Stochastic Parsing in a Hybrid Semantic Analysis System

Miloslav Konopík and Ivan Habernal

Abstract— **This article is focused on the problem of meaning recognition in spoken utterances. The goal is to find a computer algorithm capable to construct the meaning description of a given sentence. Such an algorithm is used in applications that require human computer interaction in a natural language. In this article we describe some experiments that we made in this area of computation linguistic research.**

I. INTRODUCTION

Recent achievements in the area of automatic speech recognition started the development of speech enabled applications. Currently it starts to be insufficient to merely recognize an utterance. The applications demand to understand the meaning of the utterance. Semantic analysis is a process whereby the computer representation of the sentence meaning is automatically assigned to an analyzed sentence.

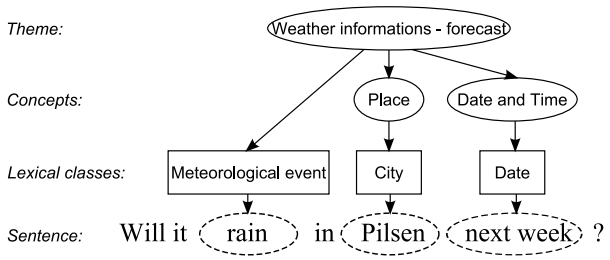

Fig. 1. The example of an annotation tree.

There are several ways how to represent semantic information contained in a sentence. In our work we use tree structures (see Figure 1) with so called *concepts* and *lexical classes*. The theme of the sentence is placed on the top of the tree. The inner nodes are called concepts. Concepts describe some portion of semantic information contained in the sentence. They can contain other sub-concepts that specify the semantic information more precisely or they can contain so called lexical classes. Lexical classes are the leaves of the tree. A lexical class cover certain phrases that contains the same type of information. For example a lexical class "date" covers phrases "tomorrow", "Monday", "next week" or "25th December" etc.

In our approach to semantic analysis we divide the process of semantic analysis into two parts: the process of lexical classes identification and the process of semantic parsing (creating the tree structure above the lexical classes). In this article we describe the second part, the parsing. Thus, from now on it will be assumed that the lexical classes are known for each sentence. The process of lexical class identification in our system is described in [1].

In our system the lexical classes are created by dedicated parsers mostly based on expert knowledge. To the contrary the subsequent step - semantic parsing - is based on stochastic training from data. Since we use two different approaches to semantic analysis together we refer to our approach as a hybrid method.

Machine learning algorithms (e.g. stochastic training) require training data for learning. The training data serve as examples for the learning algorithm. The algorithm should be capable to generalize and to extract relations from the examples (not only store the examples). We use a so called *supervised training*. That means we give the algorithm the example sentences with correct results (in our case with the correct semantic description). The process of assigning the correct semantic description to examples is called *semantic annotation* and the examples are called *annotated sentences*. People who create these sentences are called *annotators*.

The semantic representation formalism used in this thesis is taken from [2]. In this paper the formalism is called the *abstract semantic annotations*. The advantage of this approach is that it does not require annotating of all words of a sentence. The way of stochastic semantic parsing is also inspired by [3] and [4]. We extend these approaches with several modifications so that the algorithms can work with free word order languages (e.g. Czech language).

In this article we describe several approaches to semantic parsing that use different ways of training and different amount of context. The algorithms are explained and the results are given and discussed.

II. DATA AND DATA ANNOTATION

We are focused on the domain of internet queries. The testing and training sentences are questions in the natural language (Czech) that can be put into an internet search engine (similar to Google). The search engine equipped with our semantic analysis system is then able to process queries in natural language and to produce adequate answers. For example a user can ask a weather related question - "What will be the weather in Pilsen tomorrow". Then the search engine semantically analyzes the question and returns directly the picture of the tomorrow weather forecast in Pilsen. An ordinary search engine usually returns a list of pages. On some of them the user can find with some effort (fill a form or select a city form a combo box etc) the requested weather forecast. Thus the engine with semantic analysis capability

M. Konopík is with University of West Bohemia, Department of Computer Sciences, Univerzitní 22, CZ - 306 14 Plzeň, Czech Republic konopik@kiv.zcu.cz

I. Habernal is with University of West Bohemia, Department of Computer Sciences, Univerzitní 22, CZ - 306 14 Plzeň, Czech Republic habernal@kiv.zcu.cz

is able to improve user experience and significantly lower the time needed to find the answer. However the methods described in this article can be used for variety of tasks (voice driven dialog system, text analysis etc) only examples and results from internet domain are given here.

