A DOP Model for Semantic Interpretation*

Remko Bonnema, Rens Bod and **Remko Scha**
Institute for Logic, Language and Computation
University of Amsterdam
Spuistraat 134, 1012 VB Amsterdam
Bonnema@mars.let.uva.nl
Rens.Bod@let.uva.nl
Remko.Scha@let.uva.nl

Abstract

In data-oriented language processing, an annotated language corpus is used as a stochastic grammar. The most probable analysis of a new sentence is constructed by combining fragments from the corpus in the most probable way. This approach has been successfully used for syntactic analysis, using corpora with syntactic annotations such as the Penn Tree-bank. If a corpus with semantically annotated sentences is used, the same approach can also generate the most probable semantic interpretation of an input sentence. The present paper explains this semantic interpretation method. A data-oriented semantic interpretation algorithm was tested on two semantically annotated corpora: the English ATIS corpus and the Dutch OVIS corpus. Experiments show an increase in semantic accuracy if larger corpus-fragments are taken into consideration.

1 Introduction

Data-oriented models of language processing embody the assumption that human language perception and production works with representations of concrete past language experiences, rather than with abstract grammar rules. Such models therefore maintain large corpora of linguistic representations of previously occurring utterances. When processing a new input utterance, analyses of this utterance are constructed by combining fragments from the corpus; the occurrence-frequencies of the fragments are used to estimate which analysis is the most probable one.

* This work was partially supported by NWO, the Netherlands Organization for Scientific Research (Priority Programme Language and Speech Technology).

For the syntactic dimension of language, various instantiations of this data-oriented processing or "DOP" approach have been worked out (e.g. Bod (1992-1995); Charniak (1996); Tugwell (1995); Sima'an et al. (1994); Sima'an (1994; 1996a); Goodman (1996); Rajman (1995ab); Kaplan (1996); Sekine and Grishman (1995)). A method for extending it to the semantic domain was first introduced by van den Berg et al. (1994). In the present paper we discuss a computationally effective version of that method, and an implemented system that uses it. We first summarize the first fully instantiated DOP model as presented in Bod (1992-1993). Then we show how this method can be straightforwardly extended into a semantic analysis method, if corpora are created in which the trees are enriched with semantic annotations. Finally, we discuss an implementation and report on experiments with two semantically analyzed corpora (ATIS and OVIS).

2 Data-Oriented Syntactic Analysis

So far, the data-oriented processing method has mainly been applied to corpora with simple syntactic annotations, consisting of labelled trees. Let us illustrate this with a very simple imaginary example. Suppose that a corpus consists of only two trees:

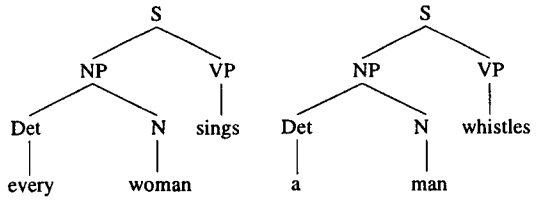

Figure 1: Imaginary corpus of two trees

We employ one operation for combining subtrees, called *composition*, indicated as $\circ$; this operation identifies the leftmost nonterminal leaf node of one tree with the root node of a second tree (i.e., the second tree is *substituted* on the leftmost nontermi-

nal leaf node of the first tree). A new input sentence like "*A woman whistles*" can now be parsed by combining subtrees from this corpus. For instance:

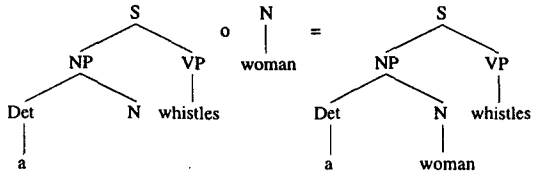

Figure 2: Derivation and parse for "*A woman whistles*"

Other derivations may yield the same parse tree; for instance[1]:

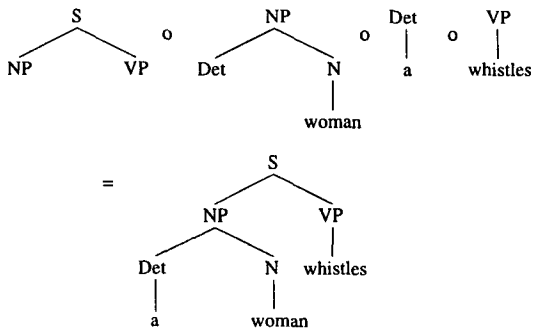

Figure 3: Different derivation generating the same parse for "*A woman whistles*"

or

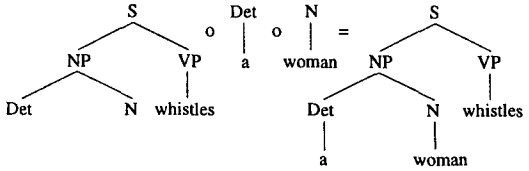

Figure 4: Another derivation generating the same parse for "*A woman whistles*"

