

A Corpus-Based Statistical Semantic Parser

Michel Génèreux

Seminar aus Informatik

Lehrveranstaltungsleiter: Ass.-Prof.Dr. Ernst Buchberger

Inst.f.Med.Kybernetik und AI d. Univ. Wien

Sommersemester 2001

Abstract

Since the advent of the Internet, there is a widespread access to a large quantity of information available online in various formats. This information could be accessed via a natural language interface (NLI) that would make it available to users that do not know or understand how that information is actually stored. However, NLIs are difficult to build and must be tailored to each domain of application. This paper presents a method for learning semantic parsers (systems for mapping natural language to logical form) based on a statistical model. That is, how to provide an interpretation (meaning) for a request by a user formulated in natural language on the basis of statistical information collected in a training phase.

The parser is mainly predicated on the idea that similarities exist between contexts in which individual parsing actions take place. In other words, this is not only the contribution and interaction in meaning of each word (or group of words) that make up the final meaning of the sentence, but their relative position in that sentence. Those similarities are then used to compute the degree of certitude of a particular parse. In the *training stage*, information is collected over the unfolding of each parse; from this collection of information, decisions are made on the basis of three criteria for the *parsing stage*: the similarities between the contexts in which the action took place, the similarities between the final meaning representation and finally, the sheer number of occurrences of those actions and final representations.

Finally, a module is provided so that training and parsing can also be done on a changing domain such as newspaper browsing.

1 Introduction

The problem of finding a good interpretation to natural language utterances is a very difficult one. *Semantic parsing*, as it is known, is described in [1] as:

The process of mapping a natural language input (a sentence) to some structured meaning representation which is suitable for manipulation by a machine.

Semantic parsing is difficult because it involves the concept of *natural language understanding*. Classical methods use hand-written rules and formal semantics to build up a suitable representation. Besides the heavy burden of finding appropriate rules for this task, formal semantics relies on the principle of *compositionality*, which does not take into account the non negligible idiomatic nature of natural language. More recently, corpus-based methods have received much attention to overcome these problems.

They have been applied with success in areas like speech recognition ([15, 13]), part-of-speech tagging ([10]), syntactic parsing ([16, 4, 5, 7, 11]) and text or discourse segmentation ([12]). They allow to build systems which satisfy desirable properties of natural language processing (NLP) applications. These properties ([2]) are:

1. Acquisition, i.e. automatically acquiring knowledge (domain specific or not) that would be necessary for the task.
2. Coverage, i.e. handling the potentially wide range of possibilities that could arise in the application.
3. Robustness, i.e. accommodating real data which may not be "perfect" (like having noise) and still being able to perform reasonably well.
4. Portability, i.e. easily applicable to a different task in a new domain.

Corpus-based methods rely on a training corpus which is, in general, a collection of sentence-meaning pairs (for the task of semantic parsing). Two main areas of corpus-based approaches for semantic parsing can be distinguished: the *Machine Learning* approach and the *Statistical approach*.

In the Machine Learning approach, some learning algorithm is used to learn how to go from utterances to meaning. Statistical approaches collect a number of parameters from the training corpus to help a semantic parser discriminate between good and less good meanings. Some approaches use both.

In this paper, we examine a fully statistical, corpus-based approach to build semantic representations. The semantic parser is a variant of a *shift-reduce* (see 4.1) syntactic parser, which maps natural language utterances to *First*

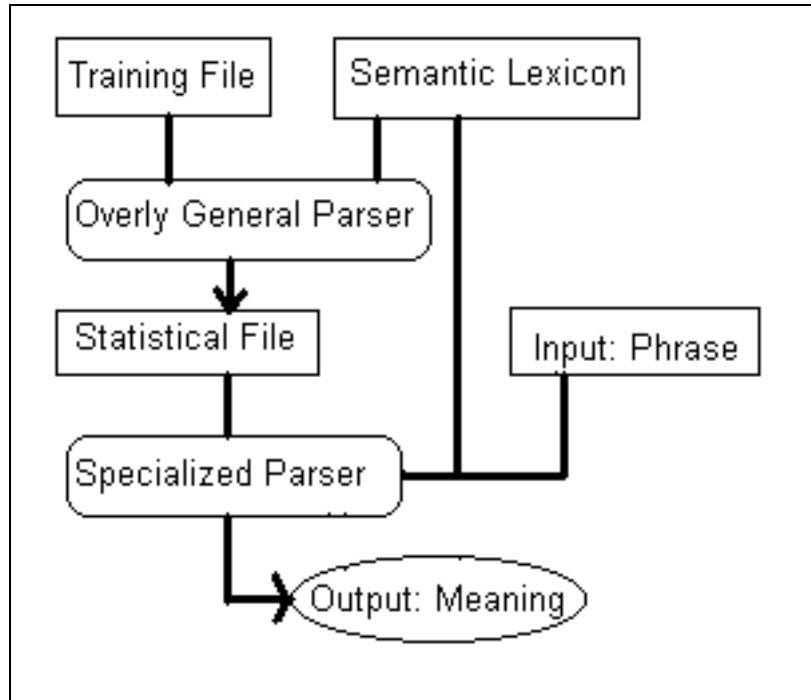


Figure 1: The Parser Architecture

Order Logic (see 2.3) meaning representations. The rest of the paper is organized as follows. Section 2 provides the background necessary in semantic statistical parsing. Section 3 reviews previous work on semantic parsing. Section 4 introduces the mechanisms involved in our parser before going through a simple parsing example without statistics. Section 5 focus on the training phase and section 6 on the parsing phase with statistics. We introduce a useful area of research for corpus-based semantic parsing in section 7 before concluding in section 8. Figure 1 shows the overall architecture of the system.

2 Background

2.1 Semantic Parsing

Traditionally, methods used to construct semantic parsers very often involve the creation of rules by a knowledge expert. Those hand-crafted rules made the parser incomplete, even for a specific domain. As knowledge to encode grew in size, hand-crafted becomes more and more difficult as we approach the so-called *knowledge engineering bottleneck*. The result was a very inefficient and fragile parser.

More recent approaches tend to avoid this knowledge-engineering perspective in favor of an empirical, corpus-based approach where parsers are constructed through learning. This is the perspective we are taking, and we have built a parser which learns how to parse new natural language utterances by collecting statistics on a *training corpus* of those utterances, and by applying a statistical model to choose the parse with the highest probability. Our domain is German newspapers browsing, and we want to use it in a context where people could navigate through their favorite newspaper by using natural language for searching or browsing.

2.2 Statistical Parsing

To get a good idea on how semantic parsing is performed, it is instructive to look at syntactic parsing first. In syntactic parsing, *probabilistic context-free grammars* (PCFGs) ([5]), *probabilistic left-corner grammars* (PLCGs) ([4]), *dependency grammars* ([7]), and *maximum entropy model* ([16]) have been developed for building syntactic trees. In particular, a PCFG-based system is trying to find a parse P such that:

$$P = \max_t \mathcal{P}(t, s|G)^1 \quad (1)$$

i.e. to find the parse \mathcal{P} with the highest probability, given a grammar G , where t is a parse tree, s a sentence and where each grammar rule is assigned a probability according to its frequency in a corpus. In semantic parsing, we are trying to find a parse P such that:

$$P = \max_f \mathcal{P}(f, s|L) \quad (2)$$

i.e. to find the parse \mathcal{P} with the highest probability, given a first-order logical language L , where f is a formula, s a sentence and where each formula is

¹ $P(A|B)$ means the probability of A given B.

assigned a probability according to its frequency in a corpus.

More precisely, the probability of a syntactic parse tree amounts to the probability of each subtrees multiplied together, while the probability of a semantic parse is the probability of each parsing actions multiplied together.

2.3 First Order Logic

As of now there has not been developed a "universal" semantic representation language covering the world of possible meaning structures. But the semantics offered by first-order languages is expressive enough to provide a good starting point for encoding the interpretation of natural language, and has the advantage of being well documented. First order languages are define by the five following elements:

- A vocabulary. This includes atomic terms such as *Artikel*, as well as predicates such as *suche(-,-)*
- A set of variables A, B, C, \dots, X, Y, Z
- Boolean operators: negation (\sim), implication (\supset), disjunction (\vee) and conjunction ($\&$)
- The universal quantifier (\forall) and the existential quantifier (\exists)
- Punctuation symbols () and ,

In order to build semantic representation out of these basic elements, one must first define terms. Terms are either constant (such as *Zeitung*) or variable. A basic semantic formula is defined as follows:

- $\text{predicate}(\text{term1}, \text{term2}, \dots)$

A full semantic representation is obtained via a well-formed formula²:

- All basic formula are well-formed formula
- A Boolean operator applies to a well-formed formula is also a well-formed formula
- A quantifier operator applies to a well-formed formula is also a well-formula
- Nothing else is a well-formed formula

Example 3 is a well-formed formula.

(3) $\text{forall}(A, \text{exists}(B, \text{artikel}(B) \& \text{suche}(A, B)) \& \text{zeitung}(A) \supset \text{return}(A))$

The parser we are going to describe does not, however, include the treatment of *questions*. This means that *variables* are not required in our model.

²Taken from [3].

3 Previous work

3.1 Semantic Tagging

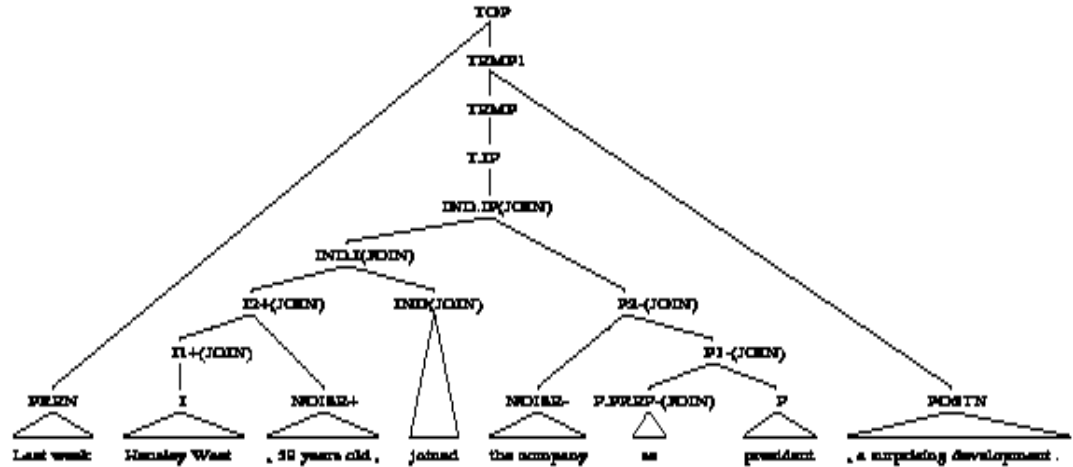
Collins and Miller [6] describes a statistical model for extraction of *events*. This approach is typically useful in Information Extraction, but it is instructive to examine briefly how it works. As figure 2 shows, the approach is based on a Probabilistic Context-Free (Semantic) Grammar (PCFG), a set of hand-crafted rules for a specific domain (management succession in this case) to tag sentences with three slots: the post, the person coming to the post and the person leaving the post. In the training phase, a set of training examples are annotated (or labeled) with the context-free rules, and statistics are collected (basically the number of occurrences of each rules is counted) to make up the PCFG. The authors report that after training on 560 sentences and testing on 356 sentences, the accuracy is 77.5%.

