PROBABILISTIC METHODS IN DEPENDENCY GRAMMAR PARSING

Job M. van Zuijlen

BSO/Research
P.O.B. 8348
NL-3503 RH Utrecht
The Netherlands
e-mail: zuijlen@dlt1.uucp

June 1989

ABSTRACT

Authentic text as found in corpora cannot be described completely by a formal system, such as a set of grammar rules. As robust parsing is a prerequisite for any practical natural language processing system, there is certainly a need for techniques that go beyond merely formal approaches. Various possibilities, such as the use of simulated annealing, have been proposed recently and we have looked at their suitability for the parse process of the DLT machine translation system, which will use a large structured bilingual corpus as its main linguistic knowledge source. Our findings are that parsing is not the type of task that should be tackled solely through simulated annealing or similar stochastic optimization techniques but that a controlled application of probabilistic methods is essential for the performance of a corpus-based parser. On the basis of our explorative research we have planned a number of small-scale implementations in the near future.

1. Introduction

Usually a parser is viewed as a program that takes a sentence in a particular language as its input and delivers one or more analyses for that sentence. This is no different in the present prototype of DLT (Distributed Language Translation), a multilingual translation system under development at the Dutch software house BSO. In the prototype, we use an ATN-parser that delivers all syntactic analyses of an input sentence in the source language (SL). Each analysis undergoes structural and lexical transfer resulting in one or more target language (TL) trees.[1]

In order to limit the size of the ATN, we have used Technical English as the basis for our grammar. This type of English has been specially designed for writing technical manuals. It has certain limitations, such as the number of verb forms to be used, the number of elements that may be coordinated, sentence length and the like. Nevertheless, it proves to be very difficult to specify a complete grammar, let alone formulate grammar rules. Moreover, even with such a limited grammar we have to deal with the combinatorial explosion due to the parsing of ambiguous sentences.

[1] In fact, DLT consists of two separate but similar translation processes. The first translates the SL into the IL, DLT's Esperanto-based Intermediate Language; the second translates from the IL into the TL.

A typical complication of a translation system is that, apart from the SL grammar for the parser, we need a grammar for TL generation and a contrastive grammar (metataxis) to link source and target language. Then, there are three dictionaries, one for each language and one for the language pair. Finally, semantic information has to be included. On a prototype scale, it is already difficult to maintain consistency between the various knowledge sources, but for a large-scale industrial version this is almost impossible.

Two recent inventions by members of the DLT research team have contributed to the solution of the complications mentioned previously. Van Zuijlen (1988) has introduced the **Structured Syntactic Network** (SSN) to achieve the compact representation of all dependency-type analyses of a sentence in a single structure. The problem of consistency of knowledge sources has been tackled by Sadler (1989), who has proposed the **Bilingual Knowledge Bank** (BKB), a large structured bilingual corpus. It contains for each sentence the preferred syntactic analysis and translation in the given context, as well as certain other referential and co-referential information. An important structural element is the Translation Unit (TU), a dependency subtree for which there is a non-compositional translation, e.g. expressions like *kick the bucket.*

The introduction of the BKB places the various processes commonly found in a translation system (parsing, structural transfer, semantic evaluation, generation) in a different perspective. We will not deal here with structural transfer and generation but concentrate on the consequences for the parse process, which will be dealt with in a number of sections:

- linguistic theory and representation;
- interfacing parser and BKB;
- corpus-based parsing;
- probabilistic methods.

We conclude with a few remarks about research we have planned for the near future.

2. Linguistic Theory and Representation

The linguistic theory used in DLT is Dependency Grammar, one of the less frequently used formalisms in natural language processing projects (see Schubert (1987) for a discussion on its suitability for machine translation). The dependency grammar of a language describes syntactic relations or **dependencies** between pairs of words. The relation is directed, i.e. one word, the **governor** governs (dominates) the other, the **dependent**. In general, the dependencies range over word classes (syntactic categories) rather than specific words. A useful feature of dependency grammar is that the resulting analysis may be used directly by the semantic component of the translation system, i.e. a single type of representation suffices for all processes in the system.

