Harman, 2000), pages 175–183. Available at http://trec.nist.gov/pubs.html.

Moortgat, Michael, Ineke Schuurman, and Ton van der Wouden. 2001. CGN syntactische annotatie. Available from: http://lands.let.kun.nl/cgn/home.htm.

Muñoz, Marcia, Vasin Punyakanok, Dan Roth, and Dav Zimak. 1999. A learning approach to shallow parsing. In Pascale Fung and Joe Zhou, editors, *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, pages 168–178.

Noreen, Eric W. 1989. *Computer-intensive methods for testing hypotheses: an introduction.* John Wiley and Sons, Inc.

Oard, Douglas W., Jianqiang Wang, Dekang Lin, and Ian Soboroff. 2000. TREC-8 experiments at Maryland: CLIR, QA and routing. In Voorhees and Harman (Voorhees and Harman, 2000), pages 623–636. Available at http://trec.nist.gov/pubs.html.

Pollard, Carl and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics, Volume 1: Fundamentals*, volume 13 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford.

Pollard, Carl and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. Studies in contemporary linguistics. University of Chicago Press.

Prager, John, Dragomir Radev, Eric Brown, Anni Cohen, and Valerie Samn. 2000. The use of predictive annotation for question answering in TREC8. In Voorhees and Harman (Voorhees and Harman, 2000), pages 399–410. Available at http://trec.nist.gov/pubs.html.

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

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

Quirk, R., S. Greenbaum, G. Leech, and J. Svartvik. 1985. *A comprehensive grammar of the English language*. Longman, London.

Radev, Dragomir, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. 2002. Probabilistic question answering on the web. In *Proceedings of the Eleventh International World Wide Web Conference (WWW2002)*.

Ramshaw, L.A. and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora*, pages 82–94.

Ratnaparkhi, A. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-2)*, pages 1–10.

Ratnaparkhi, A., J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the ARPA Human Language Technology Workshop*, March.

Ratnaparkhi, Adwait. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278.

Sampson, G. 1995. *English for the computer.* Oxford University Press, Oxford, UK.

Sankoff, David and Joseph B. Kruskal. 1983. *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison.* Addison-Wesley.

Scha, R. 1992. Virtual Grammars and Creative Algorithms. *Gramma/TTT Tijdschrift voor Taalkunde*, 1:57–77.

Scott, S. and R. Gaizauskas. 2001. University of Sheffield TREC-9 Q&A system. In Voorhees and Harman (Voorhees and Harman, 2001). Available at http://trec.nist.gov/pubs.html.

Sejnowski, T. J. and C. S. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

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

Singhal, Amit, Steve Abney, Michiel Bacchiani, Michael Collins, Donald Hindle, and Fernando Pereira. 2000. AT&T at TREC-8. In Voorhees and Harman (Voorhees and Harman, 2000), pages 317–330. Available at http://trec.nist.gov/pubs.html.

Skut, Wojciech, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1997a. Annotating unrestricted German text. In *Proceedings der 6. Fachtagung der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft.*

Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997b. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, pages 88–95.

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

Tesnière, Lucien. 1959. *Eléments de syntaxe structurale.* Klincksieck, Paris.

Tjong Kim Sang, E. and J.B. Veenstra. 1999. Representing text chunks. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pages 173–179.

Tjong Kim Sang, Erik. 2002. Memory-based shallow parsing. *Journal of Machine Learning Research*, 2:559–594.

Tjong Kim Sang, Erik and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the Fourth Conference on Computational Natural Language Learning (CoNLL) and the Second Learning Language in Logic Workshop (LLL)*, pages 127–132.

Ushioda, Akira, David Evans, Ted Gibson, and Alex Waibel. 1993. Frequency estimation of verb subcategorization frames based on syntactic and multidimensional statistical analysis. In *Proceedings of the Third International Workshop on Parsing Technologies (IWPT-3)*.

van den Bosch, A. 1999a. Instance-family abstraction in memory-based language learning. In I. Bratko and S. Dzeroski, editors, *Machine Learning: Proceedings of the Sixteenth International Conference, (ICML)*, pages 39–48, Bled, Slovenia.

van den Bosch, Antal. 1999b. Careful abstraction from instance families in memory-based language learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(3):339–368.

van den Bosch, Antal and Sabine Buchholz. 2002. Shallow parsing on the basis of words only: A case study. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 433–440.

van Halteren, Hans, Jakub Zavrel, and Walter Daelemans. 2001. Improving accuracy in word class tagging through combination of machine learning systems. *Computational Linguistics*, 27(2):199–230.

van Rijsbergen, C.J. 1979. *Information Retrieval*. Buttersworth, London.

Veenstra, J. B. 1998. Fast NP chunking using memory-based learning techniques. In *Proceedings of BENELEARN'98*, pages 71–78.

Veenstra, Jorn and Antal van den Bosch. 2000. Single-classifier memory-based phrase chunking. In *Proceedings of the Fourth Conference on Computational Natural Language Learning (CoNLL) and the Second Learning Language in Logic Workshop (LLL)*, pages 157–159.

Veenstra, Jorn, Antal van den Bosch, Sabine Buchholz, Walter Daelemans, and Jakub Zavrel. 2000. Memory-based word sense disambiguation. *Computers and the Humanities*, 34(1-2). Special issue on SENSEVAL, guest editors: Adam Kilgarriff and Martha Palmer.