Although the approach offers a simple mechanism to extract basic information from natural languages texts, it suffers mainly from the following two common problems for a traditional approach:

1. The burden of having to create a set of adequate rules for parsing.
2. The fact that those rules are tied to one specific domain makes them useless when the domain changes.

3.2 Machine Learning approach

Thompson, Mooney and Tang [9] propose a novel and very interesting approach to the problem of semantic parsing. First, their system (called *CHILL*) is based on a bottom-up parser (see 4.1). The idea is that the variant of the parser used has a finite set of actions applicable at each step of the parsing process. Figure 3 shows the different stages leading to the final parser. During *Parsing Operator Generation*, a set of a general actions templates is instantiated to a much bigger set of specific parsing actions. This set of actions produces many spurious parses for each training examples, because they do not take into account the various context information for each action. These contexts, such as the *Parse Stack* and the *Input String* will constraint those actions into a more limited range of useful instances. These contexts are collected during the *Example Analysis* phase. In *Control Rule Induction*, a machine learning algorithm called *ILP* (Inductive Logic Programming) is applied to induce general *Control Rules*, which will serve as "guard" for each actions. These control rules are incorporated into the initial



Rule	Interpretation
TOP \Rightarrow PREN TEMP1	Choose to have pre-noise (PREN)
TEMP1 \Rightarrow TEMP POSTN	Choose to have post-noise (POSTN)
TEMP \Rightarrow T,IP	Choose to have IN and POST slots
T,IP \Rightarrow IND,IP(JOIN)	Choose to use a member of the JOIN class of indicators
IND,IP(JOIN) \Rightarrow IND,I(JOIN) P2-(JOIN)	Generate the POST slot to the right of the indicator
IND,I(JOIN) \Rightarrow I2+(JOIN) IND(JOIN)	Generate the IN slot to the left of the indicator
I2+(JOIN) \Rightarrow I1+(JOIN) NOISE+	Have noise between the IN slot and the indicator
I1+(JOIN) \Rightarrow I	Choose not to have a preposition for the IN slot
P2-(JOIN) \Rightarrow NOISE- P1-(JOIN)	Have noise between the POST slot and the indicator
P1-(JOIN) \Rightarrow P,Prep-(JOIN) P	Choose to have a preposition attached to the POST slot

Figure 2: Semantic tagging for: *Last week, Hensley West, 59 years old, joined the company as president, a surprising development.*

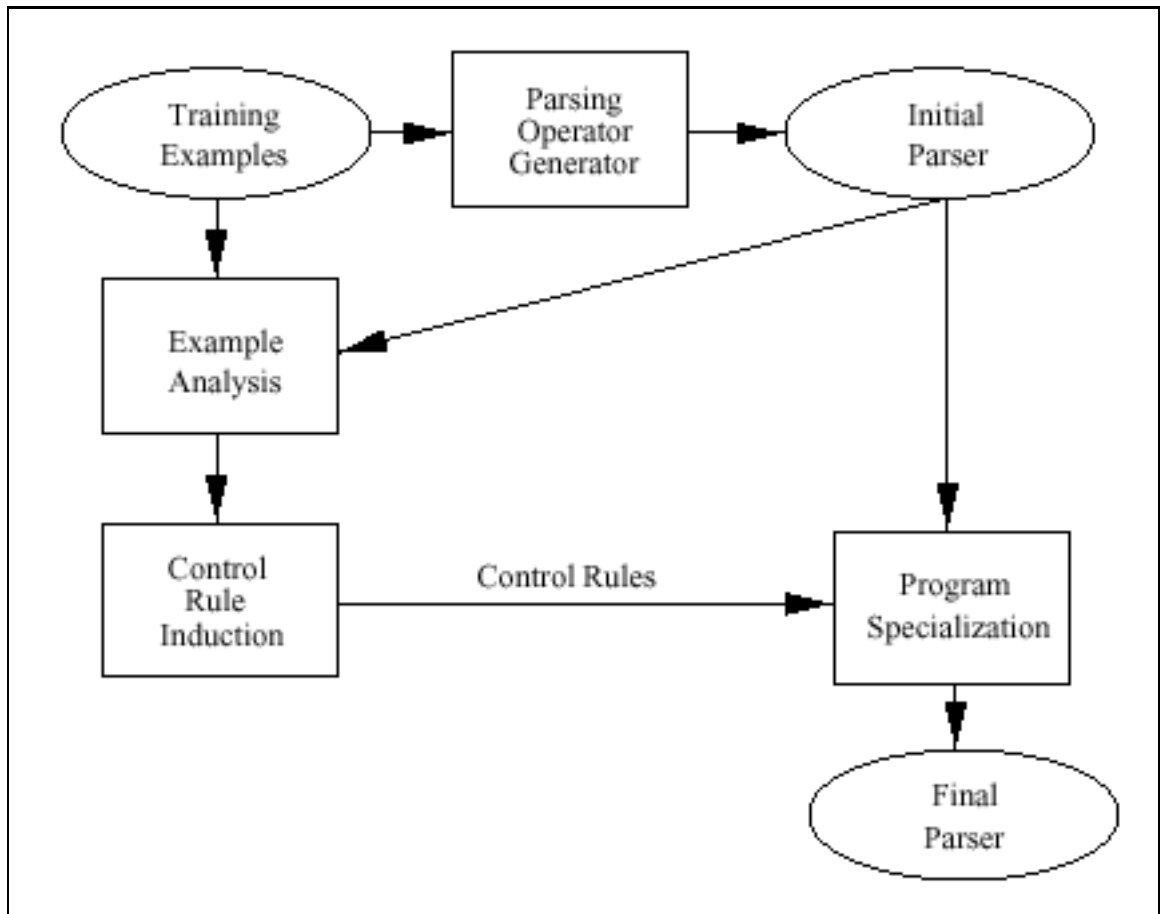


Figure 3: The CHILL Architecture

overly general parser to specialize it³. The authors report that after training on 225 training examples, the accuracy of the system on geography queries is around 68%. The main benefit of this approach is to couple a flexible bottom-up parser with a method for guiding its actions. However, it is not clear whether or not the "guiding" mechanism proposed (the machine learning algorithm) is the best approach. Recently, Tang and Mooney [17] have designed a new learning algorithm called *TABULATE* to learn multiple models from the training data and introduce statistics in their machine learning (logic) method to choose parses with the highest probabilities. They have reported an improvement over the pure logic-based method (CHILL). Our system offers yet a different way of guiding the parser, by means of a pure stochastic model. It is fully described from section 4 onward.

3.3 A Fully Statistical Approach

Miller, Stallard, Bobrow and Schwartz [14] describes an approach entirely based on a trained statistical model. Figure 4 shows the different stages of the parsing process. Processing proceeds in three stages⁴:

1. Word string W arrives at the parsing model. The full space of possible parses T is searched for n -best candidates according to the measure $P(T)*P(W|T)$. These parses, together with probability scores, are passed to the semantic interpretation model.
2. The constrained space of candidate parses T (received from the parsing model), combined with the full space of possible pre-discourse meanings M_S , is searched for n -best candidates according to the measure $P(M_S, T)*P(W|T)$. These pre-discourse meanings, together with their associated probability scores, are passed to the discourse model.
3. The constrained space of candidate pre-discourse meanings M_S (received from the semantic interpretation model), combined with the full space of possible post-discourse meanings M_D , is searched for the single candidate that maximizes $P(M_D|H, M_S, T)*P(W|T)$, conditioned on the current history H . The discourse history is then updated and the post-discourse meaning is returned.

³The authors mentioned that these control rules *would (hopefully) capture all the contextual knowledge present in the parse states necessary for classifying which (future) parse states the action should apply to.*

⁴Taken directly from [14].