Thus, a parse tree can have many derivations involving different corpus-subtrees. DOP estimates the probability of substituting a subtree t on a specific node as the probability of selecting t among all subtrees in the corpus that could be substituted on that node. This probability is equal to the number of occurrences of a subtree t, divided by the total number of occurrences of subtrees t' with the same root node label as t : $P(t) = |t| / \sum_{t':root(t')=root(t)} |t'|$. The probability of a derivation $t_1 \circ \cdots \circ t_n$ can be computed as the product of the probabilities of the subtrees this derivation consists of: $P(t_1 \circ \cdots \circ t_n) = \prod_i P(t_i)$. The probability of a parse tree is equal to the probability that any of its distinct derivations is generated, which is the sum of the probabilities of all derivations of that parse tree. Let t_{id} be the i-th subtree in the derivation d that yields tree T, then the probability of T is given by: $P(T) = \sum_d \prod_i P(t_{id})$.

The DOP method differs from other statistical approaches, such as Pereira and Schabes (1992), Black et al. (1993) and Briscoe (1994), in that it does not predefine or train a formal grammar; instead it takes subtrees directly from annotated sentences in a treebank with a probability proportional to the number of occurrences of these subtrees in the treebank. Bod (1993b) shows that DOP can be implemented using context-free parsing techniques. To select the most probable parse, Bod (1993a) gives a Monte Carlo approximation algorithm. Sima'an (1995) gives an efficient polynomial algorithm for a sub-optimal solution.

The model was tested on the Air Travel Information System (ATIS) corpus as analyzed in the Penn Treebank (Marcus et al. (1993)), achieving better test results than other stochastic grammars (cf. Bod (1996), Sima'an (1996a), Goodman (1996)). On Penn's Wall Street Journal corpus, the data-oriented processing approach has been tested by Sekine and Grishman (1995) and by Charniak (1996). Though Charniak only uses corpus-subtrees smaller than depth 2 (which in our experience constitutes a less-than-optimal version of the data-oriented processing method), he reports that it "outperforms all other non-word-based statistical parsers/grammars on this corpus". For an overview of data-oriented language processing, we refer to (Bod and Scha, 1996).

3 Data-Oriented Semantic Analysis

To use the DOP method not just for syntactic analysis, but also for semantic interpretation, four steps must be taken:

1. decide on a formalism for representing the meanings of sentences and surface-constituents.
2. annotate the corpus-sentences and their surface-constituents with such semantic representations.
3. establish a method for deriving the meaning representations associated with arbitrary corpus-subtrees and with compositions of such subtrees.
4. reconsider the probability calculations.

We now discuss these four steps.

3.1 Semantic formalism

The decision about the representational formalism is to some extent arbitrary, as long as it has a well-

[1]Here $t \circ u \circ v \circ w$ should be read as $((t \circ u) \circ v) \circ w$.

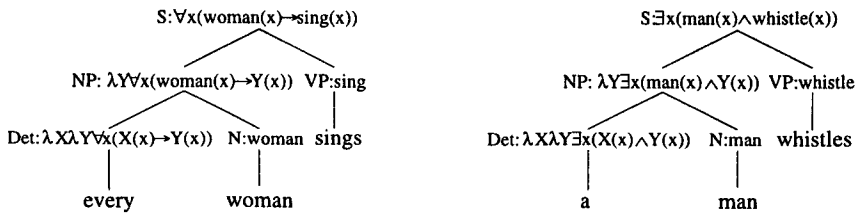

Figure 5: Imaginary corpus of two trees with syntactic and semantic labels.

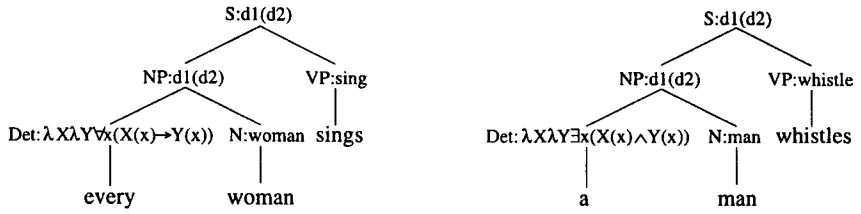

Figure 6: Same imaginary corpus of two trees with syntactic and semantic labels using the daughter notation.

defined model-theory and is rich enough for representing the meanings of sentences and constituents that are relevant for the intended application domain. For our exposition in this paper we will use a wellknown standard formalism: extensional type theory (see Gamut (1991)), i.e., a higher-order logical language that combines lambda-abstraction with connectives and quantifiers. The first implemented system for data-oriented semantic interpretation, presented in Bonnema (1996), used a different logical language, however. And in many application contexts it probably makes sense to use an A.I.-style language which highlights domain structure (frames, slots, and fillers), while limiting the use of quantification and negation (see section 5).