Voorhees, E. M. and D. K. Harman, editors. 2000. *The Eighth Text REtrieval Conference (TREC-8)*, volume 500-246 of *NIST Special Publication*, Gaithersburg, MD. National Institute of Standards and Technology. Available at http://trec.nist.gov/pubs.html.

Voorhees, E. M. and D. K. Harman, editors. 2001. *The Ninth Text REtrieval Conference (TREC-9)*, volume 500-249 of *NIST Special Publication*, Gaithersburg, MD. National Institute of Standards and Technology. Available at http://trec.nist.gov/pubs.html.

Voorhees, E. M. and D. K. Harman, editors. 2002. *The Tenth Text REtrieval Conference (TREC 2001)*, volume 500-250 of *NIST Special Publication*, Gaithersburg, MD. National Institute of Standards and Technology. Available at http://trec.nist.gov/pubs.html.

Voorhees, Ellen M. 2001. Overview of the TREC-9 question answering track. In Voorhees and Harman (Voorhees and Harman, 2001), pages 71–80. Available at http://trec.nist.gov/pubs.html.

Voorhees, Ellen M. and Dawn M. Tice. 2000. The TREC-8 question answering track evaluation. In Voorhees and Harman (Voorhees and Harman, 2000), pages 83–106. Available at http://trec.nist.gov/pubs.html.

Weiss, S. and C. Kulikowski. 1991. *Computer systems that learn.* San Mateo, CA: Morgan Kaufmann.

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

Yeh, Alexander. 2000a. Comparing two trainable grammatical relations finders. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 1146–1150.

Yeh, Alexander. 2000b. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING).*

Yeh, Alexander. 2000c. Using existing systems to supplement small amounts of annotated grammatical relations training data. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 126–132.

Yeh, Alexander and M. Vilain. 1998. Some properties of preposition and subordinate conjunction attachments. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 1436–1442.

Zavrel, J., W. Daelemans, and J. Veenstra. 1997. Resolving PP attachment ambiguities with memory-based learning. In M. Ellison, editor, *Proceedings of the First Computational Natural Language Learning workshop (CoNLL).*

Zelenko, Dmitry, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Appendix A

Tags, labels, classes, and heads

A.1 Penn Treebank II part-of-speech tags

A.1.1 Punctuation and currency tags

PoS Tag	Examples
“	‘ “
”	’ ”
(	({
)	) }
,	,
.	! . ?
:	- -- ... : ;
#	#
$	$ HK$ NZ$ US$

The # is the treebank's transcription for the £ sign.

A.1.2 Word tags

PoS Tag	Description	Examples
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential there	*there* is
FW	foreign word	d'oeuvre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener

PoS Tag	Description	Examples
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	*both* the boys
POS	possessive ending	friend*'s*
PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give *up*
TO	to	*to* go, *to* him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3rd	take
VBZ	verb, 3rd person sing. present	takes
WDT	*wh*-determiner	which
WP	*wh*-pronoun	who, what
WP$	possessive *wh*-pronoun	whose
WRB	*wh*-abverb	where, when

A.2 Penn Treebank II syntactic categories

A.2.1 Clause level

S	Simple declarative clause, i.e. one that is not introduced by a (possibly empty) subordinating conjunction or *wh*-word and that does not exhibit subject-verb inversion.
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ	Direct question introduced by a *wh*-word or *wh*-phrase. Indirect questions and relative clauses should be bracketed as `SBAR`, not `SBARQ`.

SINV	Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.
SQ	Inverted yes/no question, or main clause of a *wh*-question, following the *wh*-phrase in `SBARQ`.

A.2.2 Phrase level

ADJP	Adjective Phrase. Phrasal category headed by an adjective (including comparative and superlative adjectives). Example: *outrageously expensive*.
ADVP	Adverb Phrase. Phrasal category headed by an adverb (including comparative and superlative adverbs). Examples: *rather timidly, very well indeed, rapidly*.
CONJP	Conjunction Phrase. Used to mark certain "multi-word" conjunctions, such as *as well as, instead of*.
FRAG	Fragment.
INTJ	Interjection. Corresponds approximately to the part-of-speech tag `UH`.
LST	List marker. Includes surrounding punctuation.
NAC	Not A Constituent; used to show the scope of certain prenominal modifiers within a noun phrase.
NP	Noun Phrase. Phrasal category that includes all constituents that depend on a head noun.
NX	Used within certain complex noun phrases to mark the head of the noun phrase. Corresponds very roughly to N-bar level but used quite differently.
PP	Prepositional Phrase. Phrasal category headed by a preposition.
PRN	Parenthetical.
PRT	Particle.
QP	Quantifier Phrase (i.e., complex measure/amount phrase); used within `NP`.
RRC	Reduced Relative Clause.
UCP	Unlike Coordinated Phrase.
VP	Verb Phrase. Phrasal category headed by a verb.
WHADJP	*Wh*-adjective Phrase. Adjectival phrase containing a *wh*-adverb, as in *how hot*.
WHADVP	*Wh*-adverb Phrase. Introduces a clause with an `ADVP gap`. May be null (containing the 0 complementizer) or lexical, containing a *wh*-adverb such as *how* or *why*.
WHNP	*Wh*-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some *wh*-word, e.g. *who, which book, whose daughter, none of which*, or *how many leopards*.
WHPP	*Wh*-prepositional Phrase. Prepositional phrase containing a *wh*-noun phrase (such as *of which* or *by whose authority*) that either introduces a `PP` gap or is contained by a `WHNP`.
X	Unknown, uncertain, or unbracketable. `X` is often used for bracketing typos and in bracketing *the...the* constructions.

