A non-projective dependency parser

Pasi Tapanainen and **Timo Järvinen**
University of Helsinki, Department of General Linguistics
Research Unit for Multilingual Language Technology
P.O. Box 4, FIN-00014 University of Helsinki, Finland
{Pasi.Tapanainen,Timo.Jarvinen}@ling.Helsinki.fi

Abstract

We describe a practical parser for unrestricted dependencies. The parser creates links between words and names the links according to their syntactic functions. We first describe the older Constraint Grammar parser where many of the ideas come from. Then we proceed to describe the central ideas of our new parser. Finally, the parser is evaluated.

1 Introduction

We are concerned with surface-syntactic parsing of running text. Our main goal is to describe syntactic analyses of sentences using dependency links that show the head-modifier relations between words. In addition, these links have labels that refer to the syntactic function of the modifying word. A simplified example is in Figure 1, where the link between *I* and *see* denotes that *I* is the modifier of *see* and its syntactic function is that of subject. Similarly, *a* modifies *bird*, and it is a determiner.

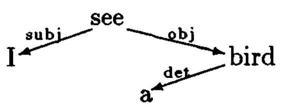

Figure 1: Dependencies for sentence *I see a bird.*

First, in this paper, we explain some central concepts of the Constraint Grammar framework from which many of the ideas are derived. Then, we give some linguistic background to the notations we are using, with a brief comparison to other current dependency formalisms and systems. New formalism is described briefly, and it is utilised in a small toy grammar to illustrate how the formalism works. Finally, the real parsing system, with a grammar of some 2 500 rules, is evaluated.

The parser corresponds to over three man-years of work, which does not include the lexical analyser and the morphological disambiguator, both parts of the existing English Constraint Grammar parser (Karlsson et al., 1995). The parsers can be tested via WWW[1].

2 Background

Our work is partly based on the work done with the *Constraint Grammar* framework that was originally proposed by Fred Karlsson (1990). A detailed description of the English Constraint Grammar (ENGCG) is in Karlsson et al. (1995). The *basic rule types* of the Constraint Grammar (Tapanainen, 1996)[2] are `REMOVE` and `SELECT` for discarding and selecting an alternative reading of a word. Rules also have contextual tests that describe the condition according to which they may be applied. For example, the rule

REMOVE (V)   IF (-1C DET);

discards a verb (`V`) reading if the preceding word (`-1`) is unambiguously (`C`) a determiner (`DET`). More than one such test can be appended to a rule.

The rule above represents a local rule: the test checks only neighbouring words in a foreknown position before or after the target word. The test may also refer to the positions somewhere in the sentence without specifying the exact location. For instance,

SELECT (IMP) IF (NOT *-1 NOM-HEAD);

means that a nominal head (`NOM-HEAD` is a set that contains part-of-speech tags that may represent a nominal head) may not appear anywhere to the left (`NOT *-1`).

[1] at *http://www.ling.helsinki.fi/~tapanain/dg/*

[2] The CG-2 notation here (Tapanainen, 1996) is different from the former (Karlsson et al., 1995). A concise introduction to the formalism is also to be found in Samuelsson et al. (1996) and Hurskainen (1996).

This "anywhere" to the left or right may be restricted by BARRIERs, which restrict the area of the test. Basically, the barrier can be used to limit the test only to the current clause (by using clause boundary markers and "stop words") or to a constituent (by using "stop categories") instead of the whole sentence. In addition, another test may be added relative to the unrestricted context position using keyword LINK. For example, the following rule discards the syntactic function[3] @I-OBJ (*indirect object*):

```
REMOVE (@I-OBJ)
    IF (*-1C VFIN BARRIER SVOO
        LINK NOT 0 SVOO);
```

The rule holds if the closest finite verb to the left is unambiguously (C) a finite verb (VFIN), and there is no ditransitive verb or participle (subcategorisation SVOO) between the verb and the indirect object. If, in addition, the verb does not take indirect objects, i.e. there is no SVOO in the same verb (LINK NOT 0 SVOO), the @I-OBJ reading will be discarded.

In essence, the same formalism is used in the syntactic analysis in Järvinen (1994) and Anttila (1995). After the morphological disambiguation, all legitimate surface-syntactic labels are added to the set of morphological readings. Then, the syntactic rules discard contextually illegitimate alternatives or select legitimate ones.