3.2 Semantic annotation

We assume a corpus that is already syntactically annotated as before: with labelled trees that indicate surface constituent structure. Now the basic idea, taken from van den Berg et al. (1994), is to augment this syntactic annotation with a semantic one: to every meaningful syntactic node, we add a type-logical formula that expresses the meaning of the corresponding surface-constituent. If we would carry out this idea in a completely direct way, the toy corpus of Figure 1 might, for instance, turn into the toy corpus of Figure 5.

Van den Berg et al. indicate how a corpus of this sort may be used for data-oriented semantic interpretation. Their algorithm, however, requires a procedure which can inspect the semantic formula of a node and determine the contribution of the semantics of a lower node, in order to be able to "factor out" that contribution. The details of this procedure have not been specified. However, van den Berg et al. also propose a simpler annotation convention which avoids the need for this procedure, and which is computationally more effective: an annotation convention which indicates explicitly how the semantic formula for a node is built up on the basis of the semantic formulas of its daughter nodes.

Using this convention, the semantic annotation of the corpus trees is indicated as follows:

- For every meaningful lexical node a type logical formula is specified that represents its meaning.
- For every meaningful non-lexical node a formula schema is specified which indicates how its meaning representation may be put together out of the formulas assigned to its daughter nodes.

In the examples below, these schemata use the variable *d1* to indicate the meaning of the leftmost daughter constituent, *d2* to indicate the meaning of the second daughter constituent, etc. Using this notation, the semantically annotated version of the toy corpus of Figure 1 is the toy corpus rendered in Figure 6. This kind of semantic annotation is what will be used in the construction of the corpora described in section 5 of this paper. It may be noted that the rather oblique description of the semantics of the higher nodes in the tree would easily lead to mistakes, if annotation would be carried out completely manually. An annotation tool that makes the expanded versions of the formulas visible for the annotator is obviously called for. Such a tool was developed by Bonnema (1996), it will be briefly described in section 5.

This annotation convention obviously assumes that the meaning representation of a surface-constituent *can* in fact always be composed out of the meaning representations of its subconstituents. This assumption is not unproblematic. To maintain it in the face of phenomena such as non-standard quantifier scope or discontinuous constituents creates complications in the syntactic or semantic analyses assigned to certain sentences and their constituents. It is therefore not clear yet whether our current treatment ought to be viewed as completely general, or whether a treatment in the vein of van den Berg et al. (1994) should be worked out.

3.3 The meanings of subtrees and their compositions

As in the purely syntactic version of DOP, we now want to compute the probability of a (semantic) analysis by considering the most probable way in which it can be generated by combining subtrees from the corpus. We can do this in virtually the same way. The only novelty is a slight modification in the process by which a corpus tree is decomposed into subtrees, and a corresponding modification in the composition operation which combines subtrees. If we extract a subtree out of a tree, we replace the semantics of the new leaf node with a unification variable of the same type. Correspondingly, when the composition operation substitutes a subtree at this node, this unification variable is unified with the semantic formula on the substituting tree. (It is required that the semantic type of this formula matches the semantic type of the unification variable.)

A simple example will make this clear. First, let us consider what subtrees the corpus makes available now. As an example, Figure 7 shows one of the decompositions of the annotated corpus sentence "*A man whistles*". We see that by decomposing the tree into two subtrees, the semantics at the breakpoint-node *N: man* is replaced by a variable. Now an analysis for the sentence "A woman whistles" can, for instance, be generated in the way shown in Figure 8.

3.4 The Statistical Model of Data-Oriented Semantic Interpretation

We now define the probability of an interpretation of an input string.

Given a partially annotated corpus as defined above, the multiset of corpus subtrees consists of all subtrees with a well-defined top-node semantics, that are generated by applying to the trees of the corpus the decomposition mechanism described

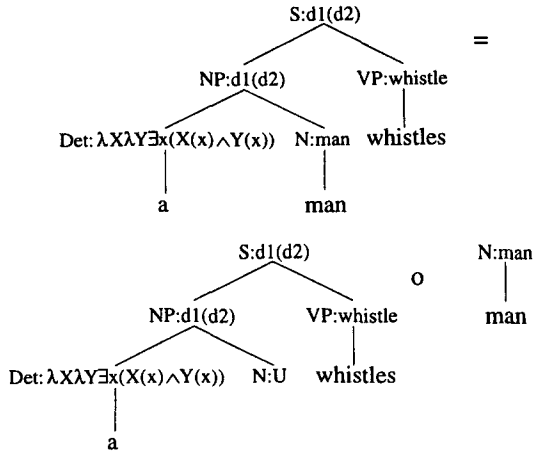

Figure 7: Decomposing a tree into subtrees with unification variables.