The syntactic relations in dependency grammar are derived from the **function** of a word in the sentence. For example, *man* is the subject of *walks* in *The man walks*. It is important to realize that dependency grammar is primarily concerned with words; there are no phrasal categories.

A dependency tree has a geometry that is quite different from that of a constituent tree (Figure 1). Notice that in a constituent tree nodes are either phrasal or lexical, but that in a dependency tree nodes are always lexical. The branches of a dependency tree are labeled with syntactic relations. A dependency tree is not ordered, which means that a particular relation is only defined by the governor and the dependent and not by the position of the dependent with respect to other dependents. In the example word order does play a role to identify the subject and the object of the sentence but order is not reflected in the representation.

In order to facilitate the interfacing between the BKB and the parse process (see Section 3), we use an alternative representation, which we will refer to as a **Dependency Link**

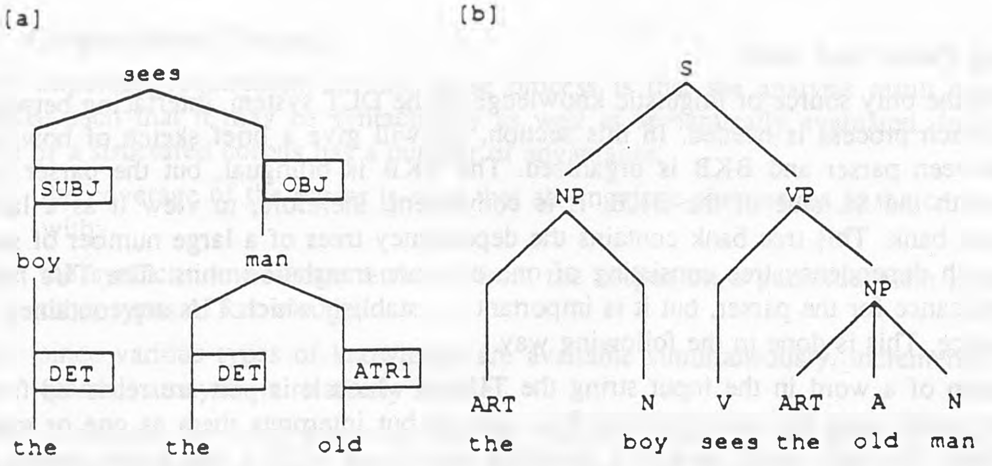

Figure 1. [a] dependency tree and [b] constituent tree for the sentence *The boy sees the old man.*

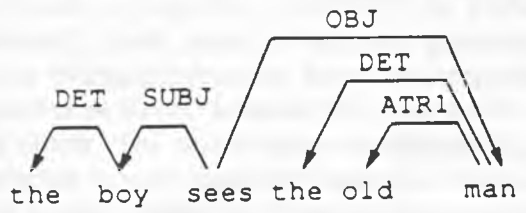

Figure 2. The dependency link representation of *The boy sees the old man.*

Representation (DLR). A dependency link consists of a governor, a dependent and their relation. The link is projective, i.e. it takes the position of governor and dependent with respect to each other into account. We obtain a graphical representation of a DLR by writing down the sentence as a linear string of words and then draw the dependencies as arcs (Dependency Links) connecting the words. Figure 2 shows the dependency link representation of *The boy sees the old man.*

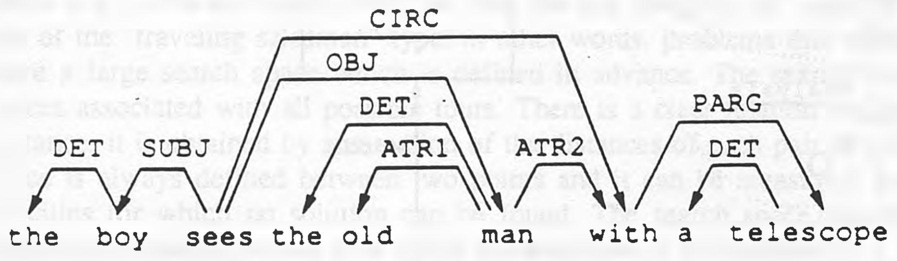

Figure 3. The dependency link representation of *The boy sees the old man with a telescope.*