A.3 Penn Treebank II function tags

A.3.1 Form/function discrepancies

ADV	adverbial	marks a constituent other than ADVP or PP when it is used adverbially (e.g., NPs or free ("headless") relatives). However, constituents that themselves are modifying an ADVP generally do not get -ADV.
NOM	nominal	marks free ("headless") relatives and gerunds when they act nominally.

A.3.2 Grammatical role

DTV	dative	marks the dative object in the unshifted form of the double object construction. If the preposition introducing the "dative" object is *for*, it is considered benefactive (BNF).
LGS	logical subject	is used to mark the logical subject in passives. It attaches to the NP object of *by* and not to the PP node itself.
PRD	predicate	marks any predicate that is not VP.
PUT		marks the locative complement of *put*.
SBJ	surface subject	marks the structural surface subject of both matrix and embedded clauses, including those with null subjects.
TPC	topicalized	marks elements that appear before the subject in a declarative sentence, but in two cases only: (i) if the fronted element is associated with a *T* in the position of the gap. (ii) if the fronted element is left-dislocated (i.e, it is associated with a resumptive pronoun in the position of the gap).
VOC	vocative	marks nouns of address, regardless of their position in the sentence. It is not coindexed to the subject and does not get -TPC when it is sentence-initial.

A.3.3 Adverbials

BNF	benefactive	marks the beneficiary of an action (attaches to NP or PP). This tag is used only when (1) the verb can undergo dative shift and (2) the prepositional variant (with the same meaning) uses *for*. The prepositional objects of dative-shifting verbs with other prepositions than *for* (such as *to* or *of*) are annotated -DTV.
DIR	direction	marks adverbials that answer the questions "from where?" and "to where?". It implies motion, which can be metaphorical as in "...rose 5 pts. to 57 1/2" or "increased 70% to 5.8 billion yen". -DIR is most often used with verbs of motion/transit and financial verbs.

EXT	extent	marks adverbial phrases that describe the spatial extent of an activity. `-EXT` was incorporated primarily for cases of movement in financial space, but is also used in analogous situations elsewhere.
LOC	locative	marks adverbials that indicate place/setting of the event. `-LOC` may also indicate metaphorical location, e.g. *amongst yourselves; a drop in domestic truck sales*. There is likely to be some variation in the use of `-LOC` due to differing annotator interpretations. In cases where the annotator is faced with choosing between `-LOC` or `-TMP`, the default is `-LOC`.
MNR	manner	marks adverbials that indicate manner, including instrument phrases.
PRP	purpose or reason	marks purpose or reason clauses and `PP`s.
TMP	temporal	marks temporal or aspectual adverbials that answer the questions when, how often, or how long. It has some uses that are not strictly adverbial, such as with dates that modify other `NP`s.

A.3.4 Miscellaneous

CLR	closely related	marks constituents that occupy some middle ground between argument and adjunct of the verb phrase. These roughly correspond to "predication adjuncts", prepositional ditransitives, and some "phrasal verbs", as defined in Quirk et al. (1985). The precise meaning of `-CLR` depends somewhat on the category of its phrase: on `S` or `SBAR` — These categories are usually arguments, so the `-CLR` tag indicates that the clause is more adverbial than normal clausal arguments. The most common case is the infinitival semicomplement of *use*, but there are a variety of other cases. On `PP, ADVP, SBAR-PRP`, etc. — On categories that are ordinarily interpreted as (adjunct) adverbials, `-CLR` indicates a somewhat closer relationship to the verb.
CLF	cleft	marks *it*-clefts ("true" clefts) and may be added to the labels `S, SINV`, or `SQ`.
HLN	headline	marks headlines and datelines. Note that headlines and datelines always constitute a unit of text that is structurally independent from the following sentence.
TTL	title	is attached to the top node of a title when this title appears inside running text. `-TTL` implies `-NOM`. The internal structure of the title is bracketed as usual.

A.4 Classes and their frequency in the unrestricted tenfold cross validation material