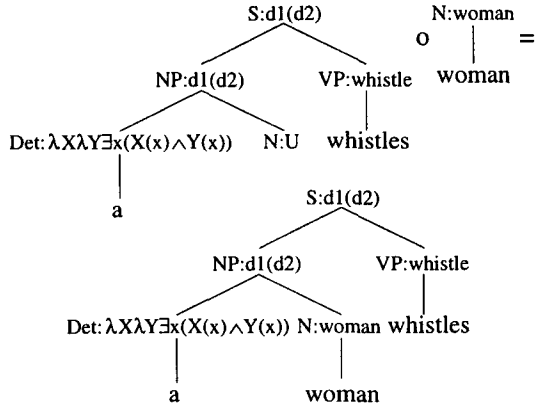

Figure 8: Generating an analysis for "*A woman whistles*".

above. The probability of substituting a subtree t on a specific node is the probability of selecting t among all subtrees in the multiset that could be substituted on that node. This probability is equal to the number of occurrences of a subtree t, divided by the total number of occurrences of subtrees t' with the same root node label as t:

$$P(t) = \frac{|t|}{\sum_{t':root(t')=root(t)} |t'|} \qquad (1)$$

A derivation of a string is a tuple of subtrees, such that their composition results in a tree whose yield is the string. The probability of a derivation $t_1 \circ \cdots \circ t_n$ is the product of the probabilities of these subtrees:

$$P(t_1 \circ \cdots \circ t_n) = \prod_i P(t_i) \qquad (2)$$

A tree resulting from a derivation of a string is called a parse of this string. The probability of a parse is

the probability that any of its derivations occurs; this is the sum of the probabilities of all its derivations. Let t_{id} be the i-th subtree in the derivation d that yields tree T, then the probability of T is given by:

$$P(T) = \sum_{d} \prod_{i} P(t_{id}) \quad (3)$$

An interpretation of a string is a formula which is provably equivalent to the semantic annotation of the top node of a parse of this string. The probability of an interpretation I of a string is the sum of the probabilities of the parses of this string with a top node annotated with a formula that is provably equivalent to I. Let t_{idp} be the i-th subtree in the derivation d that yields parse p with interpretation I, then the probability of I is given by:

$$P(I) = \sum_{p} \sum_{d} \prod_{i} P(t_{idp}) \quad (4)$$

We choose the most probable interpretation I of a string s as the most appropriate interpretation of s.

In Bonnema (1996) a semantic extension of the DOP parser of Sima'an (1996a) is given. But instead of computing the most likely interpretation of a string, it computes the interpretation of the most likely combination of semantically annotated subtrees. As was shown in Sima'an (1996b), the most likely interpretation of a string cannot be computed in *deterministic* polynomial time. It is not yet known how often the most likely interpretation and the interpretation of the most likely combination of semantically enriched subtrees do actually coincide.

4 Implementations

The first implementation of a semantic DOP-model yielded rather encouraging preliminary results on a semantically enriched part of the ATIS-corpus. Implementation details and experimental results can be found in Bonnema (1996), and Bod et al. (1996). We repeat the most important observations:

- Data-oriented semantic interpretation seems to be robust; of the sentences that could be parsed, a significantly higher percentage received a correct semantic interpretation (88%), than an exactly correct syntactic analysis (62%).
- The coverage of the parser was rather low (72%), because of the sheer number of different semantic types and constructs in the trees.
- The parser was fast: on the average six times as fast as a parser trained on syntax alone.

The current implementation is again an extension of Sima'an (1996a), by Bonnema[2]. In our experiments, we notice a robustness and speed-up comparable to our experience with the previous implementation. Besides that, we observe higher accuracy, *and* higher coverage, due to a new method of organizing the information in the tree-bank before it is used for building the actual parser.

A semantically enriched tree-bank will generally contain a wealth of detail. This makes it hard for a probabilistic model to estimate all parameters. In sections 4.1 and 4.2, we discuss a way of generalizing over semantic information in the tree-bank, *before* a DOP-parser is trained on the material. We automatically learn a simpler, less redundant representation of the same information. The method is employed in our current implementation.

4.1 Simplifying the tree-bank

A tree-bank annotated in the manner described above, consists of tree-structures with syntactic and semantic attributes at every node. The semantic attributes are rules that indicate how the meaning-representation of the expression dominated by that node is built-up out of its parts. Every instance of a semantic rule at a node has a semantic type associated with it. These types usually depend on the lexical instantiations of a syntactic-semantic structure.

If we decide to view subtrees as identical iff their syntactic structure, the semantic rule at each node, *and* the semantic type of each node is identical, any fine-grained type-system will cause a huge increase in different instantiations of subtrees. In the two tree-banks we tested on, there are many subtrees that differ in semantic type, but otherwise share the same syntactic/semantic structure. Disregarding the semantic types completely, on the other hand, will cause syntactic constraints to govern both syntactic substitution and semantic unification. The semantic types of constituents often give rise to differences in semantic structure. If this type information is not available during parsing, important clues will be missing, and loss of accuracy will result.