The DLR shown in Figure 2 has the same representative power as a dependency tree. However, in contrast to a tree, connections in a DLR are by reference and, as a consequence, it is possible to represent directed graphs as well. Graphs are a means to represent multiple analyses of a sentence in a single representation. The ideas behind such a representation for dependency grammar, the SSN, are discussed in Van Zuijlen (1988). The dependency link may be viewed as a common building brick for trees as well as SSNs. This is shown in Figure 3 where we see the two analyses for *The boy sees the old man with a telescope* in a single DLR. By selecting either the link *man*-ATR2-*with* or *sees*-CIRC-*with* we obtain the respective interpretations. The set of dependency links that constitute one interpretation is called an ensemble.

3. Interfacing Parser and BKB

As the BKB is the only source of linguistic knowledge in the DLT system, interfacing between the BKB and each process is needed. In this section, we will give a brief sketch of how the interfacing between parser and BKB is organized. The BKB is bilingual, but the parser has only to deal with the SL side of the BKB. It is convenient, therefore, to view it as a large dependency tree bank. This tree bank contains the dependency trees of a large number of sentences, with each dependency tree consisting of one or more translation units. The TUs have no direct significance for the parser, but it is important to establish which TUs are contained in the input sentence. This is done in the following way.

After recognition of a word in the input string the TUs of which it is part are retrieved from the BKB. The parser does not deal with the TUs directly but interprets them as one or more dependency links. For each word there is a (possibly empty) set of DLs that either govern or depend on the word. By combining DLs into ensembles we obtain dependency trees the **projection** (linearization) of which has to match the input string. So parsing is not carried out by parse tree construction guided by the input string but by matching the input string with the projection of a parse tree synthesized from dependency links (Figure 4).