The *syntactic tagset* of the Constraint Grammar provides an underspecific dependency description. For example, labels for functional heads (such as @SUBJ, @OBJ, @I-OBJ) mark the word which is a head of a noun phrase having that function in the clause, but the parent is not indicated. In addition, the representation is shallow, which means that, e.g., objects of infinitives and participles receive the same type of label as objects of finite verbs. On the other hand, the non-finite verb forms functioning as objects receive only verbal labels.

When using the grammar formalism described above, a considerable amount of syntactic ambiguity can not be resolved reliably and is therefore left pending in the parse. As a consequence, the output is not optimal in many applications. For example, it is not possible to reliably pick head–modifier pairs from the parser output or collect arguments of verbs, which was one of the tasks we originally were interested in.

To solve the problems, we developed a more powerful rule formalism which utilises an explicit dependency representation. The basic Constraint Grammar idea of introducing the information in a piecemeal fashion is retained, but the integration of different pieces of information is more efficient in the new system.

3 Dependency grammars in a nutshell

Our notation follows the classical model of dependency theory (Heringer, 1993) introduced by Lucien Tesnière (1959) and later advocated by Igor Mel'čuk (1987).

3.1 Uniqueness and projectivity

In Tesnière's and Mel'čuk's dependency notation every element of the dependency tree has a unique head. The verb serves as the head of a clause and the top element of the sentence is thus the main verb of the main clause. In some other theories, e.g. Hudson (1991), several heads are allowed.

Projectivity (or adjacency[4]) was not an issue for Tesnière (1959, ch. 10), because he thought that the linear order of the words does not belong to the syntactic level of representation which comprises the structural order only.

Some early formalisations, c.f. (Hays, 1964), have brought the strict projectivity (context-free) requirement into the dependency framework. This kind of restriction is present in many dependency-based parsing systems (McCord, 1990; Sleator and Temperley, 1991; Eisner, 1996).

But obviously any recognition grammar should deal with non-projective phenomena to the extent they occur in natural languages as, for example, in the analysis shown in Figure 2. Our system has no in-built restrictions concerning projectivity, though the formalism allows us to state when crossing links are not permitted.

We maintain that one is generally also interested in the linear order of elements, and therefore it is presented in the tree diagrams. But, for some purposes, presenting all arguments in a canonical order might be more adequate. This, however, is a matter of output formatting, for which the system makes several options available.

3.2 Valency and categories

The verbs (as well as other elements) have a valency that describes the number and type of the modifiers they may have. In valency theory, usually, complements (obligatory) and adjuncts (optional) are distinguished.

[3]The convention in the Constraint Grammar is that the tags for syntactic functions begin with the @-sign.

[4]D is adjacent to H provided that every word between D and H is a subordinate of H (Hudson, 1991).

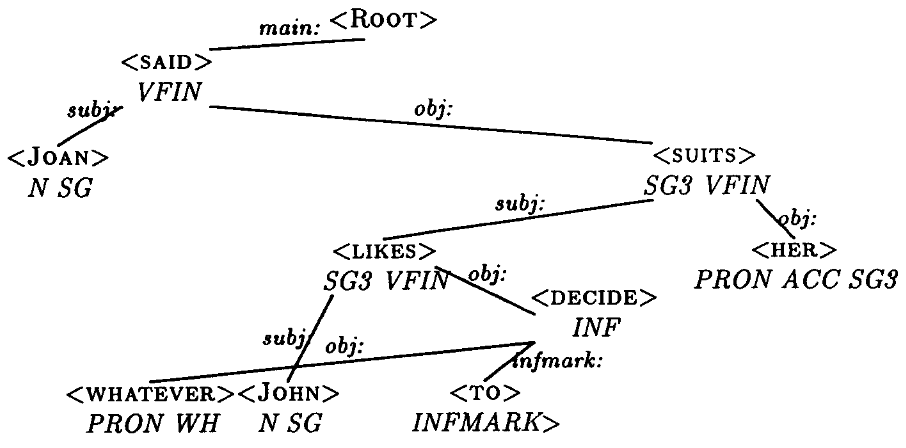

Figure 2: A dependency structure for the sentence: *Joan said whatever John likes to decide suits her.*

Our notation makes a difference between valency (rule-based) and subcategorisation (lexical): the valency tells which arguments are expected; the subcategorisation tells which combinations are legitimate. The valency merely provides a possibility to have an argument. Thus, a verb having three valency slots may have e.g. subcategorisation SVOO or SVOC. The former denotes: Subject, Verb, indirect Object and Object, and the latter: Subject, Verb, Object and Object Complement. The default is a nominal type of complement, but there might also be additional information concerning the range of possible complements, e.g., the verb *say* may have an object (SVO), which may also be realised as a *to*-infinitive clause, WH-clause, *that*-clause or quote structure.