13	ADJP
8	ADJP-ADV
11	ADJP-CLR
1	ADJP-MNR
57	ADJP-OBJ
3472	ADJP-PRD
16	ADJP-PRD-TPC;T-ADJP-PRD
2	ADJP-PRD/S
103	ADJP-PRD/S-ADV
29	ADJP-PRD/S-CLR
1	ADJP-PRD/S-LOC
1	ADJP-PRD/S-MNR
51	ADJP-PRD/S-OBJ
1	ADJP-PRD/S/SBAR-CLR
10	ADJP-PRD/UCP-PRD
1	ADJP-PRD;T-ADJP-PRD
1	ADJP-TPC;T-ADJP-PRD
1	ADJP-TPC;T-ADJP-PRD/S/SBAR-OBJ
2	ADJP/UCP
1	ADJP/UCP-ADV
1	ADJP/UCP-CLR
1	ADJP/UCP-MNR
31	ADJP/UCP-PRD
4	ADJP/UCP-PRD/S-ADV
1	ADJP;T-ADJP-PRD
3196	ADVP
425	ADVP-CLR
1	ADVP-CLR-MNR
1	ADVP-CLR-TPC;T-ADVP-CLR
312	ADVP-DIR
3	ADVP-DIR-CLR
40	ADVP-EXT
215	ADVP-LOC
27	ADVP-LOC-CLR
39	ADVP-LOC-PRD
16	ADVP-LOC-PRD-TPC;T-ADVP-LOC-PRD
1	ADVP-LOC-TPC-PRD;T-ADVP-LOC-PRD
1	ADVP-LOC-TPC;T-ADVP-LOC-TPC
712	ADVP-MNR
3	ADVP-MNR-CLR
4	ADVP-MNR/UCP
215	ADVP-PRD
1	ADVP-PRD-LOC
3	ADVP-PRD-TPC;T-ADVP-PRD
6	ADVP-PRD/S-ADV
2	ADVP-PRD/S-CLR
1	ADVP-PRD/S-OBJ
5	ADVP-PRP
13	ADVP-PUT
1863	ADVP-TMP
6	ADVP-TMP-CLR
5	ADVP-TMP-PRD
1	ADVP-TMP-PRD/S-PRD
1	ADVP-TMP-TPC
2	ADVP-TMP/UCP
1	ADVP-TPC-PRD;T-ADVP-PRD
1	ADVP-TPC;T-ADVP
3	ADVP/UCP
1	ADVP/UCP-ADV
1	ADVP/UCP-LOC
3	ADVP/UCP-MNR
8	ADVP/UCP-PRD
4	ADVP/UCP-TMP
50	CONJP
41	INTJ
2	INTJ-CLR
1	INTJ/FRAG
1	INTJ/FRAG-PRD
1	INTJ/FRAG-TPC;T-FRAG
22	LST
93	NP
140	NP-ADV
1	NP-ADV/ADVP
1	NP-ADV/ADVP-PRD
7	NP-BNF
91	NP-CLR
9	NP-DIR
1109	NP-EXT
12	NP-LOC
4	NP-LOC-CLR
4	NP-LOC-PRD
26	NP-MNR
20898	NP-OBJ
3384	NP-PRD
1	NP-PRD-TPC/S/PRN
1	NP-PRD-TPC;T-NP-PRD
3	NP-PRD/S
25	NP-PRD/S-ADV
28	NP-PRD/S-OBJ
3	NP-PRD;T-NP-PRD
35141	NP-SBJ
2	NP-SBJ-TTL
2948	NP-SBJ;T-NP-OBJ
5	NP-SBJ;T-NP-OBJ;T-NP-SBJ

3	NP-SBJ;T-NP-PRD
3	NP-SBJ;T-NP-SBJ
1724	NP-TMP
56	NP-TMP-CLR
4	NP-TMP-PRD
1	NP-TPC
4	NP-TPC;T-NP-OBJ
1	NP-TPC;T-NP-PRD
5	NP-TTL
2	NP-TTL-PRD
5	NP-TTL-PRD/S-OBJ
15	NP-TTL-SBJ
3	NP-TTL-SBJ;T-NP-OBJ
1	NP-TTL-SBJ;T-NP-TTL
1	NP-TTL-TPC
8	NP-VOC
6	NP/UCP
1	NP/UCP-CLR
1	NP/UCP-LOC
19	NP/UCP-PRD
2	NP/UCP-PRD/S-ADV
3	NP/UCP-TMP
2	NP/WHNP/SBAR-NOM
2	NP/WHNP/SBAR-OBJ
9	NP;T-NP-OBJ
4225	PP
14	PP-BNF
6143	PP-CLR
7	PP-CLR-LOC
1	PP-CLR-TPC;T-PP-CLR
1	PP-CLR/UCP
2	PP-CLR;T-PP-CLR
2826	PP-DIR
7	PP-DIR-CLR
1	PP-DIR-PRD
1	PP-DIR/UCP
216	PP-DTV
131	PP-EXT
1367	PP-LGS
3230	PP-LOC
405	PP-LOC-CLR
3	PP-LOC-CLR-TPC;T-PP-LOC-CLR
1	PP-LOC-MNR
271	PP-LOC-PRD
20	PP-LOC-PRD-TPC;T-PP-LOC-PRD
2	PP-LOC-PRD/S
3	PP-LOC-PRD/S-ADV
1	PP-LOC-PRD/S-CLR
1	PP-LOC-PRD/S-OBJ
2	PP-LOC-PRD/UCP-PRD
1	PP-LOC-PRD;T-PP-PRD-LOC
1	PP-LOC-TPC-PRD;T-PP-LOC-PRD
2	PP-LOC/UCP
1	PP-LOC/UCP-ADV
6	PP-LOC/UCP-PRD
2	PP-LOC;T-PP-LOC
746	PP-MNR
2	PP-MNR-PRD
1	PP-MNR/UCP-CLR
1	PP-NOM
1	PP-NOM/NP-OBJ
432	PP-PRD
4	PP-PRD-LOC
1	PP-PRD-LOC/S-OBJ
4	PP-PRD-TPC;T-PP-PRD
2	PP-PRD/S
5	PP-PRD/S-ADV
1	PP-PRD/S-ADV/PRN
4	PP-PRD/S-OBJ
1	PP-PRD/UCP-PRD
1	PP-PRD;T-PP-PRD
518	PP-PRP
1	PP-PRP-CLR
9	PP-PRP-PRD
1	PP-PRP/UCP
98	PP-PUT
2	PP-SBJ
3656	PP-TMP
19	PP-TMP-CLR
18	PP-TMP-PRD
1	PP-TMP-TPC
1	PP-TMP-TPC;T-PP-TMP
1	PP-TMP/UCP
20	PP-TPC
1	PP-TPC-CLR;T-PP-CLR
1	PP-TPC-LOC-PRD;T-PP-LOC-PRD
1	PP-TPC-PRD;T-PP-PRD
1	PP-TPC;T-PP
2	PP-TPC;T-PP-PRD
1	PP-TTL-PRD
10	PP/UCP
2	PP/UCP-LOC
4	PP/UCP-MNR
12	PP/UCP-PRD
2	PP/UCP-PRD/S-ADV
7	PP/UCP-PRP
4	PP/UCP-TMP
5	PP;T-PP
1	PP;T-PP-PUT
1405	PRT
34	SBAR
1443	SBAR-ADV
1	SBAR-ADV-TPC
2	SBAR-ADV/FRAG