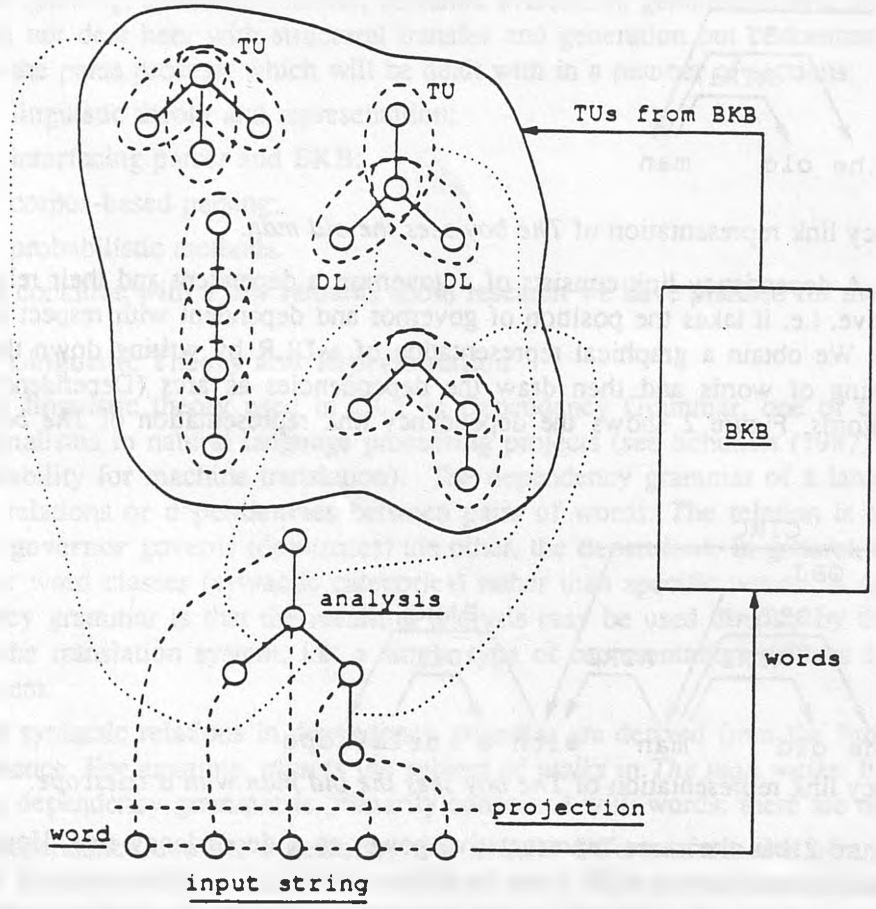

Figure 4. Parsing with a treebank. The words in the input string control the retrieval of TUs from the BKB. Each TU consists of a number of DLs which are used to synthesize an analysis tree. The projection of this tree should match the input string.

The dependency links that are "used" for the analysis (in Figure 4 connected with the analysis tree by dotted lines) select in turn those parts of the TUs retrieved from the BKB that are relevant for the translation of the input string.

4. Corpus-Based Parsing

An important requirement for the parse process is that the analysis result matches with the BKB, such that it may be syntactically as well as semantically evaluated. In that respect the use of a structured corpus has a number of advantages.

(1) the coverage of the parser is such that all linguistic phenomena in the corpus will be dealt with;

(2) the syntactic knowledge retrieved from the corpus on a particular item is consistent with other types of knowledge;

(3) since various types of knowledge are available simultaneously, incremental evaluation of (partial) analyses is relatively simple.

This is evident for input sentences that are literally present in the BKB and for which – in a manner of speaking – direct pattern matching is possible. However, we want to extend the coverage beyond that and, therefore, we have done some explorative research in the field of corpus-based parsing, primarily by reviewing work of others in the light of our specific needs.

Recent work in corpus-based parsing has a common characteristic. A parsed corpus is used as a source of linguistic knowledge and probabilistic methods are used to arrive at an analysis. Basically, parse trees are randomly generated until the optimal parse tree is found with respect to an evaluation measure based on comparison of the parse tree with the corpus. Robustness is guaranteed since, whatever the value of the evaluation, one of the analyses will be better than all others. The search space associated with the investigation of all possible parse trees for a sentence is very large and, therefore, Haigh, Sampson & Atwell (1988) apply simulated annealing in their Annealing Parser for Realistic Input Language (APRIL) as an efficient way to find this optimal parse tree for a complete sentence. Atwell, O'Donoghue & Souter (1989) have developed the Realistic Annealing Parser (RAP) which also uses simulated annealing but works incrementally, thus reducing the search space drastically. Both projects evaluate the resulting trees with corpus information, either in the form of a tree bank (Haigh et al. 1988) or first-order recursive Markov chains (Atwell et al. 1989).

Comparing APRIL and RAP shows that a slightly different approach to the same problem already results in a large reduction of the search space. This justifies the question whether simulated annealing is really a very suitable technique. If we examine the literature on that point (e.g. Aarts and Korst 1989) we find that the problems for which it is successfully applied are of the "traveling salesman" type, in other words, problems that are highly unstructured and have a large search space which is defined in advance. The search space consists of the distances associated with all possible tours. There is a clear relation between a tour and the total distance; it is obtained by summation of the distances of each pair of connected cities. The distance is always defined between two points and it can be measured; there is no configuration of cities for which no solution can be found. The search space may become very large and simulated annealing serves as a means to investigate it efficiently.

At first sight, parsing a language seems to be a similar problem. We have a number of words (cities) and, in the case of a dependency grammar representation, we have to find optimal connections between them. For each connected pair of words we compute the grammaticality of the connection (distance) by comparing it with the linguistic information we have available. Here the problem starts. The "syntactic distance" cannot be calculated straightforwardly but has to be approximated on a probabilistic basis, e.g. by counting the number of occurrences of the particular relation in a corpus. If the relation never occurs it is not possible to say anything sensible about the distance. We might assign a default value to it, but we have no certainty that it contributes to an optimal solution. This in contrast with the "traveling salesman" problem where a long distance between two points does not exclude the connection from being part of the optimal solution.