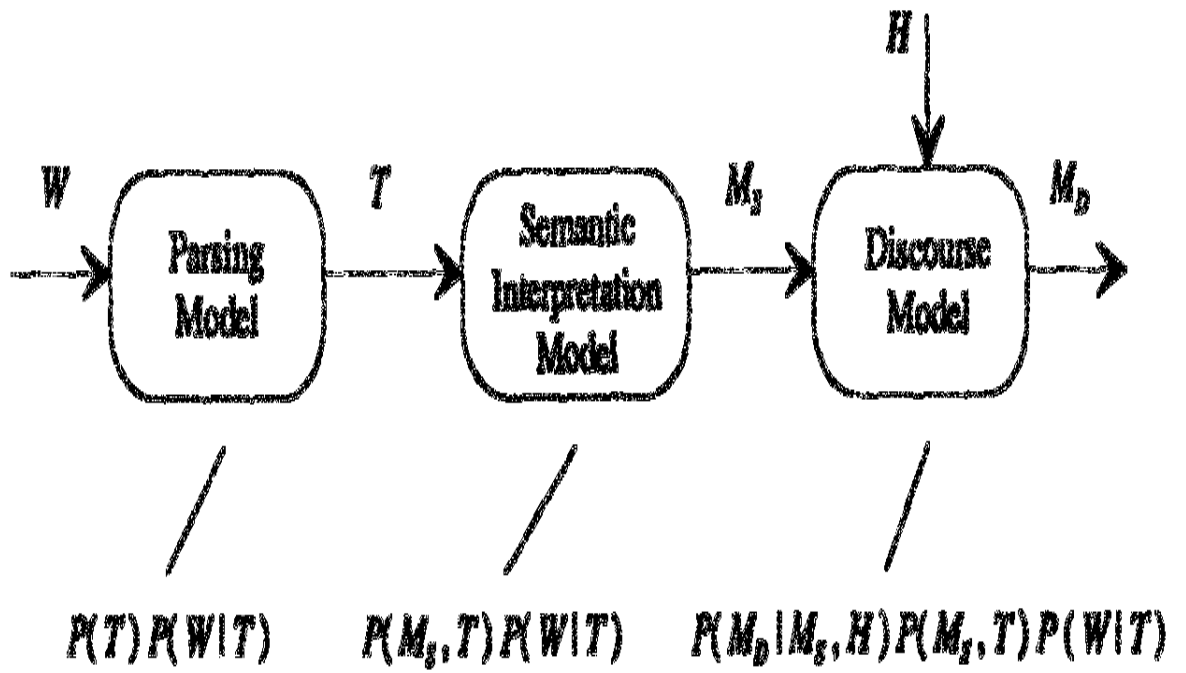


Figure 4: Overview of Miller's statistical processing

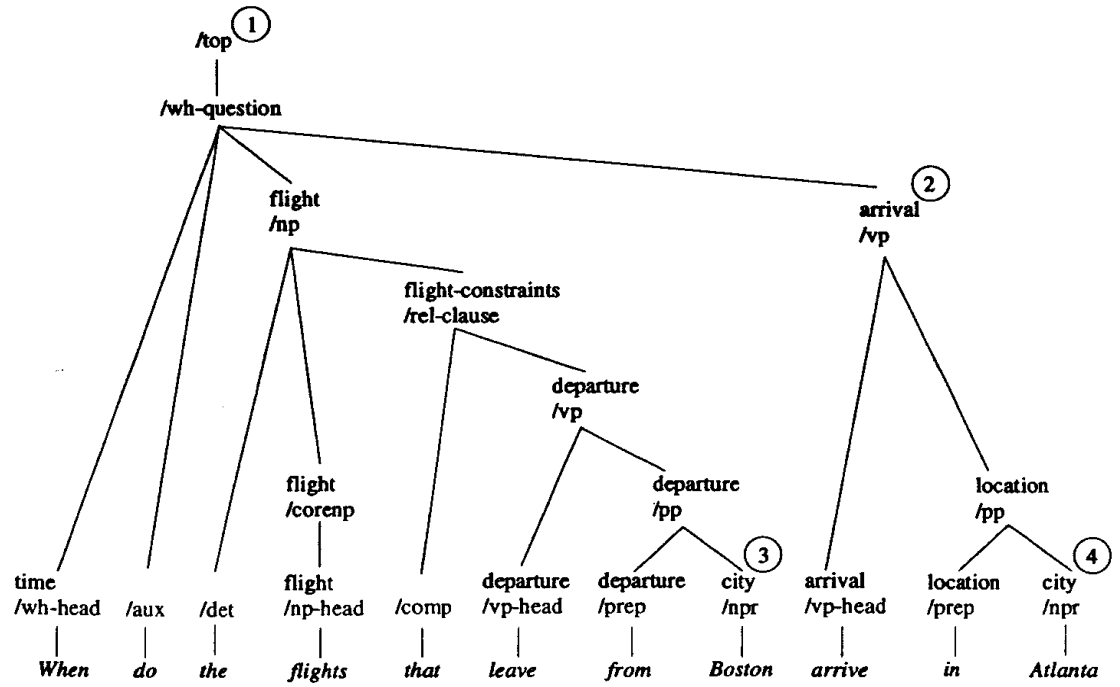


Figure 5: Miller's sample parse tree

A simple parse tree is shown in figure 5. Both pre-discourse and post-discourse meanings are represented using a simple frame representation such as in the following⁵ example from the air transportation domain:

- Show: (Arrival-Time)
- Origin: (City "Boston")
- Destination: (City "Atlanta")

Discourse information includes anaphora, as in the following example⁶:

- USER1: I want to fly from Boston to Denver.
- SYSTEM1: <displays Boston to denver flights>
- USER2: Which flights are available on Tuesday?
- SYSTEM2: <displays Boston to Denver flights for Tuesday>

While training on approximately 4000 sentences, the authors report an error rate of 21.6% when the system was given a new sentence to find a semantic interpretation for. Clearly, the main contribution of this work is to include the treatment of *discourse* information within the model. However, the system suffers from the same problems as the semantic tagger presented in section 3.1, that is, a lack of portability from one domain to another, given the hand-crafted nature of the rules used in parsing.

4 Overview of the Parsing Process

This section is meant to give a flavor of the parsing process and provides a "light" introduction to the parser. All statistical considerations are for the moment deferred to section 6. The parser used is a variant of a *Shift-Reduce* parser. It actually comprises three different actions that the parser uses to get to the final parse, which is a semantic interpretation (in first-order logic) of a natural language utterance. We first introduce some concepts of the *Shift-Reduce* parser, present the various elements making up our *specialized* parser and then go through a concrete parsing example.

4.1 The SHIFT-REDUCE parser

A *shift-reduce* parser [8] is a parser using a *BOTTOM-UP* parsing algorithm. A bottom-up parser accepts words and tries to combine them as constituents. Figure 6 shows a *shift-reduce* parse for *Der Hund grübt* (The dog digs). The

⁵Taken from [14].

⁶Taken from [14].

Step	Action	Parse Stack	Input String
	(Start)		Der Hund gräbt
1	Shift	Der	Hund gräbt
2	Reduce	D	Hund gräbt
3	Shift	D Hund	gräbt
4	Reduce	D N	gräbt
5	Reduce	NP	gräbt
6	Shift	NP gräbt	
7	Reduce	NP V	
8	Reduce	NP VP	
9	Reduce	S	

Figure 6: Shift-Reduce Parser

shift-reduce algorithm is as follows:

1. Shift a word onto the stack.
2. Reduce the stack repeatedly using lexical entries and PS rules, until no further reductions are possible.
3. If there are more words in the input string, go to Step 1. Otherwise stop.

One key characteristic of the shift-reduce parser is that it has no expectations. You give it an input string, and it will tell you what kind of constituent it is. You are not limited to parse specific constituents such as *S*, *NP*, etc. This is well-suited for semantic parsing, since we do not know beforehand what is the meaning of the input string, although we can make some validity checks on the format of the semantics that comes out of the parse. We will modify this algorithm to adapt it for semantic parsing in section 4.10.

4.2 The Input String

The input string is a list of words to give an interpretation for. When no actions are applicable and the input string is empty, then the parsing process is completed. Typically, one word is removed from the list for a *SHIFT* action, one or more for a *INTRODUCE* action and none for a *DROP* action. It gives some contextual information while applying a parsing action. For example:

(4) [ich,suche,einen,Artikel,über,Condit]
is an input string⁷.

4.3 The Parse Stack

The parse stack is the actual parse state, the current interpretation of the input string found so far. It is a list of binary terms, each term representing a combination of the introduced predicate (or concept) with its context of introduction. The context gives partial (but useful) information on the words following the concept at the time of introduction. Each concept must be in the semantic lexicon (see 4.9). Here is the general format of the parse stack followed by an example:

(5) [concept1:[context1],concept2:[context2],...]

(6) [suche([],zeitung(_),zeit(_)):suche,einen,Artikel],start:[ich]]

Example 6 shows that the concept *suche* (which has three arguments) was introduced in the context of *suche einen Artikel* following it. The first argument ([]) will hold the list of topics (see 4.12) which are searched for. The second and third arguments are placeholders for the newspaper and the time period of the search. The types of arguments can be defined in the semantic lexicon (see 4.9). The *start* predicate is there only to provide room for words from the input string which would be shifted at the very beginning of the parse; *start* does NOT contribute to the meaning of the input phrase.

4.4 The sHIFT action

Syntax:

sSHIFT(word_to_be_shifted)

A *sSHIFT* action simply puts the first word from the input string into the end of the context of the concept on the top of the parse stack. For example, the action *sSHIFT(über)* on the parse stack 6 would result in the following new parse stack:

(7) [suche([],zeitung(_),zeit(_)):suche,einen,Artikel,über],start:[ich]]

⁷Although not yet topicalized, see section 4.12.

4.5 The iNTRODUCE action

Syntax:

iNTRODUCE(*concept_to_be_introduced*)

The iNTRODUCE action takes a concept from the semantic lexicon and puts it on the top of the parse stack, initializing its context of introduction to the word (or list of words) that triggered (see 4.9) this concept. These concepts will then participate to the meaning representation. For example, the action *iNTRODUCE(topic(1))* on the parse stack 6 would result in the following new parse stack:

(8) [topic(1):[topic(1)],suche([],zeitung(_),zeit(_)):[suche,einen,Artikel],start:[ich]]

4.6 The dROP action

Syntax:

dROP(*source_term*, *target_term*)

The dROP action attempts to place a term from the parse stack as argument to another term of the parse stack. The context of the source term is lost in the process. Some restrictions might be imposed in the semantic lexicon to prevent some undesirable combinations. For example, only newspaper names can be dropped into the *zeitung(_)* argument of the concept *suche*. This action has no effect on the input string. For example, the action *dROP(topic(1),suche([],zeitung(_),zeit(_)))* on the parse stack 8 would result in the following new parse stack:

(9) [suche([topic(1)],zeitung(_),zeit(_)):[suche,einen,Artikel],start:[ich]]

4.7 A Parse State (no statistics)

A parse state is given by the following expression:

op(aCTION(arguments)#Parse_Stack#Input_String)

It indicates in which context, i.e. how the Parse Stack and the Input String looked like, when the action took place. *Op* is simply a container⁸ for all types of actions.

⁸A container is a term which subsumes other terms.

4.8 A Final State

A final state is given by the following expression:

```
final(Parse_Stack)
```

It indicates the final aspect of a parse, i.e. the meaning we have found for an input string.