13	SBAR-ADV/PRN
1	SBAR-ADV/SQ-TPC;T-SQ-OBJ
21	SBAR-CLR
22	SBAR-MNR
1	SBAR-MNR/PRN
1	SBAR-MNR/UCP
6	SBAR-NOM
1	SBAR-NOM-SBJ
1953	SBAR-OBJ
1	SBAR-OBJ;T-SBAR
144	SBAR-PRD
1	SBAR-PRD/UCP
415	SBAR-PRP
13	SBAR-PRP-PRD
15	SBAR-SBJ
625	SBAR-TMP
2	SBAR-TMP-PRD
1	SBAR-TMP/FRAG
1	SBAR-TMP/UCP-ADV
7	SBAR/PRN
1	SBAR/SQ
2	SBAR/UCP
1	SBAR/UCP-PRD
4	SBAR/UCP-PRP
1	SBAR;T-SBAR-OBJ
1	T-ADJP-OBJ
19	T-ADJP-PRD
20	T-ADVP
8	T-ADVP-DIR
4	T-ADVP-EXT
163	T-ADVP-LOC
4	T-ADVP-LOC-CLR
21	T-ADVP-LOC-PRD
102	T-ADVP-MNR
6	T-ADVP-PRD
59	T-ADVP-PRP
1	T-ADVP-PUT
661	T-ADVP-TMP
3	T-ADVP-TMP-CLR
1	T-ADVP-TMP-PRD
1	T-FRAG
1	T-NP-EXT
514	T-NP-OBJ
482	T-NP-OBJ;T-NP-SBJ
40	T-NP-PRD
6015	T-NP-SBJ
2	T-NP-SBJ;T-NP-OBJ
3	T-NP-TMP
34	T-PP
17	T-PP-CLR
4	T-PP-DIR
3	T-PP-EXT
74	T-PP-LOC
1	T-PP-LOC-CLR
6	T-PP-MNR
1	T-PP-PRP
22	T-PP-TMP
1	T-S-ADV
71	T-S-OBJ
4	T-S-PRP
324	T-S/SBAR-OBJ
1	T-S/SBAR-PRD
1	T-SBAR-ADV
2	T-SBAR-OBJ
7	T-SBARQ-OBJ
5	T-SINV/SBAR-OBJ
1	T-SQ-OBJ
106	VP
736	VP-OBJ
1	VP-OBJ;T-VP/S-OBJ
1	VP-TPC/SINV
1	VP-TPC/SINV-TPC;T-SINV/SBAR-OBJ
10	VP/PRN
491	VP/S
1150	VP/S-ADV
9	VP/S-ADV/PRN
1	VP/S-ADV/SBAR-PRD
1	VP/S-ADV/UCP
3	VP/S-CLF
1	VP/S-CLF-TPC;T-S-CLF
418	VP/S-CLR
1	VP/S-CLR/UCP
20	VP/S-MNR
2	VP/S-MNR-CLR
6	VP/S-NOM
10	VP/S-NOM-PRD
136	VP/S-NOM-SBJ
1	VP/S-NOM/FRAG
2	VP/S-NOM/NP-OBJ
2	VP/S-NOM/NP-SBJ
1469	VP/S-OBJ
125	VP/S-PRD
684	VP/S-PRP
33	VP/S-PRP-CLR
1	VP/S-PRP-PRD
2	VP/S-PRP-TPC
1	VP/S-PRP/PRN
1	VP/S-PRP/UCP
1	VP/S-PRP/UCP-CLR
7	VP/S-SBJ
3	VP/S-TMP
2	VP/S-TMP/PRN
24	VP/S-TPC
1130	VP/S-TPC;T-S-OBJ