The adjuncts are not usually marked in the verbs because most of the verbs may have e.g. spatio-temporal arguments. Instead, adverbial complements and adjuncts that are typical of particular verbs are indicated. For instance, the verb *decide* has the tag <P/on> which means that the prepositional phrase *on* is typically attached to it.

The distinction between the complements and the adjuncts is vague in the implementation; neither the complements nor the adjuncts are obligatory.

4 Introducing the dependencies

Usually, both the dependent element and its head are implicitly (and ambiguously) present in the Constraint Grammar type of rule. Here, we make this dependency relation explicit. This is done by declaring the heads and the dependents (complement or modifier) in the context tests.

For example, the subject label (@SUBJ) is chosen and marked as a dependent of the immediately following auxiliary (AUXMOD) in the following rule:

```
SELECT (@SUBJ) IF (1C AUXMOD HEAD);
```

To get the full benefit of the parser, it is also useful to name the *valency slot* in the rule. This has two effects: (1) the valency slot is unique, i.e. no more than one subject is linked to a finite verb[5], and (2) we can explicitly state in rules which kind of valency slots we expect to be filled. The rule thus is of the form:

```
SELECT (@SUBJ)
    IF (1C AUXMOD HEAD = subject);
```

The rule above works well in an unambiguous context but there is still need to specify more tolerant rules for ambiguous contexts. The rule

```
INDEX (@SUBJ)
    IF (1C AUXMOD HEAD = subject);
```

differs from the previous rule in that it leaves the other readings of the noun intact and only adds a (possible) subject dependency, while both the previous rules disambiguated the noun reading also.

But especially in the rule above, the contextual test is far from being sufficient to select the subject reading reliably. Instead, it leaves open a possibility to attach a dependency from another syntactic function, i.e. the dependency relations remain ambiguous. The grammar tries to be careful not to introduce false dependencies but for an obvious reason this is not always possible. If several syntactic functions of a word have dependency relations, they form a dependency forest. Therefore, when the syntactic function is not rashly disambiguated, the correct reading may survive even after illegitimate

[5] Coordination is handled via the coordinator that collects coordinated subjects in one slot.

linking, as the global pruning (Section 5) later extracts dependency links that form consistent trees.

Links formed between syntactic labels constitute partial trees, usually around verbal nuclei. But a new mechanism is needed to make full use of the structural information provided by multiple rules. Once a link is formed between labels, it can be used by the other rules. For example, when a head of an object phrase (`@OBJ`) is found and indexed to a verb, the noun phrase to the right (if any) is probably an object complement (`@PCOMPL-O`). It should have the same head as the existing object if the verb has the proper subcategorisation tag (`SVOC`). The following rule establishes a dependency relation of a verb and its object complement, if the object already exists.

```
INDEX (@PCOMPL-O)
    IF (*-1 @OBJ BARRIER @NPHEAD
        LINK 0 UP object SVOC HEAD=o-compl);
```

The rule says that a dependency relation (`o-compl`) should be added but the syntactic functions should not be disambiguated (`INDEX`). The object complement `@PCOMPL-O` is linked to the verb readings having the subcategorisation `SVOC`. The relation of the object complement and its head is such that the noun phrase to the left of the object complement is an object (`@OBJ`) that has established a dependency relation (`object`) to the verb.

Naturally, the dependency relations may also be followed downwards (`DOWN`). But it is also possible to declare the last item in a chain of the links (e.g. the verb chain *would have been wanted*) using the keywords `TOP` and `BOTTOM`.

5 Ambiguity and pruning

We pursue the following strategy for linking and disambiguation.