4.9 Semantic Lexicon

We also need a semantic lexicon. It comprises all the concepts and their triggering phrase(s) that we wish our parser to process. A triggering phrase is simply a word (or group of words) in the input string that triggers some concept. For example, *suche* or *ich brauche* would trigger the *suche* concept. The format of a lexical entry is:

```
lexicon(CONCEPT, [TRIGGERING_PHRASE]).
```

Here are two examples of lexical entries:

```
lexicon(topic(1),[topic(1)]).  
lexicon(suche([],zeitung(_),zeit(_)),[suche]).
```

The term *topic(1)* is a placeholder for the interpretation of successive topics (here it is the first one in the input); this means that if an input phrase holds for example the topic *der Krieg*, then *topic(1)* is some sort of synonym for it during the time of parsing. *Der Krieg* is only reintroduced at the end of the (hopefully successful) parse. The term *zeitung(_)* is a placeholder for newspapers and *zeit(_)* for the time period of the search. The underscore (*_*) signifies that it is still not yet specified. The type of the arguments for *[]*, *zeit* and *zeitung* must be specified in the semantic lexicon file. See APPENDIX for more examples.

4.10 Our variant of the shift-reduce parser

We are now ready to present the variant of the shift-reduce parser we are using. The algorithm presented in section 4.1 must be modified as follows:

1. Try to *introduce* a new concept or *shift* a word.
2. If possible, make one *drop* action.
3. If there are more words in the input string, go back to Step 1. Otherwise stop.

4.11 The programming language used

PROLOG (*PRO*gramming in *LOGic*) is used as programming language to implement the shift-reduce algorithm presented in section 4.10. The interesting characteristic of PROLOG for us here is that it uses the *backtracking* mechanism to generate all possible solutions available for one parse. This is the mechanism that allows to implement a beam⁹ in our parser.

4.12 Topic_extraction

First, the parser goes through a preprocessing phase that we have termed *topic_extraction*. This phase is useful and necessary when we have a changing domain such as newspapers. We explain it here briefly. The idea behind *topic_extraction* is that we don't want the parser to learn about specific topics from newspapers for a particular time frame, since this learning process is not likely to be useful at parsing time. Therefore, only "topicalized" sentences should appear in the training corpus for the learning process to take place adequately. For our running example 4, the preprocessing phase returns a single topic:

[über,Condit](Condit)

This means that the part of the phrase [*über,Condit*] is considered as a particular topic, and its meaning is *Condit*. Note that for newspapers, the word *Artikel* is not considered as a relevant topic. Therefore, after *topic_extraction*, this is the following phrase which is passed on to the parser:

[ich,suche,einen,Artikel,topic(1)]

The term *topic(1)* becomes a placeholder for the first topic in the phrase, and the parser will replace it by its actual meaning (*Condit*) in the final (hopefully successful) parse. Typically, *topic_extraction* replaces relevant noun phrases or prepositional phrases in the input string by successive *topic()* terms¹⁰. This phase is part of a module in the parser which can be considered as a *plug in*; if topics are relevant for the learning process, then the plug-in can be omitted.

⁹A beam is the word used to express that we are not considering all possible solutions, but a subset of them. With the backtracking mechanism of PROLOG, this means the very first n solutions, n being the size of the beam.

¹⁰When adjectives are present in noun phrases, the *topic_extraction* phase returns more than one interpretations for the topic: one with adjectives, and one without. For example, [in,den,osteuropäischen,Staaten]((osteuropäischen+Staaten),(Staaten)).

4.13 A parsing example

We now show a parse for *Ich suche einen Artikel über Condit*. We assume for simplicity that the parser always takes the best available action. The following trace presents the successive actions taken by the parser. The initial parse stack and input string states for *Ich suche einen Artikel über Condit* are:

[start:[]]

and

[ich,suche,einen,Artikel,topic(1)]

Here is the complete parse, using the two lexical entries shown in section 4.9. Each line represents a Parse State (see 4.7):

- sSHIFT(ich)#[start:[]]#[ich,suche,einen,Artikel,topic(1)]

The word *ich* is not in the semantic lexicon¹¹, so the only action possible is to shift it on the parse stack.

- iINTRODUCE(suche([],zeitung(-),zeit(-)))#[start:[ich]]#[suche,einen,Artikel,topic(1)]

The word *suche* is in the lexicon, so it can be introduced as a new predicate on the parse stack. Another possibility would be to shift it.

- sSHIFT(einen)#[suche([],zeitung(-),zeit(-)):[suche],start:[ich]]#[einen,Artikel,topic(1)]

The word *einen* is not in the lexicon, it must be shifted.

- sSHIFT(Artikel)#[suche([],zeitung(-),zeit(-)):[suche,einen],start:[ich]]#[Artikel,topic(1)]

The word *Artikel* is not in the lexicon (it is actually an *empty_word* for the newspaper domain) and is therefore shifted.

- iINTRODUCE(topic(1))#[suche([],zeitung(-),zeit(-)):[suche,einen,Artikel],start:[ich]]#[topic(1)]

topic(1) is in the lexicon, so it can be introduced.

- dROP(topic(1),suche([],zeitung(-),zeit(-)))
#[topic(1):[topic(1)],suche([],zeitung(-),zeit(-)):[suche,einen,Artikel],start:[ich]]
#[[]]

We can drop the predicate *topic(1)* into the first argument of the *suche* predicate. The final parse stack¹² or final state (see section 4.8) is:

¹¹Because *ich* is not semantically relevant for interpreting newspaper browsing commands or search.

¹²To be valid, a final Parse Stack must have only two predicates (including start) and no unwanted unbound variables. However, these constraints could be relaxed, especially if we want to parse *questions*.

[suche([topic(1)],zeitung(_),zeit(_)):suche,einen,Artikel],start:[ich]]

In the final stage, the parser simply puts back the meaning for *topic(1)* collected during the *topic_extraction* phase and the final parse is, without contextual information:

suche([(Condit)],zeitung(_),zeit(_))

which signals a search for the topic *Condit* with no specific newspaper or time frame.

5 Training

The training phase uses an overly general parser, which produces all possible path of actions (only limited by the *search beam*, see 5.1) to be taken in order to get from each training example utterance to its semantic representation. In the process, it records successful actions (called *op*), as well as the different final states. For each of them uniquely defined, it assigns a frequency measure, defined as follows:

$$Frequency = \frac{Occurrence_of_a_particular_action_in_a_specific_context}{Total_number_of_occurrence_of_this_action} \quad (10)$$

5.1 Training file

The training file is the file in which training examples are stored. These examples have the following format:

training([topicalized_phrase], meaning).

Meaning is in the form of first order logic expressions (see section 2.3). A training file example could be:

- training([ich,möchte,jetzt,eine,neue,Suche,beginnen],neue_suche).
- training([bitte,bearbeiten,Sie,meinen,Suchauftrag],bestätigen(suche([],zeitung(_),zeit(_)))).
- training([ich,suche,einen,Artikel,topic(1)],suche([topic(1)],zeitung(_),zeit(_))).

Note that the examples for which we wish to train for should be topicalized (see 4.12) for a changing domain. The third example could be helpful in training for a sentence like:

Ich suche einen Artikel über Condit.

While training, an *overlyGeneralParser* is used. It is overly general in the sense that it tries any possible actions to get to the final parse, without considering any information (such as *statistics*) that could be helpful to guide the parsing process.

In training, a *training beam* can be specified. This means that only a certain number of parses will be recorded in the *statistical file* (see 5.2) for each training example.

5.2 File used by the Specialized Parser: the Statistical File

The *overlyGeneralParser* parses the *training file* (see 5.1) to generate the *statistical file*. Every step needed to go from the *topicalized phrase* to the *meaning* (see 5.1) is recorded, as well as final states themselves. Final states (see 4.8) are simply the states of the parse stack themselves at the end of the parse. Each of them (actions and final states) are assigned a frequency measure as described previously (see section 5). Each line has either one of the following format (recall that *op* is a container for any action):

- `op(ACTION#PARSE_STACK#INPUT_STRING#FREQUENCY).`
- `final(FINAL_STATE#FREQUENCY).`

Here are two examples:

- `op(SHIFT(ich)#[start:[]][ich,suche,einen,Text,for,topic(1),topic(2),bitte,bearbeiten,Sie,meinen,Suchauftrag]#0.3333).`
- `final([bestätigen(neue_suche):[bearbeiten,Sie,meinen,Suchauftrag],start:[ich,möchte,jetzt,eine])#0.2).`

These lines are used by the *specializedParser* to compute the best parse. The next section describes the statistical parsing process.

6 Statistical Parsing

The actual parsing of the input phrase is done by a *specializedParser*. It is specialized in the sense that it uses a statistical model to process all the information available from the training phase in order to get the best possible parse (the one with the highest probability). This section presents a detailed description of the statistical model used. In particular, it shows how different parameters in the model can be adapted to influence the parsing phase outcome. Some default values for the parameters are discussed.

6.1 The Search space

Like in the training phase, the most obvious way to influence the parse is to tell the parser how many parses it should try before taking a decision. We call it the *search beam* parameter.

6.2 Measure of similarity between lists

This is a crucial aspect of the specialized parser. When the parser tries to choose a suitable parse, it must compare list of words (to compare *Actions*, *Parse stacks* or *Input strings*). A good *similarity* measure between lists is essential, but because computing similarity is very demanding on computer resources, one must find a trade-off that preserves computational efficiency. At the top level, the similarity measure is simply a measure of the number of identical elements in both lists, divided by the size of the greatest list. Therefore, we have:

$$\textit{Similarity} = \frac{\textit{Number_of_identical_elements}}{\textit{Size_of_the_biggest_List}} \quad (11)$$

For example, omitting case-sensitivity:

$$(12) \text{ similarity}([\textit{Ich,suche,einen,Artikel}],[\textit{Das,suche,ich}]) = 2/4 = 0.5$$

Comparisons sometimes involves comparing structures. Structures can be compared by using the following identity relation, readily available in PROLOG:

$$\textit{predicate}(\textit{arg1,arg2,...}) = [\textit{predicate,arg1,arg2,...}]$$

This method allows us to compare structures as lists.