The temporary acceptance of "odd" constructions in simulated annealing parsers is motivated by the fact that during the search of a new solution the current solution is changed by means of a number of primitive modifications which may lead to intermediary results which are not well-formed. The acceptance of these results doesn't depend on their leading to a solution which may be evaluated by comparing it with the linguistic information available but on a stochastic function that states the probability with which a "bad" result is to be accepted. What is missing is the observation that language is structured and enables predictions on the basis of available partial information. So instead of a random walk (or unguided city tour) it is possible to select those transformations that are most likely to lead to an optimal solution.[2]

A corpus is very useful to make such predictions and if we intend to use the same corpus for the evaluation of the solutions we have the certainty that we only generate those solutions that are verifiable.

Again we may observe a difference with the "traveling salesman" problem. The latter has a predefined solution space and it is easy to specify primitive transformations that will lead from one solution to the other. In the case of parsing the solution space is not predefined but has to be generated on the basis of the linguistic information available. This is either a set of grammar rules or a tree bank based on a parsed corpus.

Souter (1989) discusses how difficult it is to express the grammatical information contained in a such corpus in a limited number of rules. In fact, thousands of rules are needed, many of which are only applied once or twice. He observes a close resemblance between a rule-frequency curve and the more familiar word-frequency curve (Zipf 1936). These findings support the idea that the usual grammar with a few hundred rules is not very adequate and may contain "gaps". Also, our experience with the DLT prototype has made clear to us that a rule-based approach has unacceptable limitations. Still, we are not convinced that it is necessary to apply statistical optimization all the time when a corpus is used to find the correct analysis. When dealing with input that is covered by the corpus the latter may be viewed as large set of rules and a solution will be found in a straightforward, efficient manner. Nevertheless, there is room for probabilistic methods and in the next section we will discuss some applications.

5. Probabilistic Methods

It should be clear from the discussion in the previous section that probabilism is only useful when it is applied in a controlled way. For the parse process in a BKB-based DLT system there are three application areas:

- handling input errors and unusual input;
- restricting the number of analyses;
- ordering of alternatives.

We will discuss each of these areas in the following subsections.

5.1. Incorrect and Unusual Input

As far as the parser is concerned incorrect and unusual input relate to input for which no acceptable solution can be found by straightforward matching with the BKB. The main difference is that if the input is incorrect the user should be consulted for clarification. If the input is unusual a solution should preferably be found without asking. The border between the two is

[2] In RAP (Atwell et al. 1989) the rate of convergence is improved by introducing a bias towards the transformation of low-valued parts of the tree.

determined by the fact whether it is possible to find a **single** analysis that matches with the BKB.

The ability to process deviant input is a requirement for any robust parser. In RAP and APRIL this is achieved by always generating a parse tree, even if the result is implausible. For our application this will not do. Each analysis should match with the BKB, otherwise translation is not possible. If such an analysis cannot be obtained the parser should try and find out what is wrong and, if necessary, consult the user – preferably by making some sensible suggestions.

5.1.1. Input Errors

Input errors may be of various types which ask for different approaches. However, a general principle is that we need to know what the "correct" version is in order to say something sensible about the deviations. This is a severe requirement, but if an error has only local consequences and if there is enough surrounding context it should be possible to determine the cause of the deviation.

Since error analysis may need a combined effort of different knowledge sources, the BKB approach seems to be ideal for intelligent error handling. Some types of errors we may consider are:

(1) word form errors;

(2) syntactic deviations;

(3) spelling mistakes.