Apparently, we do need *some* of the information present in the types of semantic expressions. Ignoring semantic types will result in loss of accuracy, but distinguishing all different semantic types will result in loss of coverage and generalizing power. With these observations in mind, we decided to group the types, and relax the constraints on semantic unification. In this approach, every semantic expression,

[2]With thanks to Khalil Sima'an for fruitful discussions, and for the use of his parser

and every variable, has a set of types associated with it. In our semantic DOP model, we modify the constraints on semantic unification as follows: A variable can be unified with an expression, if the intersection of their respective sets of types is not empty.

The semantic types are classified into sets that can be distinguished on the basis of their behavior in the tree-bank. We let the tree-bank data decide which types can be grouped together, and which types should be distinguished. This way we can generalize over semantic types, *and* exploit relevant type-information in the parsing process at the same time. In learning the optimal grouping of types, we have two concerns: keeping the number of different sets of types to a minimum, and increasing the semantic determinacy of syntactic structures enhanced with type-information. We say that a subtree T, with type-information at every node, is semantically determinate, iff we can determine a unique, correct semantic rule for every CFG rule R[3] occurring in T. Semantic determinacy is very attractive from a computational point of view: if our processed tree-bank has semantic determinacy, we do not need to involve the semantic rules in the parsing process. Instead, the parser yields parses containing information regarding syntax and semantic types, and the actual semantic rules can be determined on the basis of that information. In the next section we will elaborate on how we learn the grouping of semantic types from the data.

4.2 Classification of semantic types

The algorithm presented in this section proceeds by grouping semantic types occurring with the same syntactic label into mutually exclusive sets, and assigning to every syntactic label an index that indicates to which set of types its corresponding semantic type belongs. It is an iterative, greedy algorithm. In every iteration a tuple, consisting of a syntactic category and a set of types, is selected. Distinguishing this tuple in the tree bank, leads to the greatest increase in semantic determinacy that could be found. Iteration continues until the increase in semantic determinacy is below a certain threshold.

Before giving the algorithm, we need some definitions:

[3]By "CFG rule", we mean a subtree of depth 1, without a specified root-node semantics, but *with* the features relevant for substitution, i.e. syntactic category and semantic type. Since the subtree of depth 1 is the smallest structural building block of our DOP model, semantic determinacy of every CFG rule in a subtree, means the whole subtree is semantically determinate.

tuples()
: *tuples*($\mathbb{T}$) is the set of all pairs $\langle c, s\rangle$ in a tree-bank $\mathbb{T}$, where c is a syntactic category, and s is the set of all semantic types that a constituent of category c in $\mathbb{T}$ can have.

apply()
: **if** c is a category, s is a set of types, and $\mathbb{T}$ is a tree-bank
then *apply*$(\langle c, s\rangle, \mathbb{T})$ yields a tree-bank $\mathbb{T}'$, by indexing each instance of category c in $\mathbb{T}$, such that the c constituent is of semantic type $t \in s$, with a unique index i.

amb()
: **if** $\mathbb{T}$ is a tree-bank
then *amb*($\mathbb{T}$) yields an $n \in \mathbb{N}$, such that n is the sum of the frequencies of all CFG rules R that occur in $\mathbb{T}$ with more than one corresponding semantic rule.

The algorithm starts with a tree-bank $\mathbb{T}_0$; in $\mathbb{T}_0$, the cardinality of *tuples*($\mathbb{T}_0$) equals the number of different syntactic categories in $\mathbb{T}_0$.

$$
\begin{array}{ll}
1. & \mathbb{T}_{i=0} \\
\text{repeat} & \\
2. & \mathbf{D}(\langle c, s\rangle) = amb(\mathbb{T}_i) - amb(apply(\langle c, s\rangle, \mathbb{T}_i)) \\
 & \mathcal{T}_i = \{\langle c, s'\rangle | \exists \langle c, s\rangle \in tuples(\mathbb{T}_i) \,\&\, s' \in 2^{|s|}\} \\
 & \tau_i = \underset{\tau' \in \mathcal{T}_i}{\text{argmax}}\, \mathbf{D}(\tau') \\
3. & i := i + 1 \\
4. & \mathbb{T}_i := apply(\tau_i, \mathbb{T}_{i-1}) \\
\text{until} & \mathbf{D}(\tau_{i-1}) \leqq \delta \\
5. & \mathbb{T}_{i-1}
\end{array}
\tag{5}
$$

$2^{|s|}$ is the powerset of s. In the implementation, a limit can be set to the cardinality of $s' \in 2^{|s|}$, to avoid excessively long processing time. Obviously, the iteration will always end, if we require δ to be ≥ 0. When the algorithm finishes, $\tau_0, \ldots, \tau_{i-1}$ contain the category/set-of-types pairs that took the largest steps towards semantic determinacy, and are therefore distinguished in the tree-bank. The semantic types not occurring in any of these pairs are grouped together, and treated as equivalent.