The situation gets a little more complicated when we have a list of lists. Each sublist has its own measure of similarity. The problem is when to consider two sublists as similar enough (a *threshold* of similarity) so that they won't be compared further with other sublists. By doing so, we choose efficiency (by not comparing all elements of each list) at the cost of some approximation of similarity. For example, if we have the following two lists of lists:

$$(13) [\textit{sublista1,sublista2}]$$

$$(14) [\textit{sublistb1,sublistb2,sublistb3}]$$

and that we have the following similarity measures:

- $\textit{similarity}(\textit{sublista1,sublistb1},0.4)$

- $\text{similarity}(\text{sublista1}, \text{sublistb2}, 0.6)$
- $\text{similarity}(\text{sublista1}, \text{sublistb3}, 0.9)$
- $\text{similarity}(\text{sublista2}, \text{sublistb1}, 0.3)$
- $\text{similarity}(\text{sublista2}, \text{sublistb2}, 0.1)$
- $\text{similarity}(\text{sublista2}, \text{sublistb3}, 0.2)$

Suppose we have a threshold of 0.5. This means that two *sublists* must have a similarity of at least 0.5 to be considered similar. While comparing *sublista1* with other members of the second list, we first find a similarity of 0.4, which is smaller than the threshold, so they are considered dissimilar. The second comparison gives 0.6, which is greater than the threshold, so both sublists are considered similar. At this point, they are both removed from their respective list, and the comparison process goes to *sublista2*. Comparisons are made, only to find out that the three of them (0.3, 0.1 and 0.2) are lower than the threshold, so they are considered dissimilar. The result of similarity is then $0.6/3 = 0.2$. It is easy to see where approximation takes place. First, *sublista1* has been found similar with 0.6 to *sublistb2*, while it was still more similar to *sublistb3* with 0.9. Second, because the similarity measures for *sublista2* are all below 0.5, these three comparisons give complete dissimilarity outcome, while it could have been 0.3 with *sublistb1*. All in all, this would give a similarity measure of $(0.9+0.3)/3 = 0.4$, instead of the 0.2 previously found. Besides the advantage of computational efficiency, always underestimating lists similarity counterbalances overestimating in some cases, specifically in the following two measures:

- Assimilating natural language utterances to *sets* instead of *lists*. Natural languages utterances are in general best assimilated to lists, except maybe for free word order languages.
- Assimilating first-order expressions to sets.

However, it is not clear whether or not these assumptions are correct; but it is also rather difficult to prove that *a priori*, they are not. Preliminary empirical results¹³ tend to show that they are sensible.

6.3 Parametrizing the model

We introduce all the equations for the model, explaining their context of use. The best parse P is found by taking the highest probability \mathcal{P}_i among

¹³See APPENDIX for details of the results.

the possible parses (limited by the search beam) available:

$$P = \max_i \mathcal{P}_i \quad (15)$$

Each of these parses \mathcal{P}_i have a probability that amounts to multiplying the probability of the individual *op* or actions together ($\prod_k a_k$) and adding the probability of the final state (*ProbF*, see equation 22). These two components are weighted by *Pop* (default 0.8) and *Pfinal* (default 0.2). The default values reflects the idea that because their probabilities are multiplied together, actions get an overall low probability compared to the *ProbF*. By choosing to weight them with 0.8, we acknowledge the fact that although the similarity of a final state with one in the statistical file is very important (even with a weight of 0.2, they contribute in average around 90% of the probability \mathcal{P}_i), the weighting of the actions taken together must be high enough to discriminate among similar final states, should that case arise. Multiplying by 100 gives a more readable value between 0 and 100.

$$\mathcal{P}_i = (Pop * (\prod_k a_k) + Pfinal * ProbF) * 100 \quad (16)$$

The way each *op* a_k is assigned a probability is by taking into account its similarity with one of the *ops* in the statistical file (see equation 18) as well as the frequency of this *op* (*Frequency*). These two components are also weighted by *Pop_sim* (default 0.5) and *Pop_occ* (default 0.5). The default values reflects the idea that the similarity of an *op* with one in the statistical file is as important as the frequency of this *op*.

$$a_k = \max_m (Pop_sim * \mathcal{P}_m + Pop_occ * Frequency) \quad (17)$$

While looking for a suitable *op* in the statistical file, the parser looks for similarity. The similarity of an *op* \mathcal{P}_m with one in the statistical file is measured by multiplying together the similarity of the *op* as such, the similarity of the *parse stack* and the similarity of the *input string* (*t* stands for *training*).

$$\mathcal{P}_m = \max(sim_A(op_action, t_{op}) * sim_PS(op_{ps}, t_{ps}) * sim_I(op_{input}, t_{input})) \quad (18)$$

The following three equations show how smoothing¹⁴ of similarities is carried out. For example, the minimum value of the similarity between two actions *smoothingAction* is a fraction *smoothing_Action* of the minimum similarity *min_Action* found by random testing on a representative set of *ops* and *final* states. Those minimum values found are:

¹⁴Smoothing is the process that assigns a minimum value of probability in case that the computed probability is zero.

- $\text{min_Action} = 0.5$
- $\text{min_PS} = 0.1111$
- $\text{min_Input} = 0.0667$

and the default fractions used are:

- $\text{smoothing_Action} = 0.1$
- $\text{smoothing_PS} = 0.5$
- $\text{smoothing_Input} = 0.5$

Those default values reflects the idea that care should be taken not to over-estimate the similarity of an action, while it is relatively safe to assign half of the minimum similarity of the *parse stack* or the *input string* in case of a computed zero similarity. Note that similarity between actions is actually only between arguments, while the actions compared *must* of course be the same.

$$\text{smoothingAction} = \text{smoothing_Action} * \text{min_Action} \quad (19)$$

$$\text{smoothingParseStack} = \text{smoothing_PS} * \text{min_PS} \quad (20)$$

$$\text{smoothingInput} = \text{smoothing_Input} * \text{min_Input} \quad (21)$$

Computing the probability of a final parse state is similar to computing the one for actions. A final state probability $\text{Prob}F$ is the weighted sum of the most similar final state in the statistical file P_f (see 23) and the frequency of this final state Frequency :

$$\text{Prob}F = \max_f(P_{\text{final_sim}} * P_f + P_{\text{final_occ}} * \text{Frequency}) \quad (22)$$

$$P_f = \max_n(\text{sim}(t_n, F)) \quad (23)$$

$$\text{smoothingFinal} = \text{smoothing_Final} * \text{min_Final} \quad (24)$$

For smoothing, we have:

- $\text{min_Final} = 0.3333$
- $\text{smoothing_Final} = 0.5$

6.4 A full parse with statistics

We are now going to parse the example shown in section 4.13 by showing how the specialized parser uses statistics (and some rules to constraint the format of the semantics) to find the best parse. Suppose we have the following two training examples:

- training([suche,im,Standard,Informationen,topic(1)],suche([topic(1)],zeitung(standard),zeit(_))).
- training([bitte,einen,Artikel,topic(1),von,gestern],suche([topic(1)],zeitung(_),zeit(tag(-1)))).

which lead, after training (using a training beam of one parse per example), to the following statistical file (once again, *op* represents a container for any action) :

- (25) op(iNTRODUCE(suche([],zeitung(_),zeit(_)))#[start:[]]#[suche,im,Standard,Informationen,topic(1)]#0.5).
- (26) op(iNTRODUCE(zeitung(standard))#[suche([],zeitung(_),zeit(_)):suche],start:[]]#[im,Standard,Informationen,topic(1)]#1.0).
- (27) op(dROP(zeitung(standard),suche([],zeitung(_),zeit(_)))#[zeitung(standard):im,Standard],suche([],zeitung(_),zeit(_)):suche],start:[]]#[Informationen,topic(1)]#1.0).
- (28) op(sHIFT(Informationen)#[suche([],zeitung(standard),zeit(_)):suche],start:[]]#[Informationen,topic(1)]#1.0).
- (29) op(iNTRODUCE(topic(1))#[suche([],zeitung(standard),zeit(_)):suche,Informationen],start:[]]#[topic(1)]#0.5).
- (30) op(dROP(topic(1),suche([],zeitung(standard),zeit(_)))#[topic(1):topic(1)],suche([],zeitung(standard),zeit(_)):suche,Informationen],start:[]]#[topic(1)]#1.0).
- (31) op(sHIFT(bitte)#[start:[]]#[bitte,einen,Artikel,topic(1),von,gestern]#1.0).
- (32) op(sHIFT(einen)#[start:[bitte]]#[einen,Artikel,topic(1),von,gestern]#1.0).
- (33) op(iNTRODUCE(suche([],zeitung(_),zeit(_)))#[start:[bitte,einen]]#[Artikel,topic(1),von,gestern]#0.5).

- (34) $\text{op}(\text{iNTRoDUCE}(\text{topic}(1))\#[\text{suche}([], \text{zeitung}(-), \text{zeit}(-)):\text{[Artikel]}, \text{start}:\text{[bitte, einen]}\#[\text{topic}(1), \text{von, gestern}]\#0.5).$
- (35) $\text{op}(\text{dRoP}(\text{topic}(1), \text{suche}([], \text{zeitung}(-), \text{zeit}(-)))\#[\text{topic}(1):\text{[topic}(1)], \text{suche}([], \text{zeitung}(-), \text{zeit}(-)):\text{[Artikel]}, \text{start}:\text{[bitte, einen]}\#[\text{von, gestern}]\#1.0).$
- (36) $\text{op}(\text{iNTRoDUCE}(\text{zeit}(\text{tag}(-1)))\#[\text{suche}([\text{topic}(1)], \text{zeitung}(-), \text{zeit}(-)):\text{[Artikel]}, \text{start}:\text{[bitte, einen]}\#[\text{von, gestern}]\#1.0).$
- (37) $\text{op}(\text{dRoP}(\text{zeit}(\text{tag}(-1)), \text{suche}([\text{topic}(1)], \text{zeitung}(-), \text{zeit}(-)))\#[\text{zeit}(\text{tag}(-1)):\text{[von, gestern]}, \text{suche}([\text{topic}(1)], \text{zeitung}(-), \text{zeit}(-)):\text{[Artikel]}, \text{start}:\text{[bitte, einen]}\#[\text{[]}\#1.0).$
- (38) $\text{final}([\text{suche}([\text{topic}(1)], \text{zeitung}(\text{standard}), \text{zeit}(-)):\text{[suche, Informationen]}, \text{start}:\text{[]}\#0.5).$
- (39) $\text{final}([\text{suche}([\text{topic}(1)], \text{zeitung}(-), \text{zeit}(\text{tag}(-1))):\text{[Artikel]}, \text{start}:\text{[bitte, einen]}\#0.5).$

Note that three lexical entries had to be added to the semantic lexicon of section 4.9 for the `overlyGeneralParser` to train properly:

```
lexicon(suche([], zeitung(-), zeit(-)), [Informationen]).
lexicon(zeit(tag(-1)), [von, gestern]).
lexicon(zeitung(standard), [im, Standard]).
```

We now ask the `specializedParser` to parse *Ich suche einen Artikel über Condit* using a search beam of 5 parses. Once again, we show the calculations for the *best* parse only (the probability is the last value on each line).