- In the best case, we are sure that some reading is correct in the current context. In this case, both disambiguation and linking can be done at the same time (with command `SELECT` and keyword `HEAD`).
- The most typical case is that the context gives some evidence about the correct reading, but we know that there are some rare instances when that reading is not correct. In such a case, we only add a link.
- Sometimes the context gives strong hints as to what the correct reading can not be. In such a case we can remove some readings even if we do not know what the correct alternative is. This is a fairly typical case in the Constraint Grammar framework, but relatively rare in the new dependency grammar. In practice, these rules are most likely to cause errors, apart from their linguistic interpretation often being rather obscure. Moreover, there is no longer any need to remove these readings explicitly by rules, because the global pruning removes readings which have not obtained any "extra evidence".

Roughly, one could say that the `REMOVE` rules of the Constraint Grammar are replaced by the `INDEX` rules. The overall result is that the rules in the new framework are much more careful than those of ENGCG.

As already noted, the dependency grammar has a big advantage over ENGCG in dealing with ambiguity. Because the dependencies are supposed to form a tree, we can heuristically prune readings that are not likely to appear in such a tree. We have the following hypotheses: (1) the dependency forest is quite sparse and a whole parse tree can not always be found; (2) pruning should favour large (sub)trees; (3) unlinked readings of a word can be removed when there is a linked reading present among the alternatives; (4) unambiguous subtrees are more likely to be correct than ambiguous ones; and (5) pruning need not force the words to be unambiguous. Instead, we can apply the rules iteratively, and usually some of the rules apply when the ambiguity is reduced. Pruning is then applied again, and so on. Furthermore, the pruning mechanism does not contain any language specific statistics, but works on a topological basis only.

Some of the most heuristic rules may be applied only after pruning. This has two advantages: very heuristic links would confuse the pruning mechanism, and words that would not otherwise have a head, may still get one.

6 Toy-grammar example

In this section, we present a set of rules, and show how those rules can parse the sentence "*Joan said whatever John likes to decide suits her*". The toy grammar containing 8 rules is presented in Figure 3. The rules are extracted from the real grammar, and they are then simplified; some tests are omitted and some tests are made simpler. The grammar is applied to the input sentence in Figure 4, where the tags are almost equivalent to those used by the English Constraint Grammar, and the final result equals Figure 2, where only the dependencies between the words and certain tags are printed.

Some comments concerning the rules in the toy grammar (Figure 3) are in order:

```
INDEX (@SUBJ) IF (1 @+F HEAD = subj:);                                              #(1)
INDEX (INF @-FMAINV) IF (-1 INFMARK) (-2 PTC1-COMPL-V + SVO HEAD = obj:);           #(2)
INDEX (@INFMARK>) IF (1 (INF @-FMAINV) HEAD = infmark:);                            #(3)
SELECT (PRON ACC @OBJ) IF (1C CLB) (-1 @MAINV HEAD = obj:);                         #(4)
INDEX (PRON WH @OBJ)                                                                #(5)
 IF (*1 @SUBJ BARRIER @NPHEAD-MAIN
      LINK 0 UP subj: @+F
      LINK 0 TOP v-ch: @MAINV
      LINK 0 BOTTOM obj: SVO + @-FMAINV HEAD = obj:);
INDEX @MAINV                                                                        #(6)
 IF (*-1 WH BARRIER @MV-CLB/CC LINK -1 @MV-CLB/CC)
    (*1C @+F BARRIER @SUBJ OR CLB HEAD = subj:);

PRUNING                                                                             #(*)
INDEX @MAINV                                                                        #(7)
 IF (NOT *1 @+F BARRIER SUBJ-BARRIER)
    (*-1 (PRON WH) BARRIER CLB LINK -1 VCOG + SVO + @MAINV HEAD = obj:);
INDEX @+FMAINV                                                                      #(8)
 IF (NOT 0 @+FAUXV) (NOT *1 @+F BARRIER CLB)
    (0 DOWN subj: @SUBJ LINK NOT *-1 @CS) (@0 (<s>) HEAD = main:);
```

Figure 3: A toy grammar of 8 rules

1. A simple rule shows how the subject (`@SUBJ`) is indexed to a finite verb by a link named *subj*.

2. The infinitives preceded by the infinitive marker *to* can be reliably linked to the verbs with the proper subcategorisation, i.e. the verb belongs to both categories `PTC1-COMPL-V` and `SVO`.