873 VP/S-TPC;T-S/SBAR-OBJ
1 VP/S-TTL-PRD
1 VP/S/INTJ
371 VP/S/PRN
1 VP/S/SBAR
22 VP/S/SBAR-ADV
1 VP/S/SBAR-CLR
2 VP/S/SBAR-MNR
1 VP/S/SBAR-NOM-SBJ
3100 VP/S/SBAR-OBJ
16 VP/S/SBAR-PRD
7 VP/S/SBAR-PRP
1 VP/S/SBAR-PRP-PRD
43 VP/S/SBAR-TMP
1 VP/S/SQ-OBJ
2 VP/S/UCP
1 VP/S/UCP-PRD
2 VP/S/UCP-PRP
27 VP/S;T-S-OBJ
5 VP/S;T-S/SBAR-OBJ
16 VP/SINV
1 VP/SINV-OBJ
2 VP/SINV-TPC;T-SINV-OBJ
1 VP/SINV-TPC;T-SINV/SBAR-OBJ
56 VP/SINV/PRN
15 VP/SINV/SBAR-ADV
1 VP/SINV/SBAR-OBJ
4 VP/SQ
5 VP/SQ-OBJ
2 VP/SQ-TPC;T-SQ-OBJ
25 VP/UCP-PRD
1 WHADJP/SBAR-ADV
2 WHADJP/SBAR-NOM
8 WHADJP/SBAR-OBJ
1 WHADJP/SBAR-PRD
1 WHADJP/SBAR-TPC;T-SBAR-OBJ
1 WHADVP
7 WHADVP/SBAR
9 WHADVP/SBAR-ADV
1 WHADVP/SBAR-DIR-TPC;T-SBAR-DIR
14 WHADVP/SBAR-LOC
1 WHADVP/SBAR-LOC-CLR
7 WHADVP/SBAR-LOC-PRD
1 WHADVP/SBAR-MNR
15 WHADVP/SBAR-NOM
2 WHADVP/SBAR-NOM-PRD
4 WHADVP/SBAR-NOM-SBJ
1 WHADVP/SBAR-NOM/NP-OBJ
118 WHADVP/SBAR-OBJ
18 WHADVP/SBAR-PRD
3 WHADVP/SBAR-PRP-PRD
3 WHADVP/SBAR-PUT
2 WHADVP/SBAR-SBJ
476 WHADVP/SBAR-TMP
1 WHADVP/SBAR-TMP-CLR
1 WHADVP/SBAR-TMP-PRD
1 WHADVP/SBAR-TMP/FRAG
1 WHADVP/SBAR-TPC
1 WHADVP/SBAR-TTL
1 WHADVP/SBAR/UCP
1 WHADVP/SBAR/UCP-PRD
1 WHADVP/SBARQ
4 WHADVP/SBARQ-OBJ
1 WHADVP/SBARQ-PRD
3 WHADVP/SBARQ-TPC;T-SBARQ-OBJ
2 WHADVP/SBARQ;T-SBARQ-OBJ
2 WHADVP/UCP;T-ADVP-TMP
16 WHNP/SBAR
29 WHNP/SBAR-ADV
1 WHNP/SBAR-ADV/PRN
83 WHNP/SBAR-NOM
17 WHNP/SBAR-NOM-PRD
55 WHNP/SBAR-NOM-SBJ
1 WHNP/SBAR-NOM-TPC;T-NP-OBJ
2 WHNP/SBAR-NOM-TPC;T-SBAR-NOM
8 WHNP/SBAR-NOM/NP-OBJ
2 WHNP/SBAR-NOM/NP-PRD
1 WHNP/SBAR-NOM/NP-SBJ
140 WHNP/SBAR-OBJ
10 WHNP/SBAR-PRD
2 WHNP/SBAR-SBJ
1 WHNP/SBAR/FRAG-TPC;T-FRAG
1 WHNP/SBAR/PRN
1 WHNP/SBAR;T-SBAR-OBJ
2 WHNP/SBARQ
3 WHNP/SBARQ-OBJ
1 WHNP/SBARQ-PRD
1 WHNP/SBARQ/UCP
1 WHNP/SBARQ/UCP-TPC;T-UCP
2 WHPP/SBAR-ADV
7 WHPP/SBAR-OBJ
1 WHPP/SBAR-TMP

A.5 The head table

```
$head{'ADJP'}='JJ|RB|VB|IN|UH|FW|RP|\$|#|DT|NN';
$head{'ADVP'}='RB|IN|TO|DT|PDT|JJ|RP|FW|LS|UH|NN|CD|VB';
$head{'CONJP'}='CC|IN|RB';
$head{'INTJ'}='UH|RB|NN|VB|FW|JJ';
$head{'LST'}='LS|JJ|:';
$head{'NAC'}='NN';
$head{'NOLABEL'}='[A-Z]';
$head{'NP'}='NN|CD|PRP|JJ|DT|EX|IN|RB|VB|FW|SYM|UH|WP|WDT';
$head{'NX'}='NN|CD|PRP|JJ|DT|EX|FW|SYM|UH|WP|WDT';
$head{'PP'}='IN|TO|RB|VBG|VBN|JJ|RP|FW';
$head{'PRT'}='RP|IN|RB|JJ';
$head{'QP'}='CD|DT|NN|JJ';
$head{'SBAR'}='IN|DT|WDT';
$head{'UCP'}='JJ|NN|VB|CD';
$head{'VP'}='VB|MD|TO|JJ|NN|POS|FW|SYM';
$head{'WHADJP'}='JJ';
$head{'WHADVP'}='WRB|IN|RB|WDT';
$head{'WHNP'}='WDT|WP|CD|DT|IN|NN|JJ|RB';
$head{'WHPP'}='IN|TO';

$headcat{'ADJP'}='ADJP';
$headcat{'ADVP'}='ADVP|.*-ADV';
$headcat{'CONJP'}='CONJP';
$headcat{'FRAG'}='FRAG|INTJ|S|VP';
$headcat{'INTJ'}='S|VP|INTJ';
$headcat{'LST'}='LST';
$headcat{'NOLABEL'}='[A-Z]';
$headcat{'NP'}='NP|NX|.*-NOM';
$headcat{'NX'}='NX';
$headcat{'PP'}='PP';
$headcat{'PRN'}='S|VP';
$headcat{'PRT'}='PRT';
$headcat{'RRC'}='S|VP';
$headcat{'S'}='S$|VP|.*-PRD';
$headcat{'SBAR'}='SBAR|S|WH';
$headcat{'SBARQ'}='SBARQ|SQ|WH';
$headcat{'SINV'}='SINV|VP|SBAR';
$headcat{'SQ'}='SQ|VP|S|WH';
$headcat{'UCP'}='[A-Z]+P(-[-A-Z]+)?$|S';
$headcat{'VP'}='VP';
$headcat{'WHADJP'}='WHADJP|ADJP';
$headcat{'WHADVP'}='WHADVP';
$headcat{'WHNP'}='WHNP|NP';
$headcat{'WHPP'}='WHPP';
$headcat{'X'}='S|[A-Z]+P(-[-A-Z]+)?$';
```