- $\text{sHIFT}(\text{ich})\#[\text{start}:\text{[]}\#[\text{ich, suche, einen, Artikel, topic}(1)]\#0.5125$

Among all the `sHIFT` actions in the statistical file, line 31 was the best suited (or most similar). So, using equation 17 for this action:

$$a_k = \max_m (\text{Pop_sim} * \mathcal{P}_m + \text{Pop_occ} * \text{Frequency}) \quad (40)$$

$$a = (0.5 * \mathcal{P} + 0.5 * 1.0) \quad (41)$$

and equation 19 to compute individual similarities:

$$\text{sim}_A(\text{sHIFT}(\text{ich}), \text{sHIFT}(\text{bitte})) = 0.1 * 0.5 = 0.05 \text{ (smoothed)}$$

$$\text{sim}_{PS}([\text{start} : []], [\text{start} : []]) = 1.0$$

$$\text{sim}_I([\text{ich, suche, einen, Artikel, topic}(1)], [\text{bitte, einen, Artikel, topic}(1), \text{von, gestern}]) = 0.5$$

Now equation 18 gives us the overall similarity:

$$\mathcal{P} = (0.05 * 1.0 * 0.5) = 0.025 \quad (42)$$

Putting it all back to 41, we have:

$$a = (0.5 * 0.025 + 0.5 * 1.0) = 0.5125 \quad (43)$$

The parser will do that for each of the parsing actions, leading to the following results:

- iNTRODUCE(suche([],zeitung(-),zeit(-)))#[start:[ich]]#[suche,einen,Artikel,topic(1)]#0.2583
- sHIFT(einen)#[suche([],zeitung(-),zeit(-)):[suche],start:[ich]]#[einen,Artikel,topic(1)]#0.6
- sHIFT(Artikel)#[suche([],zeitung(-),zeit(-)):[suche,einen],start:[ich]]#[Artikel,topic(1)]#0.5042
- iNTRODUCE(topic(1))#[suche([],zeitung(-),zeit(-)):[suche,einen,Artikel],start:[ich]]#[topic(1)]#0.4167
- dROP(topic(1),suche([],zeitung(-),zeit(-)))#[topic(1):[topic(1)],suche([],zeitung(-),zeit(-)):[suche,einen,Artikel],start:[ich]]#[topic(1)]#0.6389

The parser has now reached a suitable parse or final state,

[suche([topic(1)],zeitung(-),zeit(-)):[suche,einen,Artikel],start:[ich]]

and looks in the statistical file to assign it a probability. The most suitable (similar) final state in the statistical file is 38, and equation 23 is used to compute similarity \mathcal{P} between:

[suche([topic(1)],zeitung(standard),zeit(-)):[suche,Informationen],start:[]]

and

[suche([topic(1)],zeitung(-),zeit(-)):[suche,einen,Artikel],start:[ich]]

which is 0.3333. Now equation 22

$$ProbF = (0.5 * 0.3333 + 0.5 * 0.5) = 0.4167 \quad (44)$$

Using equation 16 to compute \mathcal{P} , we get:

$$\begin{aligned} &= (0.8 * (.5125+.2583+.6+.5042+.4167+.6389)+.2*.4167)*100 \\ &= (0.5*0.0107 + 0.2*0.0833) * 100 \\ &= (0.0919) * 100 \\ &= 9.19 \end{aligned}$$

the rating of this parse.

In the final stage, the parser uses equation 15 to collect the best parse P (and possibly discard those with an invalid semantics or with unbound variables) and simply puts back the meaning for *topic(1)* collected during the *topic_extraction* phase and the final parse is, without contextual information:

```
suche([(Condit)],zeitung(-),zeit(-))
```

7 Semantic Lexicon Acquisition

Before concluding, we wish to introduce briefly an area of research which could prove very useful, if not essential, for semantic parser construction. *Automated Semantic Lexicon Acquisition* ([18]) is the process by which semantic lexicons, to be used as background knowledge by a semantic parser, are learned from the same corpus of sentence/representation pairs used for the training of the parser as such. Let's look at an example¹⁵. Suppose we have the following pair sentence/representation in a corpus:

```
What is the longest river that does not run through Texas?  
answer(B,longest(B,(river(B),not(traverse(B,A),eq(A,stateid(texas)))))).
```

Uppercase characters represent variables. Then, an automated semantic lexicon acquisition could lead to the following entries in the lexicon:

```
[longest,longest(-,-)].  
[river,river(-)].  
[through,traverse(-,-)].  
etc...
```

It has therefore been learned that the word *through* triggers a two-place predicate (or concept) *traverse(-,-)*. This can be compared with the hand-crafted semantic lexicon in APPENDIX that we built for our domain. Clearly, this would ease greatly the construction of natural language interfaces. WOLFIE (see [18]) is an automated semantic lexicon learner which has been successfully implemented for Database-Query parsing.

8 Conclusion

The task of semantic parsing for natural language processing is essential if one wants to build natural language interfaces to access information from

¹⁵Reported in [18].

a computer or communicate with it. Traditional approaches have focused on hand-crafted methods to build such parsers, which have proven time-consuming and ineffective. Corpus-based methods offer a more effective way to deal with real data, and statistics offer an efficient and robust way to model and implement methods on a computer.

In this paper, a new probabilistic framework for semantic parsing is presented. The combination of a *shift-reduce* parser and a purely statistic model makes it unique. More precisely, the parser learns efficient ways of parsing new sentences by collecting statistics on the context in which each parsing action takes place. It computes probabilities on the basis of the similarities of those contexts and their frequencies. The result is a simple and robust parser. Its configuration can be change in many ways, to fit different types of corpus or domains. At this point, the system has not yet been fully tested on very large corpora, to see if the statistic model remains as efficient. It does not include the treatment of variables, which means that there is no treatment of questions¹⁶.

Compare to similar systems using some machine-learning techniques, ours offers an approach in which linguistics can play a decisive role. One crucial aspect of the parser, the computation of similarities between context, relies on a good interpretation of linguistic patterns found in phrases, and how syntax may determine the particular meaning of a word. This is essential to interpret, and maybe *understand*, natural language utterances.

References

- [1] J.F. Allen. *Natural Language Understanding*. Benjamin-Cummings, 1995.
- [2] S. Armstrong-Warwick. Preface (to the special issue on using large corpora), 19 1993.
- [3] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language*. Computational Linguistics at the University of the Saarland, 1997.
- [4] B. Carpenter C.D. Manning. Three generative, lexicalised models for statistical parsing. *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 147–158, 1997.
- [5] E. Charniak. Tree-bank grammars. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036, 1996.

¹⁶Such as *Wer ist der Präsident von Frankreich?*

- [6] Michael Collins and Scott Miller. Semantic tagging using a probabilistic context free grammar. In *Proceedings of the Sixth Workshop on Very Large Corpora*, 1998.
- [7] M.J. Collins. Three generative, lexicalised models for statistical parsing. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, 1997.
- [8] Michael A. Covington. *Natural Language Processing for Prolog Programmers*. Prentice-Hall, 1994.
- [9] Raymond J. Mooney Cynthia A. Thompson and Lappoon R.Tang. Learning to parse natural language database queries into logical form. *Proceedings of the ML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*, 1997.
- [10] C. Jacobson E. Charniak, C. Hendrickson and M. Perkowitz. Equations for part-of-speech tagging. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789, 1993.
- [11] Y. Schabes F. Pereira. Inside-outside reestimation from partially bracketed corpora. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, 1992.
- [12] D.J. Litman. Cue phrase classification using machine learning. *Journal of Artificial Intelligence Research*, 5:53–95, 1996.
- [13] R. Mercer L.R. Bahl, F. Jelinek. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 179–190, 1983.
- [14] Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. A fully statistical approach to natural language interfaces. *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 55–61, 1996.
- [15] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [16] A. Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–176, 1996.
- [17] Lappoon R. Tang and Raymond J. Mooney. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141, 2000.

- [18] Cynthia Ann Thompson. Semantic lexicon acquisition for learning natural language interfaces. *PhD Dissertation-University of Texas*, December 1998.

APPENDIX - A Lexicon, a Training File and some Parsing Results for the Newspaper Domain

```
/******  
    LEXICON:      lexicon.pl  
    This is a typical lexicon for the newspaper domain.  
    Four types of entry can be found in this lexicon:  
    1. Introduction of concepts:  
        lexicon(concept,[triggering_phrase]).  
    2. Substantives which do not participate to meaning as topic:  
        empty_word(Word).  
    3. Expressions that could be used as concept or topic:  
        mid_empty([expression]).  
    4. Define what kind of arguments we have:  
        argument(TYPE,LEXICON)  
*****/  
%% ARGUMENTS  
argument([],[topic(_)]).      %% Associate [] with topic(_)  
argument(zeit,[zeit(_)]).    %% Associate zeit with zeit(_)  
argument(zeitungen,[zeitungen(_)]).  %% Associate zeitungen with zeitungen(_)  
  
%% TOPICS  
lexicon(topic(1),[topic(1)]).  %% topic(N) is a placeholder for a topic  
lexicon(topic(2),[topic(2)]).  
lexicon(topic(3),[topic(3)]).  
lexicon(topic(4),[topic(4)]).  
lexicon(topic(5),[topic(5)]).  
lexicon(topic(6),[topic(6)]).  
lexicon(topic(7),[topic(7)]).  
lexicon(topic(8),[topic(8)]).  
lexicon(topic(9),[topic(9)]).  
  
%% NEUE SUCHE  
lexicon(neue_suche,[neue, ' Suche ']).  
lexicon(neue_suche,[mit,der, ' Suche ' ,wird,begonnen]).  
  
%% SUCHE  
lexicon(suche([],zeitung(_),zeit(_)),[' Suchauftrag ']).  %% Les listes [ ] n'acceptent que topic(_).  
lexicon(suche([],zeitung(_),zeit(_)),[suche]).  
lexicon(suche([],zeitung(_),zeit(_)),[besonders]).  
lexicon(suche([],zeitung(_),zeit(_)),[ich,brauche]).  %% zeitung(_ ) is a placeholder for a Zeitung  
lexicon(suche([],zeitung(_),zeit(_)),[' Artikel ']).  %% zeit(_ ) is a placeholder for a Zeit Ausdruck  
lexicon(suche([],zeitung(_),zeit(_)),[' Genaueres ']).  
lexicon(suche([],zeitung(_),zeit(_)),[' Berichte ']).  
lexicon(suche([],zeitung(_),zeit(_)),[speziell]).
```

```

lexicon(suche([],zeitung(_),zeit(_),[insbesondere])).
lexicon(suche([],zeitung(_),zeit(_),['Info'])).
lexicon(suche([],zeitung(_),zeit(_),['Infos'])).
lexicon(suche([],zeitung(_),zeit(_),['Informationen'])).
lexicon(suche([],zeitung(_),zeit(_),['Thema'])).
lexicon(suche([],zeitung(_),zeit(_),[betrifft])).
lexicon(suche([],zeitung(_),zeit(_),[es,geht,um])).
lexicon(suche([],zeitung(_),zeit(_),[und,zwar,über])).
lexicon(suche([],zeitung(_),zeit(_),[anzeigen])).
lexicon(suche([],zeitung(_),zeit(_),[berücksichtigen])).
lexicon(suche([],zeitung(_),zeit(_),[ergänzen])).
lexicon(suche([],zeitung(_),zeit(_),[in,'Verbindung',mit])).
lexicon(suche([],zeitung(_),zeit(_),['Seite',über])).
lexicon(suche([],zeitung(_),zeit(_),[alles,über])).
lexicon(suche([topic(N)],zeitung(_),zeit(_),[topic(N)]). % A topic alone can trigger a search.