3. The infinitive marker is indexed to the infinitive by the link named `infmark`.

4. Personal pronouns have morphological ambiguity between nominative (`NOM`) and accusative (`ACC`) readings. Here, the accusative reading is selected and linked to the main verb immediately to the left, if there is an unambiguous clause boundary immediately to the right.

5. The WH-pronoun is a clause boundary marker, but the only reliable means to find its head is to follow the links. Therefore, the WH-pronoun is not indexed before the appropriate subject is linked to the verb chain which also has a verbal object.

   The rule states: the first noun phrase head label to the right is a subject (`@SUBJ`), link `subj` exists and is followed up to the finite verb (`@+F`) in a verb chain (`v-ch`), which is then followed up to the main verb. Then object or complement links are followed downwards (`BOTTOM`), to the last verbal reading (here *decide*). If then a verb with subcategorisation for objects is encountered, an object link from the WH-pronoun is formed.

   This kind of rule that starts from word A, follows links up to word B and then down to word C, introduces a non-projective dependency link if word B is between words A and C.

   Note that the conditions `TOP` and `BOTTOM` follow the chain of named link, if any, to the upper or lower end of a chain of a multiple (zero or more) links with the same name. Therefore *TOP v-ch: @MAINV* always ends with the main verb in the verb chain, whether this be a single finite verb like *likes* or a chain like *would have been liked.*

6. The WH-clause itself may function as a subject, object, etc. Therefore, there is a set of rules for each function. The "WH-clause as subject" rule looks for a finite verb to the right. No intervening subject labels and clause boundaries are allowed.

* Rules 1–5 are applied in the first round. After that, the pruning operation disambiguates finite verbs, and rule 6 will apply.

  Pruning will be applied once again. The sentence is thus disambiguated both morphologically and morphosyntactically, and a syntactic

```
"<Joan>"
    "joan" N NOM SG @NH @SUBJ @OBJ @I-OBJ @PCOMPL-S @PCOMPL-O @APP @A> @<P @O-ADVL
"<said>"
    "say" PCP2 @<P-FMAINV @-FMAINV
    "say" V PAST VFIN @+FMAINV
    "say" A ABS @PCOMPL-S @PCOMPL-O @A> @APP @SUBJ @OBJ @I-OBJ @<P @<NOM
"<whatever>"
    "whatever" ADV @ADVL @AD-A>
    "whatever" DET CENTRAL WH SG/PL @DN>
    "whatever" <CLB> PRON WH SG/PL @SUBJ @OBJ @I-OBJ @PCOMPL-S @PCOMPL-O @<P @<NOM
"<John>"
    "john" N NOM SG @NH @SUBJ @OBJ @I-OBJ @PCOMPL-S @PCOMPL-O @APP @A> @<P @O-ADVL
"<likes>"
    "like" N NOM PL @NH @SUBJ @OBJ @I-OBJ @PCOMPL-S @PCOMPL-O @APP @A> @<P @O-ADVL
    "like" V PRES SG3 VFIN @+FMAINV
"<to>"
    "to" PREP @<NOM @ADVL
    "to" INFMARK> @INFMARK>
"<decide>"
    "decide" V SUBJUNCTIVE VFIN @+FMAINV
    "decide" V IMP VFIN @+FMAINV
    "decide" V INF @-FMAINV @<P-FMAINV
    "decide" V PRES -SG3 VFIN @+FMAINV
"<suits>"
    "suit" V PRES SG3 VFIN @+FMAINV
    "suit" N NOM PL @NH @SUBJ @OBJ @I-OBJ @PCOMPL-S @PCOMPL-O @APP @A> @<P @O-ADVL
"<her>"
    "she" PRON PERS FEM GEN SG3 @GN>
    "she" PRON PERS FEM ACC SG3 @OBJ
"<.>"
```

Figure 4: A sentence after morphological analysis. Each line presents a morphological and @-signs morphosyntactic alternatives, e.g. *whatever* is ambiguous in 10 ways. The subcategorisation/valency information is not printed here.

reading from each word belongs to a subtree of which the root is *said* or *suits*.

7. The syntactic relationship between the verbs is established by a rule stating that the rightmost main verb is the (clause) object of a main verb to the left, which allows such objects.