Errors of type (1) or (2) are relatively easy to detect by comparing the input to the linguistic information available. An interesting method to deal with such grammatical errors has been suggested by Charniak (1983). In a rule-based parser a rule for which one or more atomic tests (e.g. agreement) fail is not applied. By modifying the tests it is possible to assign a kind of applicability measure to a rule. Instead of returning simply "yes" or "no" each test returns a value that is added to the current value of the applicability measure if the test succeeds and subtracted if the test fails.

Charniak's proposal is also very useful when a grammar is based on a corpus. For instance, it could be that, considering their word class, two words have a relation but that there is a mismatch between their features. An example is *The boy see the man*, in which subject-verb agreement is violated. However, by establishing that *the boy* could be a subject and that *see* takes one and that complete feature unification is not possible the parser classifies the error. The user will then be consulted for clarification, e.g. by being presented two correct alternatives one of which must be chosen:

(a) *The boys see the old man*

(b) *The boy sees the old man*

By using corpus information a likelihood value could be assigned to each alternative, which may be decisive if one alternative turns out to be far more plausible than any of the others, in which case user consultation is not needed.

There are errors that cannot be described on the basis of features or syntactic structures, but may be solved by using knowledge on individual words or their relations. In such cases a corpus-based system is superior. A typical example is a misspelled word, such as *foz*, which might be *fez* or *fox*. By taking the context into account and comparing it with corpus information the selection of one or the other alternative is supported. Compare:

(a) *In Morocco men wear a caftan and a foz.*

(b) *The foz hunts at night.*

The context in (a) points to the interpretation *fez*, whereas the context in (b) points to the interpretation *fox*.

5.1.2. Unusual Input

In this section we will show by means of a simple example how use of a corpus supports the handling of unusual input. We mentioned earlier that in dependency grammar dependencies range over word classes. There are cases, however, in which a word has a syntactic function that is not typical for its word class. Nouns, such as *week*, *month* and *year*, may be used as time adverbials, as in *I saw him last week.* We don't want to call *week* an adverb because it cannot perform the same functions as an adverb. On the other hand, we don't want to extend the functions that are possible for nouns because only a small number of nouns may be used in the same way as *week.*

In a rule-based parser categories are used to formulate some general distributional criteria, as it is not feasible to state for a each word the syntactic functions it may perform. Such information is, however, available in a corpus. We may find:

(1) *He came last week.*

(2) *I have had a very bad week.*

(3) *A week is enough to finish this job.*

From the available parse trees we derive the distribution of *week* in terms of governing or depending relations. Now suppose that we have the input sentence *He arrives next month*, but that we don't have direct evidence that *month* could perform the same function as *week* in (1). The parser will then compare the distribution of *month* and *week*, in order to establish if they are used in the same way, i.e. show **syntactic synonymity**. The more correspondence is found, the higher the probability that *month* may indeed be used as a time adverbial.

The method to establish the possibility for *month* to be used as time adverbial may also be applied in other cases. The syntactic context of a word may suggest a function or even word class for which there is no direct evidence. For example, in *He computers all the time* the noun *computer* is used as verb. From the corpus we may deduce that in English "any noun may be verbed" and that the use of *computer* as a verb is acceptable.

5.2. Restricting the Number of Alternatives

An exhaustive parser often generates alternatives without taking aspects of language use into account. For a system that features user interaction this results in asking the user questions about alternatives that are counter-intuitive. Consider, for example,

Daily inspections should be performed.

Here *daily* modifies *inspections* and although it could modify the verb in an alternative analysis, this interpretation is only evident when *daily* is placed at the end of the sentence:

Inspections should be performed daily.

This is an example in which a corpus could be used to limit the number of possible analyses and, thus, assist the system to behave sensibly in the eyes of the user.

The fact that the corpus sometimes extends and sometimes restricts the number of possible interpretations indicates that there is an important lexical influence in syntax which causes words to behave differently from what we expect, considering their word classes. This suggests that a strict separation between syntax and semantics (or at least language use) is not possible in the case of "realistic" language. The acceptability of certain distributions cannot be explained syntactically; there is no reason why only specific nouns may serve as adverbials. By the same token, there is no reason to exclude some potential analyses other than by observing that a language user would never interpret them that way.