```

%% ZEIT

```

lexicon(zeit(tag(-1)),[von,gestern]).
lexicon(zeit(woche(-1)),[von,letzter,'Woche']).
lexicon(zeit(monat(-1)),[von,letztem,'Monat']).
lexicon(zeit(jahr(-1)),[von,letztem,'Jahr']).
lexicon(zeit(woche(-2)),[von,vor,einer,'Woche']).
lexicon(zeit(monat(-2)),[von,vor,einem,'Monat']).
lexicon(zeit(jahr(-2)),[von,vor,einem,'Jahr']).
lexicon(zeit(woche(-1)),[letzte,'Woche']).
lexicon(zeit(monat(-1)),[letztes,'Monat']).
lexicon(zeit(monat(-1)),[letzten,'Monat']).
lexicon(zeit(jahr(-1)),[letztes,'Jahr']).

```

%% ZEITUNGEN

```

lexicon(zeitung(presse),[in,der,'Presse']).
lexicon(zeitung(standard),[im,'Standard']).
lexicon(zeitung(salzbürger_nachrichten),[in,den,'Salzburger','Nachrichten']).
lexicon(zeitung(tiroler_tageszeitung),[in,der,'Tiroler','Tageszeitung']).
lexicon(zeitung(presse),[der,'Presse']).
lexicon(zeitung(salzbürger_nachrichten),[der,'Salzburger','Nachrichten']).
lexicon(zeitung(tiroler_tageszeitung),[der,'Tiroler','Tageszeitung']).
lexicon(zeitung(presse),[der,'Zeitung','Die','Presse']).
lexicon(zeitung(standard),[der,'Zeitung','Der','Standard']).
lexicon(zeitung(salzbürger_nachrichten),[der,'Zeitung','Salzburger','Nachrichten']).
lexicon(zeitung(tiroler_tageszeitung),[der,'Zeitung','Tiroler','Tageszeitung']).

```

%% VOLLTEXT

```

lexicon(volltext,['Volltext',sehen]).

```

```

%% BESTÄTIGEN
lexicon(bestätigen(vvv(_)),[bestätigen]).
lexicon(bestätigen(vvv(_)),['Eingabe']).
lexicon(bestätigen(vvv(_)),[bearbeiten, 'Sie']).

```

%% vvv(N) is a placeholder for anything
%% it should not appear in the final interpretation

```

%% COMMANDS
lexicon(command(vorwärts),[vor]).
lexicon(command(zurück),[zurück]).
lexicon(command(vorwärts),[vorwärts]).
lexicon(command(zurück),[rückwärts]).
lexicon(command(vorwärts),[oben]).
lexicon(command(zurück),[unten]).
lexicon(command(vorwärts),[nach,oben]).
lexicon(command(zurück),[nach,unten]).
lexicon(command(vorwärts),[weiter]).
lexicon(command(zurück),[vorher]).

```

100

```

%BISHERIG
lexicon(search_zurück(topic(_)),[zurück]).
lexicon(search_zurück(topic(_)),[zurück,zu,der, 'Seite']).
lexicon(search_zurück(topic(_)),[zurück,zu,der, 'Seite']).
lexicon(search_zurück(topic(_)),[vor]).
lexicon(search_zurück(topic(_)),[vor,zu,der, 'Seite']).
lexicon(search_vorwärts(topic(_)),[vor,zu,der, 'Seite']).
lexicon(search_vorwärts(topic(_)),[weiter]).
lexicon(search_vorwärts(topic(_)),[weiter,zum, 'Thema']).

```

```

%% RESSORT
lexicon(section('Sport'),['Sport']).
lexicon(section('Politik'),['Politik']).
lexicon(section('Wirtschaft'),['Wirtschaft']).
lexicon(section('Kultur'),['Kultur']).
lexicon(section('Sport'),[zurück,zum, 'Sportressort']).
lexicon(section('Sport'),[zurück,zum, 'Sportteil']).
lexicon(section('Sport'),[zurück,zu,den, 'Sportseiten']).
lexicon(section('Sport'),[zurück,zu,den, 'Sportmeldungen']).
lexicon(section('Politik'),[zurück,zur, 'Politik']).

```

%% _____

% Empty words. These are simply words which cannot be topics.

```

empty_word('Artikel').
empty_word('Artikeln').
empty_word('Bericht').

```

```

empty_word('Berichte' ).
empty_word('Bestätigung' ).
empty_word('Bezug' ).
empty_word('Blödsinn' ).
empty_word('Charakterisierung' ).
empty_word('Details' ).
empty_word('Eingabe' ).
empty_word('Frage' ).
empty_word('Genaueres' ).
empty_word('Info' ).
empty_word('Information' ).
empty_word('Informationen' ).
empty_word('Infos' ).
empty_word('Suchauftrag' ).
empty_word('Suche' ).
empty_word('Tageszeitung' ).
empty_word('Tageszeitungen' ).
empty_word('Text' ).
empty_word('Texte' ).
empty_word('Texten' ).
empty_word('Thema' ).
empty_word('Themen' ).
empty_word('Zusammenhang' ).
empty_word('Verbindung' ).
empty_word('Volltext' ).
empty_word('Seite' ).
empty_word(X,_):-
    member(X,['Woche','Monat','Jahr']).
empty_word(X,_):-
    member(X,['Sport','Politik','Wirtschaft','Kultur','Sportressort',
              'Sportteil','Sportseiten','Sportmeldungen']).

%% These will also appear as topics or in the lexicon (i.e. participate to the meaning)
mid_empty([in,der,'Presse']).
mid_empty([im,'Standard']).
mid_empty([in,den,'Salzburger','Nachrichten']).
mid_empty([in,der,'Tiroler','Tageszeitung']).
mid_empty([der,'Presse']).
mid_empty([der,'Salzburger','Nachrichten']).
mid_empty([der,'Tiroler','Tageszeitung']).
mid_empty([der,'Zeitung','Die','Presse']).
mid_empty([der,'Zeitung','Der','Standard']).
mid_empty([der,'Zeitung','Salzburger','Nachrichten']).
mid_empty([der,'Zeitung','Tiroler','Tageszeitung']).
/*****

```

TRAINING FILE: training.pl

This file has the format:

training([phrase],meaning).

Note that topics have already been extracted from the training examples,
so that for example, the first training example

could be used to train for a phrase like:

“Bush hält eine Rede über die Kosovo-Krise

bitte bearbeiten Sie meinen Suchauftrag.”