8. Finally, there is a single main verb, which is indexed to the root (<s>) (in position @0).

7 Evaluation

7.1 Efficiency

The evaluation was done using small excerpts of data, not used in the development of the system. All text samples were excerpted from three different genres in the Bank of English (Järvinen, 1994) data: American National Public Radio (*broadcast*), British Books data (*literature*), and The Independent (*newspaper*). Figure 5 lists the samples, their sizes, and the average and maximum sentence lengths. The measure is in words excluding punctuation.

	size (w)	avg. length	max. length	total time
broadcast	2281	19	44	12 sec.
literature	1920	15	51	8.5 sec.
newspaper	1427	19	47	8.5 sec.

Figure 5: Benchmark used in the evaluation

In addition, Figure 5 shows the total processing time required for the syntactic analysis of the samples. The syntactic analysis has been done in a normal PC with the Linux operating system. The PC has a Pentium 90 MHz processor and 16 MB of memory. The speed roughly corresponds to 200 words in second. The time does not include morphological analysis and disambiguation[6].

[6]The CG-2 program (Tapanainen, 1996) runs a modified disambiguation grammar of Voutilainen (1995) about 1000 words in second.

	DG		ENGCG	
	succ.	amb.	succ.	amb.
broadcast	97.0 %	3.2 %	96.8 %	12.7 %
literature	97.3 %	3.3 %	95.9 %	11.3 %
newspaper	96.4 %	3.3 %	94.2 %	13.7 %

Figure 6: ENGCG syntax and morphosyntactic level of the dependency grammar

7.2 Comparison to ENGCG syntax

One obvious point of reference is the ENGCG syntax, which shares a level of similar representation with an almost identical tagset to the new system. In addition, both systems use the front parts of the ENGCG system for processing the input. These include the tokeniser, lexical analyser and morphological disambiguator.

Figure 6 shows the results of the comparison of the ENGCG syntax and the morphosyntactic level of the dependency grammar. Because both systems leave some amount of the ambiguity pending, two figures are given: the *success rate*, which is the percentage of correct morphosyntactic labels present in the output, and the *ambiguity rate*, which is the percentage of words containing more than one label. The ENGCG results compare to those reported elsewhere (Järvinen, 1994; Tapanainen and Järvinen, 1994).

The DG success rate is similar or maybe even slightly better than in ENGCG. More importantly, the ambiguity rate is only about a quarter of that in the ENGCG output. The overall result should be considered good in the sense that the output contains information about the syntactic functions (see Figure 4) not only part-of-speech tags.

7.3 Dependencies

The major improvement over ENGCG is the level of explicit dependency representation, which makes it possible to excerpt modifiers of certain elements, such as arguments of verbs. This section evaluates the success of the level of dependencies.

7.3.1 Unnamed dependencies

One of the crude measures to evaluate dependencies is to count how many times the correct head is found. The results are listed in Figure 7. Precision is $(\frac{\text{received correct links}}{\text{received links}})$ and recall $(\frac{\text{received correct links}}{\text{desired links}})$. The difference between precision and recall is due to the fact that the parser does not force a head on every word. Trying out some very heuristic methods to assign heads would raise recall but lower precision. A similar measure is used in (Eisner, 1996) except that every word has a head, i.e. the precision equals recall, reported as 79.2 %.

	precision	recall
broadcast	93.4 %	88.0 %
literature	96.0 %	88.6 %
newspaper	95.3 %	87.9 %

Figure 7: Percentages of heads correctly attached

broadcast	precision	recall	N
subjects	95 %	89 %	244
objects	89 %	83 %	140
predicatives	96 %	86 %	57
literature	precision	recall	N
subjects	98 %	92 %	195
objects	94 %	91 %	118
predicatives	97 %	93 %	72
newspaper	precision	recall	N
subjects	95 %	83 %	136
objects	94 %	88 %	103
predicatives	92 %	96 %	23

Figure 8: Rates for main functional dependencies

7.3.2 Named dependencies

We evaluated our parser against the selected dependencies in the test samples. The samples being rather small, only the most common dependencies are evaluated: subject, object and predicative. These dependencies are usually resolved more reliably than, say, appositions, prepositional attachments etc. The results of the test samples are listed in Figure 8. It seems the parser leaves some amount of the words unlinked (e.g. 10–15 % of subjects) but what it has recognised is generally correct (precision 95–98 % for subjects).