Note that the algorithm cannot be guaranteed to achieve full semantic determinacy. The degree of semantic determinacy reached, depends on the consistency of annotation, annotation errors, the granularity of the type system, peculiarities of the language, in short: on the nature of the tree-bank. To force semantic determinacy, we assign a unique index to those rare instances of categories, i.e, left hand sides

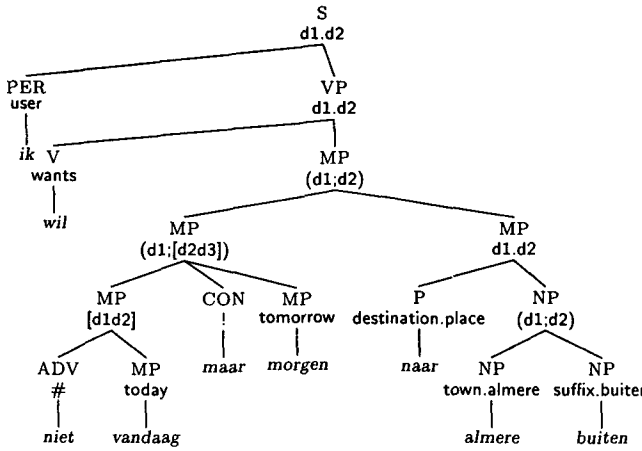

Figure 9: A tree from the OVIS tree-bank

of CFG-rules, that do not have any distinguishing features to account for their differing semantic rule. Now the resulting tree-bank embodies a function from CFG rules to semantic rules. We store this function in a table, and strip all semantic rules from the trees. As the experimental results in the next section show, using a tree-bank obtained in this way for data oriented semantic interpretation, results in high coverage, and good probability estimations.

5 Experiments on the OVIS tree-bank

The NWO[4] Priority Programme "Language and Speech Technology" is a five year research programme aiming at the development of advanced telephone-based information systems. Within this programme, the OVIS[5] tree-bank is created. Using a pilot version of the OVIS system, a large number of human-machine dialogs were collected and transcribed. Currently, 10.000 user utterances have received a full syntactic and semantic analysis. Regrettably, the tree-bank is not available (yet) to the public. More information on the tree-bank can be found on `http://grid.let.rug.nl:4321/`. The semantic domain of all dialogs, is the Dutch railways schedule. The user utterances are mostly answers to questions, like: "From where to where do you want to travel?", "At what time do you want to arrive in Amsterdam?", "Could you please repeat your destination?". The annotation method is robust and flexible, as we are dealing with real, spoken data, containing a lot of clearly ungrammatical utterances. For the annotation task, the annotation workbench SEMTAGS is used. It is a graphical interface, written by Bonnema, offering all functionality needed for examining, evaluating, and editing syntactic and semantic analyses. SEMTAGS is mainly used for correcting the output of the DOP-parser. It incrementally builds a probabilistic model of corrected annotations, allowing it to quickly suggest alternative semantic analyses to the annotator. It took approximately 600 hours to annotate these 10.000 utterances (supervision included).

Syntactic annotation of the tree-bank is conventional. There are 40 different syntactic categories in the OVIS tree-bank, that appear to cover the syntactic domain quite well. No grammar is used to determine the correct annotation; there is a small set of guidelines, that has the degree of detail necessary to avoid an "anything goes"-attitude in the annotator, but leaves room for his/her perception of the structure of an utterance. There is no conceptual division in the tree-bank between POS-tags and nonterminal categories.

Figure 9 shows an example tree from the tree-bank. It is an analysis of the Dutch sentence: "Ik(I) wil($want$) niet(not) vandaag($today$) maar(but) morgen($tomorrow$) naar(to) Almere Buiten($Almere\ Buiten$)". The analysis uses the formula schemata discussed in section 3.2, but here the interpretations of daughter-nodes are so-called "update" expressions, conforming to a frame structure, that are combined into an update of an information state. The complete interpretation of this utterance is: user.wants.(([#today];[!tomorrow]);destination.-place.(town.almere;suffix.buiten)). The semantic formalism employed in the tree-bank is the topic of the next section.