*****/

%% SEARCH and CONFIRMATION

```
training([topic(1),hält,topic(2),topic(3),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1),topic(2),topic(3)],zeitung(_),zeit(_)))).
training([ich,möchte,jetzt,eine,neue,'Suche',beginnen],neue_suche).
training([bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],bestätigen(suche([],zeitung(_),zeit(_)))).
training([ich,möchte,jetzt,eine,neue,'Suche',beginnen],neue_suche).
training([topic(1),will,topic(2),überzeugen,da,man,topic(3),moralisch,verpflichtet,war,
        bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],bestätigen(suche([topic(1),topic(2),topic(3)],
        zeitung(_),zeit(_)))).
training([wirtschaftliche,topic(1),führen,auch,bei,inner Österreichischen,topic(2),topic(3),
        bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],bestätigen(suche([topic(1),topic(2),topic(3)],
        zeitung(_),zeit(_)))).
training([topic(1),topic(2),ist,besonders,topic(3),interessant,zu,beobachten,200
        bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],bestätigen(suche([topic(1),topic(2),topic(3)],
        zeitung(_),zeit(_)))).
training([ich,möchte,jetzt,eine,neue,'Suche',beginnen,bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(neue_suche)).
training([topic(1),topic(2),topic(3),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1),topic(2),topic(3)],zeitung(_),zeit(_)))).
training([topic(1),topic(2),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1),topic(2)],zeitung(_),zeit(_)))).
training([ich,suche,'Texte',topic(1),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1)],zeitung(_),zeit(_)))).
training([topic(1),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1)],zeitung(_),zeit(_)))).
training([ich,suche,'Texte',topic(1),topic(2),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1),topic(2)],zeitung(_),zeit(_)))).
training([ich,möchte,jetzt,eine,neue,'Suche',beginnen],neue_suche).
training([bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],bestätigen(suche([],zeitung(_),zeit(_)))).
training([ich,suche,'Texte',topic(1),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1)],zeitung(_),zeit(_)))).
training([ich,suche,'Texte',topic(1),topic(2),topic(3),bitte,bearbeiten,'Sie',meinen,'Suchauftrag'],
        bestätigen(suche([topic(1),topic(2),topic(3)],zeitung(_),zeit(_)))).
training([ich,suche,einen,'Text',topic(1),topic(2),topic(3),topic(4),bitte,bearbeiten,'Sie',
        meinen,'Suchauftrag'],bestätigen(suche([topic(1),topic(2),topic(3),topic(4)],
```

```

        zeitung(_),zeit(_))).
training([ich,suche, 'Texte', topic(1),topic(2),die,für,österreichische,die,topic(3),interessant,sind,
        bitte,bearbeiten, 'Sie',meinen, 'Suchauftrag'],bestätigen(suche([topic(1),topic(2),topic(3)],
        zeitung(_),zeit(_))).
training([ich,suche, 'Texte', topic(1),topic(2),die,einen,rasanten,topic(3),topic(4),geführt,hat,
        bitte,bearbeiten, 'Sie',meinen, 'Suchauftrag'],bestätigen(suche([topic(1),topic(2),
        topic(3),topic(4)],zeitung(_),zeit(_))).
training([ich,suche,einen, 'Text',for,topic(1),topic(2),bitte,bearbeiten, 'Sie',meinen, 'Suchauftrag'],
        bestätigen(suche([topic(1),topic(2)],zeitung(_),zeit(_))).
training([ich,möchte,jetzt,eine,neue, 'Suche',beginnen,bitte,bearbeiten, 'Sie',meinen, 'Suchauftrag'],
        bestätigen(neue_suche)).
training([ich,suche, 'Texte', topic(1),topic(2),bitte,bearbeiten, 'Sie',meinen, 'Suchauftrag'],
        bestätigen(suche([topic(1),topic(2)],zeitung(_),zeit(_))).
training([ich,suche, 'Texte', topic(1),topic(2),gute,topic(3),haben,mitzuwirken,bitte,bearbeiten, 'Sie',
        meinen, 'Suchauftrag'],bestätigen(suche([topic(1),topic(2),topic(3)],zeitung(_),zeit(_))).
%% ZEIT
training([suche,topic(1),von,gestern],suche([topic(1)],zeitung(_),zeit(tag(-1)))).
%% ZEITUNGEN
training([suche,topic(1),im, 'Standard'],suche([topic(1)],zeitung(standard),zeit(_))).
%% ZEIT+ZEITUNGEN
training([suche,topic(1),im, 'Standard',von,gestern],suche([topic(1)],zeitung(standard),zeit(tag(-1)))).
%% VOLLTEXT
training([bitte, 'Volltext',sehen],volltext).
%% COMMANDS
training([weiter,bitte],command(vorwärts)).
%% BISHERIC
training([bitte,vor,zu,der, 'Seite',topic(1)],search_vorwärts(topic(1))).
%% RESSORT
training([bitte,sehr,zurück,zu,den, 'Sportmeldungen'],section('Sport')).
%% ZEITUNG as TOPICS
training([ich,suche,im, 'Standard',einen, 'Artikel',topic(1)],suche([topic(1)],zeitung(standard),zeit(_))).
/*****
        R E S U L T S   o f   S E M A N T I C   P A R S I N G   u s i n g
        =====
        LEXICON:      lexicon.pl
        TRAINING FILE: training.pl
        TRAINING BEAM: 3
        SEARCH BEAM: 20
        and all default parameters mentioned in the paper.
*****/
>>>BEGIN of the PARSE with a beam of 20 for:
[ich,möchte,eine,neue,Suche,beginnen,bitte,bearbeiten,Sie,meinen,Suchauftrag]
No TOPICS found
**** THE BEST INTERPRETATION ****

```

for [ich,möchte,eine,neue,Suche,beginnen,bitte,bearbeiten,Sie,meinen,Suchauftrag]
is bestätigen(neue_suche)
with a rating of 11.190367004310986

>>>BEGIN of the PARSE with a beam of 20 for:
[ich,suche,Texte,zur,zum,Stand,der,Wissenschaft,in,österreich,bitte,bearbeiten,Sie,meinen,Suchauftrag]
--TOPICS found-----
[zur,zum,Stand]((zum+Stand),(Stand))
[der,Wissenschaft](Wissenschaft)
[in,österreich](österreich)

After topicalization:[ich,suche,Texte,topic(1),topic(2),topic(3),bitte,bearbeiten,Sie,meinen,Suchauftrag]
**** THE BEST INTERPRETATION ****
for [ich,suche,Texte,zur,zum,Stand,der,Wissenschaft,in,österreich,bitte,bearbeiten,Sie,meinen,Suchauftrag]
is bestätigen(suche([(Wissenschaft),(österreich)],(zum+Stand),(Stand)]),zeitung(-),zeit(-))
with a rating of 9.102599774196166

>>>BEGIN of the PARSE with a beam of 20 for:
[ich,suche,Texte,zur,dürftigen,Rahmenbedingungen,der,österreichischen,Forschung,bitte,bearbeiten,Sie,meinen,
Suchauftrag]
--TOPICS found-----
[zur,dürftigen,Rahmenbedingungen]((dürftigen+Rahmenbedingungen),(Rahmenbedingungen))
[der,österreichischen,Forschung]((österreichischen+Forschung),(Forschung))

After topicalization:[ich,suche,Texte,topic(1),topic(2),bitte,bearbeiten,Sie,meinen,Suchauftrag]
**** THE BEST INTERPRETATION ****
for [ich,suche,Texte,zur,dürftigen,Rahmenbedingungen,der,österreichischen,Forschung,bitte,bearbeiten,Sie,
meinen,Suchauftrag]
is bestätigen(suche((((dürftigen+Rahmenbedingungen),(Rahmenbedingungen)),((österreichischen+Forschung),
(Forschung))),zeitung(-),zeit(-))
with a rating of 9.10327909136617

>>>BEGIN of the PARSE with a beam of 20 for: 300
[ich,möchte,jetzt,eine,neue,Suche,beginnen,bitte,bearbeiten,Sie,meinen,Suchauftrag]
No TOPICS found
**** THE BEST INTERPRETATION ****
for [ich,möchte,jetzt,eine,neue,Suche,beginnen,bitte,bearbeiten,Sie,meinen,Suchauftrag]
is bestätigen(neue_suche)
with a rating of 11.85412044326708

>>>BEGIN of the PARSE with a beam of 20 for:
[ich,suche,Texte,zum,österreichischen,Fuballsport,bitte,bearbeiten,Sie,meine,Eingabe]
--TOPICS found-----
[zum,österreichischen,Fuballsport]((österreichischen+Fuballsport),(Fuballsport))

After topicalization:[ich,suche,Texte,topic(1),bitte,bearbeiten,Sie,meine,Eingabe]
**** THE BEST INTERPRETATION ****
for [ich,suche,Texte,zum,österreichischen,Fuballsport,bitte,bearbeiten,Sie,meine,Eingabe]
is bestätigen(bestätigen(suche(((österreichischen+Fuballsport),(Fuballsport))),zeitung(_),zeit(_)))
with a rating of 6.367522179209511

>>>BEGIN of the PARSE with a beam of 20 for:

[ich,suche,etwas,über,Sharon,von,gestern]
--TOPICS found-----
[etwas,über,Sharon]((über+Sharon),(Sharon))

After topicalization:[ich,suche,topic(1),von,gestern]
**** THE BEST INTERPRETATION ****
for [ich,suche,etwas,über,Sharon,von,gestern]
is suche(((über+Sharon),(Sharon))),zeitung(_),zeit(tag(-1)))
with a rating of 8.465286679097996

>>>BEGIN of the PARSE with a beam of 20 for:

[suche,Bush,in,der,Presse,bitte]
--TOPICS found-----
Bush
[in,der,Presse](Presse)

After topicalization:[suche,topic(1),in,der,Presse,bitte]After topicalization:[suche,topic(1),topic(2),bitte]
**** THE BEST INTERPRETATION ****
for [suche,Bush,in,der,Presse,bitte]
is suche([(Bush)],zeitung(presse),zeit(_))
with a rating of 8.718886520139755

>>>BEGIN of the PARSE with a beam of 20 for:

[bitte,suche,öFAI,im,Standard,von,gestern]
--TOPICS found-----
öFAI
[im,Standard](Standard)

After topicalization:[bitte,suche,topic(1),im,Standard,von,gestern]After topicalization:[bitte,suche,topic(1),
topic(2),von,gestern]
**** THE BEST INTERPRETATION ****
for [bitte,suche,öFAI,im,Standard,von,gestern]
is suche([(öFAI)],zeitung(standard),zeit(tag(-1)))
with a rating of 7.0513693137456785

>>>BEGIN of the PARSE with a beam of 20 for:

[bitte,Volltext,sehen,jetzt]
No TOPICS found

**** THE BEST INTERPRETATION ****
for [bitte,Volltext,sehen,jetzt]
is volltext
with a rating of 14.678469678469677

>>>BEGIN of the PARSE with a beam of 20 for:
[weiter,bitte,sehr]
No TOPICS found

**** THE BEST INTERPRETATION ****
for [weiter,bitte,sehr]
is command(vorwärts)
with a rating of 18.743725410392077

>>>BEGIN of the PARSE with a beam of 20 for:
[bitte,vor,zu,der,Seite,mit,Palestine]
--TOPICS found-----
[mit,Palestine](Palestine)

After topicalization:[bitte,vor,zu,der,Seite,topic(1)]
**** THE BEST INTERPRETATION ****
for [bitte,vor,zu,der,Seite,mit,Palestine]
is search_vorwärts((Palestine))
with a rating of 50.76312576312576

>>>BEGIN of the PARSE with a beam of 20 for:
[bitte,sehr,zurück,zum,Sportteil]
No TOPICS found
**** THE BEST INTERPRETATION ****
for [bitte,sehr,zurück,zum,Sportteil]
is section(Sport)
with a rating of 17.48931623931624

>>>BEGIN of the PARSE with a beam of 20 for:
[ich,suche,im,Standard,einen,Artikel,über,die,Presse]
--TOPICS found-----
[im,Standard](Standard)
[über,die,Presse](Presse)

After topicalization:[ich,suche,im,Standard,einen,Artikel,topic(1)]After topicalization:[ich,suche,topic(1),
einen,Artikel,topic(2)]
**** THE BEST INTERPRETATION ****
for [ich,suche,im,Standard,einen,Artikel,über,die,Presse]
is suche([(Presse)],zeitung(standard),zeit(_))
with a rating of 8.463822439575516

400