Appendix B

Summary

This thesis is about automatically finding grammatical relations (GR), such as subject, direct object, temporal or locative adjunct in English sentences. The grammatical relations to be found are based on the annotations in the Wall Street Journal Corpus of the Penn Treebank II and are restricted to relations to verbs. The relation finder uses a supervised machine learning algorithm called Memory-Based Learning (MBL). It operates on the output of a Memory-Based part-of-speech tagger and chunker. Together these modules form the Memory-Based Shallow Parser (MBSP).

Input to a Memory-Based Learner must be in the form of a fixed number of feature-value pairs whereas the output of the tagger and chunker has no fixed maximum length but an internal structure (chunks have types and consist of sequences of words, which have PoS tags). The challenge thus consists of finding a suitable selection and representation of this information in the required format. Research questions are: "What information is useful for performing the task?" and "How can this information best be used by a Memory-Based Learner?". We answer these questions both quantitatively, by performing learning experiments and comparing performance, learning speed and memory requirements with different features and algorithmic settings, and qualitatively, by extensive error analysis.

The thesis is structured as follows. Chapter 1 contains the introduction. Chapter 2 sketches the theoretical background of grammatical relations: how they are defined, what phenomena are described by them and how they are annotated in treebanks. It also reviews work that is related to determining GRs. The review shows that diverse types of information are relevant to the task and that this information can be represented in different ways.

Chapter 3 describes the original Penn Treebank data and how we extracted chunks, heads and GRs from it. This conversion is complex because this information is not explicitly contained in the treebank. The second part of the chapter explains the general set-up for the experiments in the following two chapters, which form the core of this thesis.

Chapter 4 introduces the MBL algorithms that are used in this thesis and explains the various parameters. The second part of the chapter applies these algorithms with different

parameter settings to the GR data. The most interesting improvement over the default setting is achieved through the Modified Value Difference Metric (MVDM) which models task specific similarity between feature values. Our analyses show that this allows the algorithm to implicitly learn hierarchies of PoS, syntactic and semantic similarity between words and a non-linear measure of "distance" in a sentence. With the best parameter setting, the best MBL algorithm (IB1-IG) achieves an F_β of 80.09.

Chapter 5 systematically tries to improve performance and/or speed and memory requirements by deleting superfluous features and adding useful new ones and by trying different representations of the same information. The most interesting new information involves sequences of PoS or chunks. Representing information from sequences is especially challenging in a propositional format. We present two possibilities: using MVDM on sequences regarded as atomic values and using numeric features. We also show how information from words that are not semantic but syntactic heads can help the learner. Even knowledge about the absence of such words in a chunk can be relevant. With the best feature selection and representation, the algorithm achieves an F_β of 83.10.

Chapter 6 treats several practical issues. We show that training the learner with a more fine-grained class definition than is actually needed does not harm results on a coarser-grained evaluation. Training on material that is slightly different from test material, however, decreases performance. This is shown by experiments with two different corpora and with manually annotated versus automatically tagged and chunked text. With the latter, F_β is 72.59. In the last section of Chapter 6, MBSP is compared to two related systems. It performs better than Memory-Based Sequence Learning on finding unlabeled dependencies, but worse than the relation finder of Carroll, Minnen, and Briscoe (1998) on finding GRs according to their definition. However, many errors are due to the difficult mapping from our GRs to theirs and to the fact that MBSP finds relations to verbs only.

In Chapter 7, MBSP is integrated into a Question Answering prototype called Shapaqa. We describe the version with which we achieved a Mean Reciprocal Rank of 0.21 in the Question Answering track of TREC-10 and the version that is accessible via the internet. This application demonstrates that the performance that can be achieved by a fast version of the parser is sufficient for Question Answering when large numbers of documents are available. This is typically the case on the World Wide Web.

Chapter 8 summarizes the answers to our research questions and suggests future research. Information that is relevant for the task of finding GRs includes: syntactic and semantic heads of chunks, their part-of-speech, chunk types, the context of the focus and the verb, linear order, semantic types, subcategorization and lemmas, punctuation, verb chunk properties (such as passivity), and information on typical tagger/chunker errors. Much of this information can best be used by MBL with the MVDM metric. For sequences, numeric features are an alternative. Distributing information over several features or representing the same information redundantly in several features does not necessarily harm performance. Future research directions include unknown words, relations to non-verbs, automatic feature construction, and optimal work separation between the MBSP modules.

Appendix C

Samenvatting

Dit proefschrift beschrijft hoe een computer grammaticale relaties (GR), zoals onderwerp, lijdend voorwerp, tijd- of plaatsbepaling in Engelse zinnen kan vinden. De GRs zijn afgeleid van de annotaties in het Wall Street Journal corpus uit de Penn Treebank II, maar beperkt tot relaties tot werkwoorden. De relatievinder is een toepassing van een algemeen gesuperviseerd automatisch leeralgoritme: Memory-Based Learning (MBL). De module werkt op de uitvoer van een Memory-Based tagger en chunker. Samen vormen deze modules de Memory-Based Shallow Parser (MBSP).