Dekang Lin (1996) has earlier used this kind of evaluation, where precision and recall were for subjects 87 % and 78 %, and for complements (including objects) 84 % and 72 %, respectively. The results are not strictly comparable because the syntactic description is somewhat different.

8 Conclusion

In this paper, we have presented some main features of our new framework for dependency syntax. The most important result is that the new framework allows us to describe non-projective dependency grammars and apply them efficiently. This is a property that will be crucial when we will apply this framework to a language having free word-order.

Basically, the parsing framework combines the Constraint Grammar framework (removing ambiguous readings) with a mechanism that adds dependencies between readings or tags. This means that while the parser disambiguates it also builds up a dependency forest that, in turn, is reduced by other disambiguation rules and a global pruning mechanism.

This setup makes it possible to operate on several layers of information, and use and combine structural information more efficiently than in the original Constraint Grammar framework, without any further disadvantage in dealing with ambiguity.

First preliminary evaluations are presented. Compared to the ENGCG syntactic analyser, the output not only contains more information but it is also more accurate and explicit. The ambiguity rate is reduced to a quarter without any compromise in correctness. We did not have access to other systems, and care must be taken when interpreting the results which are not strictly comparable. However, the comparison to other current systems suggests that our dependency parser is very promising both theoretically and practically.

Acknowledgments

We are using *Atro Voutilainen*'s (1995) improved part-of-speech disambiguation grammar which runs in the CG-2 parser. Voutilainen and *Juha Heikkilä* created the original ENGCG lexicon.

References

Arto Anttila. 1995. How to recognise subjects in English. In Karlsson et al., chapt. 9, pp. 315–358.

Dekang Lin. 1996. Evaluation of Principar with the Susanne corpus. In John Carroll, editor, *Workshop on Robust Parsing*, pages 54–69, Prague.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *The 16th International Conference on Computational Linguistics*, pages 340–345. Copenhagen.

David G. Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40(4):511-525.

Hans Jürgen Heringer. 1993. Dependency syntax - basic ideas and the classical model. In Joachim Jacobs, Arnim von Stechow, Wolfgang Sternefeld, and Theo Venneman, editors, *Syntax - An International Handbook of Contemporary Research*, volume 1, chapter 12, pages 298–316. Walter de Gruyter, Berlin - New York.

Richard Hudson. 1991. *English Word Grammar.* Basil Blackwell, Cambridge, MA.

Arvi Hurskainen. 1996. Disambiguation of morphological analysis in Bantu languages. In *The 16th International Conference on Computational Linguistics*, pages 568–573. Copenhagen.

Timo Järvinen. 1994. Annotating 200 million words: the Bank of English project. In *The 15th International Conference on Computational Linguistics Proceedings*, pages 565–568. Kyoto.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors. 1995. *Constraint Grammar: a language-independent system for parsing unrestricted text*, volume 4 of *Natural Language Processing.* Mouton de Gruyter, Berlin and N.Y.

Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In Hans Karlgren, editor, *Papers presented to the 13th International Conference on Computational Linguistics*, volume 3, pages 168–173, Helsinki, Finland.

Michael McCord. 1990. Slot grammar: A system for simpler construction of practical natural language grammars. In R Studer, editor, *Natural Language and Logic: International Scientific Symposium*, Lecture Notes in Computer Science, pages 118–145. Springer, Berlin.

Igor A. Mel'čuk. 1987. *Dependency Syntax: Theory and Practice.* State University of New York Press, Albany.

Christer Samuelsson, Pasi Tapanainen, and Atro Voutilainen. 1996. Inducing constraint grammars. In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Inference: Learning Syntax from Sentences*, volume 1147 of *Lecture Notes in Artificial Intelligence*, pages 146–155, Springer.

Daniel Sleator and Davy Temperley. 1991. Parsing English with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University.

Pasi Tapanainen and Timo Järvinen. 1994. Syntactic analysis of natural language using linguistic rules and corpus-based patterns. In *The 15th International Conference on Computational Linguistics Proceedings*, pages 629–634. Kyoto.

Pasi Tapanainen. 1996. *The Constraint Grammar Parser CG-2.* Number 27 in Publications of the Department of General Linguistics, University of Helsinki.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale.* Éditions Klincksieck, Paris.

Atro Voutilainen. 1995. Morphological disambiguation. In Karlsson et al., chapter 6, pages 165–284.