5.1 The Semantic formalism

The semantic formalism used in the OVIS tree-bank, is a frame semantics, defined in Veldhuijzen van Zanten (1996). In this section, we give a very short impression. The well-formedness and validity of an expression is decided on the basis of a type-lattice, called a frame structure. The interpretation of an utterance, is an update of an information state. An information state is a representation of objects and the relations between them, that complies to the frame structure. For OVIS, the various objects are related to concepts in the train travel domain. In updating an information state, the notion of a slot-value assignment is used. Every object can be a slot or a value. The slot-value assignments are defined in a way that corresponds closely to the linguistic notion of a ground-focus structure. The slot is part of the common ground, the value

[4]Netherlands Organization for Scientific Research
[5]Public Transport Information System

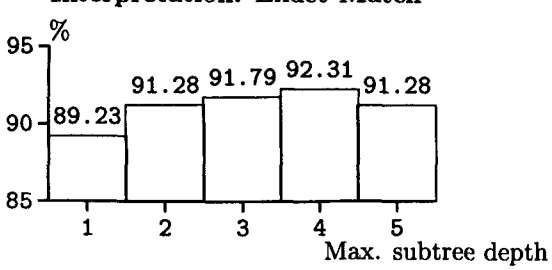

Figure 10: Size of training set: 8500

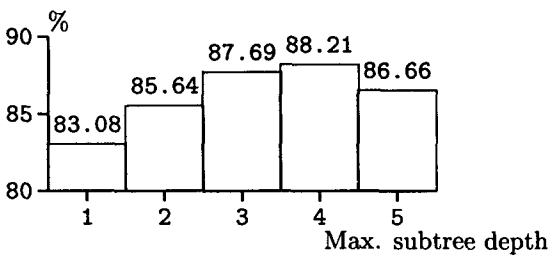

Figure 11: Size of training set: 8500

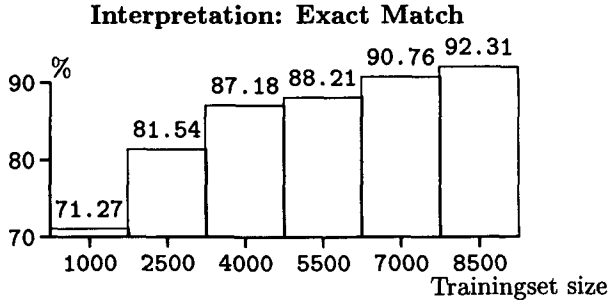

Figure 12: Max. depth of subtrees = 4

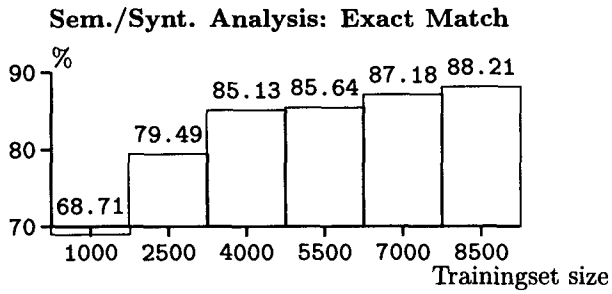

Figure 13: Max. depth of subtrees = 4

is new information. Added to the semantic formalism are pragmatic operators, corresponding to denial, confirmation , correction and assertion[6] that indicate the relation between the value in its scope, and the information state.

An update expression is a set of paths through the frame structure, enhanced with pragmatic operators that have scope over a certain part of a path. For the semantic DOP model, the semantic type of an expression ϕ is a pair of types $\langle t_1, t_2 \rangle$. Given the type-lattice $\mathcal{T}$ of the frame structure, t_1 is the lowest upper bound in $\mathcal{T}$ of the paths in ϕ, and t_2 is the greatest lower bound in $\mathcal{T}$ of the paths in ϕ.

5.2 Experimental results

We performed a number of experiments, using a random division of the tree-bank data into test- and training-set. No provisions were taken for unknown words. The results reported here, are obtained by randomly selecting 300 trees from the tree-bank. All utterances of length greater than one in this selection are used as testing material. We varied the size of the training-set, and the maximal depth of the subtrees. The average length of the test-sentences was 4.74 words. There was a constraint on the extraction of subtrees from the training-set trees: subtrees could have a maximum of two substitution-sites, and no more than three contiguous lexical nodes (Experience has shown that such limitations improve probability estimations, while retaining the full power of DOP). Figures 10 and 11 show results using a training set size of 8500 trees. The maximal depth of subtrees involved in the parsing process was varied from 1 to 5. Results in figure 11 concern a match with the total analysis in the test-set, whereas Figure 10 shows success on just the resulting interpretation. Only *exact* matches with the trees and interpretations in the test-set were counted as successes. The experiments show that involving larger fragments in the parsing process leads to higher accuracy. Apparently, for this domain fragments of depth 5 are too large, and deteriorate probability estimations[7]. The results also confirm our earlier findings, that semantic parsing is robust. Quite a few analysis trees that did not exactly match with their counterparts in the test-set, yielded a semantic interpretation that did match. Finally, figures 12 and 13 show results for differing training-set sizes, using subtrees of maximal depth 4.