MBL vraagt invoer in de vorm van een vast aantal feature-waarde-paren. Daarentegen heeft de uitvoer van de tagger en chunker een interne structuur en een variabele lengte zonder vast maximum. Belangrijke onderzoeksvragen zijn daarom: "Welke informatie is nuttig voor de taak?" en "Hoe kan deze informatie het beste door MBL gebruikt worden?". Deze vragen beantwoorden wij zowel kwantitatief, door een vergelijking van het succes, de snelheid en het geheugengebruik van leerexperimenten met verschillende features en algoritmeparameters, als kwalitatief, door uitgebreide foutenanalyses.

Hoofdstuk 1 bevat de inleiding. Hoofdstuk 2 beschrijft de theoretische achtergrond van GRs: hoe deze gedefinieerd worden, welke fenomenen zij beschrijven en hoe zij in boombanken geannoteerd zijn. Verder geven wij een overzicht van ander werk over het bepalen van GRs waaruit blijkt dat veel verschillende typen informatie hierbij een rol spelen en dat deze informatie op verschillende manieren gerepresenteerd wordt.

Hoofdstuk 3 beschrijft de Penn Treebank data en hoe wij hieruit chunks, hoofden en GRs afleiden. Deze omzetting is vrij complex omdat de informatie niet expliciet in de boombank aanwezig is. De tweede helft van het hoofdstuk legt de opzet uit van de leerexperimenten die in de daaropvolgende twee hoofdstukken beschreven worden. Deze hoofdstukken vormen de kern van het proefschrift.

Hoofdstuk 4 introduceert de MBL algoritmes en hun parameters en test deze op de GR data. De meest interessante verbetering wordt bereikt met de Modified Value Difference Metric (MVDM). Deze metriek berekent hoeveel twee featurewaarden op elkaar lijken met

betrekking tot een specifieke taak. Hierdoor leert het algoritme impliciet woordsoorthiërarchieën, syntactische en semantische gelijkenis van woorden en een niet-lineaire maat voor de afstand tussen woorden in een zin. Met de beste parameterinstelling bereikt het beste MBL algoritme (te weten IB1-IG) een F_β-waarde van 80.09.

In Hoofdstuk 5 verbeteren wij de performantie en/of snelheid en geheugengebruik van de relatievinder door overbodige features weg te laten, nuttige nieuwe toe te voegen en verschillende representaties van dezelfde informatie te testen. De meest interessante nieuwe informatie bestaat uit sequenties van woordsoorten of chunks. Wij laten twee mogelijkheden zien hoe deze sequentiële informatie in het vereiste propositionele formaat gerepresenteerd kan worden: als atomaire featurewaarden met gebruik van MVDM of door middel van een aantal numerieke features. We tonen ook hoe woorden die geen semantische maar syntactische hoofden zijn, het leerproces verbeteren. Zelfs kennis over de afwezigheid van deze woorden in een chunk kan nuttig zijn. Met de beste featureselectie en -representatie bereikt het algoritme een F_β-waarde van 83.10.

In Hoofdstuk 6 laten we zien dat de nauwkeurigheid van de voorspelling niet verslechtert als het algoritme getraind wordt met klassen die gedetaileerder zijn dan die waarop we uiteindelijk evalueren. Alleen als het trainingsmateriaal afwijkt van het testmateriaal gaat de performantie achteruit. Voorbeelden hiervan zijn verschillende corpora en handmatig versus automatisch getagde en gechunkte tekst. In het laatste geval is de F_β-waarde 72.59. In de laatste paragraaf van hoofdstuk 6 vergelijken wij de MBSP met twee andere relatievindsystemen. MBSP scoort beter dan Memory-Based Sequence Learning op ongelabelde dependenties maar minder goed dan het systeem van Carroll, Minnen, and Briscoe (1998) op GRs volgens hun definitie. De meeste fouten ontstaan doordat ons systeem alleen relaties tot werkwoorden vindt en door de moeilijke overzetting van onze relaties naar de hunne.

In Hoofdstuk 7 beschrijven wij het prototype van een vraag-beantwoord-systeem dat op MBSP gebaseerd is. We beschrijven de versie die via internet toegankelijk is en de versie die in de Question Answering track van de tiende TREC conferentie een score van 0.21 behaalde. Deze toepassing bewijst dat de performantie van een snelle versie van de parser goed genoeg is om antwoorden te vinden mits er genoeg documenten ter beschikking staan. Dit laatste is bij uitstek het geval op het World Wide Web.

Hoofdstuk 8 vat de antwoorden op onze onderzoeksvragen samen en suggereert mogelijk verder onderzoek. Relevante informatie voor het vinden van GRs is: syntactische en semantische hoofden van chunks, hun woordsoort, chunktypen, de context van de focus en van het werkwoord, de lineaire volgorde, semantische typen, subcategorisatie en lemmata, interpunctie, eigenschappen van werkwoordschunks (zoals passiviteit) en informatie over typische tagger- en chunkerfouten. De meeste informatie kan het beste met MVDM gebruikt worden. Voor sequenties vormen numerieke features een alternatief. Het distribueren van informatie over meerdere features of het redundant representeren van informatie hoeft de performantie niet te schaden. Als onderwerpen voor verder onderzoek zien wij onbekende woorden, relaties tot niet-werkwoorden, automatische featureconstructie en een optimale werkverdeling tussen de MBSP modules.