5.3. Ordering Alternatives

An interactive translation system will have to deal with alternative analyses of the SL sentence, even if some of them may be excluded in advance. Particularly in the case of coordination or post-modifier sequences there may by a number of alternatives that have to be taken into account. By using the graph representation we introduced in Section 2 it is possible to represent the alternatives in a compact way. There are various techniques to prevent the combinatorial explosion caused by the generation of the alternatives (see e.g. Tomita 1985), but then we are faced with the problem of evaluating them efficiently. We intend to solve this in the following way.

We start with the incremental generation of all dependency links that are part of one or more of the potential analyses, resulting in a DLR of the input. The DLs that constitute the best analysis according to a given evaluation function are made **active**, all others are made **dormant**. If the multiple analyses are caused by structural ambiguity, such as alternative attachment points, then a simple transformation suffices to generate an alternative analysis. In Figure 3, for example, the activation of DL *man*-ATR2-*with* and the deactivation of DL *sees*-CIRC-*with* or vice versa results in an alternative analysis. So, a transformation is performed by activating/deactivating of a pair of DLs with a common dependent.

The set of DLs with a common dependent forms a **choice point**. Only DLs that are elements of choice points will have to be considered in the search for alternatives. To order the alternatives, that is to find the second best given the current optimum, it may be necessary to perform more than one transformation without knowing what the sequence of transformations is. If there is a large number of choice points, systematic evaluation of all analyses is not feasible and a stochastic optimization technique is necessary. In contrast with the parsing of arbitrary input, such a technique is applicable here since certain requirements are met (Aarts & Korst 1989: 100). The solution space (i.e. a representation of all possible solutions) is given by the DLR and there is a primitive transformation (the activation/deactivation of a pair of DLs) to generate an alternative solution. All the same, in very simple cases it is better to evaluate and compare alternatives directly. In view of this, it is advantageous to have an adaptive optimization technique that is able to select the most efficient strategy.

6. Future Work

The result of our explorative research has been that we see many interesting aspects in corpus-based parsing in connection with probabilistic methods. However, application in a BKB-based DLT system asks for an approach that is different from related proposals by others. Therefore, we have planned a number of small-scale implementations in order to find out to what extent the various ideas and suggestions put forward in this paper are indeed feasible.

Acknowledgements

I would like to thank my colleagues from BSO/Research, in particular Dorine Tamis and Ronald Vendelmans, for fruitful discussions and critical remarks.

References

Aarts, Emile and Jan Korst (1989): *Simulated Annealing and Boltzmann Machines.* John Wiley and Sons, Chichester.

Atwell, Eric, Tim O'Donoghue and Clive Souter (1989): "The COMMUNAL RAP: A Probabilistic Approach to NL Parsing". Leeds University, Leeds.

Charniak, Eugene (1983): "A Parser with Something for Everyone". In: King, Margaret (ed.), *Parsing Natural Language*. Academic Press, London.

Haigh, Robin, Geoffrey Sampson and Eric Atwell (1988): "Project APRIL - a Progress Report". *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*. Buffalo.

Sadler, Victor (1989): *The Bilingual Knowledge Bank*. BSO/Research, Utrecht.

Schubert, Klaus (1987): *Metataxis*. (= Distributed Language Translation 2). Foris, Dordrecht.

Souter, Clive (1989): "The COMMUNAL Project: Extracting a Grammar from the Polytechnic of Wales Corpus". *ICAME Journal*, No. 13, April 1988.

Tomita, Masaru (1985): *Efficient Parsing for Natural Language*. Kluwer, Boston.

Van Zuijlen, Job M. (1988): "A Technique for the Compact Representation of Multiple Analyses in Dependency Grammar". BSO/Research, Utrecht.

Zipf, George (1936): *The Psycho-Biology of Language: An Introduction to Dynamic Philology*. George Routledge, London.