[6] In the example in figure 9, the pragmatic operators #, denial, and !, correction, are used

References

M. van den Berg, R. Bod, and R. Scha. 1994. A Corpus-Based Approach to Semantic Interpretation. In *Proceedings Ninth Amsterdam Colloquium*. ILLC,University of Amsterdam.

[7] Experiments using fragments of maximal depth 6 and maximal depth 7 yielded the same results as maximal depth 5

E. Black, R. Garside, and G. Leech. 1993. *Statistically-Driven Computer Grammars of English: The IBM/Lancaster Approach.* Rodopi, Amsterdam-Atlanta.

R. Bod. 1992. A computational model of language performance: Data Oriented Parsing. In *Proceedings COLING'92*, Nantes.

R. Bod. 1993a. Monte Carlo Parsing. In *Proceedings Third International Workshop on Parsing Technologies, Tilburg/Durbuy.*

R. Bod. 1993b. Using an Annotated Corpus as a Stochastic Grammar. In *Proceedings EACL'93*, Utrecht.

R. Bod. 1995. *Enriching Linguistics with Statistics: Performance models of Natural Language.* Phd-thesis, ILLC-dissertation series 1995-14, University of Amsterdam. `ftp://ftp.fwi.uva.nl/pub/theory/illc/-dissertations/DS-95-14.text.ps.gz`

R Bod. 1996. Two Questions about Data-Oriented Parsing. In *Proceedings Fourth Workshop on Very Large Corpora*, Copenhagen, Denmark. (cmp-lg/9606022).

R. Bod, R. Bonnema, and R. Scha. 1996. A data-oriented approach to semantic interpretation. In *Proceedings Workshop on Corpus-Oriented Semantic Analysis, ECAI-96*, Budapest, Hungary. (cmp-lg/9606024).

R. Bod and R. Scha. 1996. Data-oriented language processing. an overview. Technical Report LP-96-13, Institute for Logic, Language and Computation, University of Amsterdam. (cmp-lg/9611003).

R. Bonnema. 1996. Data oriented semantics. Master's thesis, Department of Computational Linguistics, University of Amsterdam. `http://mars.let.uva.nl/remko_b/-dopsem/scriptie.html`

T. Briscoe. 1994. Prospects for practical parsing of unrestricted text: Robust statistical parsing techniques. In N. Oostdijk and P de Haan, editors, *Corpus-based Research into Language.* Rodopi, Amsterdam.

E. Charniak. 1996. Tree-bank grammars. In *Proceedings AAAI'96*, Portland, Oregon.

L. Gamut. 1991. *Logic, Language and Meaning.* Chicago University Press.

J Goodman. 1996. Efficient Algorithms for Parsing the DOP Model. In *Proceedings Empirical Methods in Natural Language Processing*, Philadelphia, Pennsylvania.

R. Kaplan. 1996. A probabilistic approach to Lexical-Functional Grammar. Keynote paper held at the LFG-workshop 1996, Grenoble, France. `ftp://ftp.parc.xerox.com/pub/-nl/slides/grenoble96/kaplan-doptalk.ps`.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the ACL*, Newark, De.

M. Rajman. 1995a. *Apports d'une approche a base de corpus aux techniques de traitement automatique du langage naturel.* Ph.D. thesis, Ecole Nationale Superieure des Telecommunications, Paris.

M. Rajman. 1995b. Approche probabiliste de l'analyse syntaxique. *Traitement Automatique des Langues*, 36:1–2.

S. Sekine and R. Grishman. 1995. A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings Fourth International Workshop on Parsing Technologies*, Prague, Czech Republic.

K. Sima'an, R. Bod, S. Krauwer, and R. Scha. 1994. Efficient Disambiguation by means of Stochastic Tree Substitution Grammars. In *Proceedings International Conference on New Methods in Language Processing.* CCL, UMIST, Manchester.

K. Sima'an. 1995. An optimized algorithm for Data Oriented Parsing. In *Proceedings International Conference on Recent Advances in Natural Language Processing.* Tzigov Chark, Bulgaria.

K. Sima'an. 1996a. An optimized algorithm for Data Oriented Parsing. In R. Mitkov and N. Nicolov, editors, *Recent Advances in Natural Language Processing 1995*, volume 136 of *Current Issues in Linguistic Theory.* John Benjamins, Amsterdam.

K. Sima'an. 1996b. Computational Complexity of Probabilistic Disambiguation by means of Tree-Grammars. In *Proceedings COLING'96*, Copenhagen, Denmark.

D. Tugwell. 1995. A state-transition grammar for data-oriented parsing. In *Proceedings European Chapter of the ACL'95*, Dublin, Ireland.

G. Veldhuijzen van Zanten. 1996. Semantics of update expressions. NWO priority Programme Language and Speech Technology, `http://grid.let.rug.nl:4321/`.