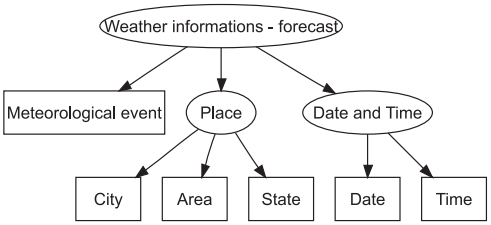

Fig. 2. The annotation scheme for weather forecast.

During the semantic annotation process (data creation process) we designed a structure we call the annotation scheme (see Figure 2). One scheme was created for each covered topic (weather, accommodation, public transport, ...). The annotation scheme clearly states how the annotations can look like. An annotation of a sentence is always a "subset"[1] of the annotation scheme.

During our research 20292 sentences were obtained and filtered. Unfortunately more than 70% of the collected sentences were not usable for further processing. Then we assembled a team of annotators who by hand assigned the semantic representation to each collected sentence. We have so far processed 1 344 sentences from 3 domains (weather forecast, accommodation, public transport information).

III. SEMANTIC PARSING

The goal of semantic analysis is to build a tree that describes the semantics in a given sentence. At first it has to localize the lexical classes and then create the tree. In this article we expect that the lexical classes are found and we have to create the tree. Now we describe several algorithms that we designed for that purpose.

A. Parsing according to the annotation scheme

This parser is the simplest one of all parser described here. It does not need any training data. It just requires loading the annotation schemes. The input of the algorithm are sentences with identified lexical classes. Since the resulting semantic annotation of a sentence always has to be a "subset" of the annotation scheme the algorithm simply tries to create all "subset" trees of the loaded schemes. Only trees that contain all lexical classes of the analyzed sentence are accepted as results. If more than one scheme can be applied then multiple results are returned. In that way we basically get all trees that are acceptable for a given sentence according to all annotation schemes provided. In some cases there exists just one tree for the given sentence because the sentence contains a specific lexical class or a specific combination of lexical classes that occurs only in one scheme. Then this one tree is returned as a result without any trouble. In the case of multiple trees one of them is randomly chosen as the result (there is no clue to guide the selection).

The algorithm is a backtracking bottom-up parser. It takes one lexical class at a time a tries to connect it to some superior concept. Then the superior concept is added to agenda and later it is tried to append it above another concept and so on. When a root of some scheme is reached then the algorithm has found one result. If the concept cannot be connected to any concept then this hypothesis is rejected and the concept is discarded.

B. Chart parsing

This parser works in two modes: training and analysis. The training phase requires aforementioned annotated data. During the training the annotation trees are transformed to context free grammar rules in the following way. Every node is transformed to one rule. The node name makes the left side of the rule and the children of the node make the right sides of the rule (for example see node "Place" in Figure 1, this node is transformed into the rule `Place -> City`). In this way all the nodes of the annotation tree are processed and transformed into grammar rules. Naturally, an identical rule can be created more than once. In that case all duplicate rules are discarded.

The analysis phase is in no way different from standard context-free parsing. The sentence is passed to the parsing algorithm. In our research we use the active chart parsing algorithm (see e.g. [5]) that runs in polynomial time. The lexical classes identified in the sentence are treated as terminal symbols and passed to the parsing algorithm. The words that are not members of any lexical class are ignored. The result of parsing - the parse tree - is directly in the form of the result tree we need.

One problem of this algorithm is the ambiguity. The grammars created by the aforementioned way are usually strongly ambiguous. Ambiguous grammars cause that more than one parse tree is created during parsing. When more than one parse tree is returned it is unclear which one should be treated as the result. Since we have no clue in this type of parser we select randomly one that is returned as the result.

The ambiguity in grammars also introduces problems to the parsing algorithm. It is necessary to explore all possibilities during parsing because otherwise a correct result can be overlayed by an incorrect ambiguous alternative. Thus it is necessary to modify the original chart parsing algorithm to explore all possible ambiguous ways of parsing a substring.

C. Stochastic Chart Parsing

In this parser we introduce probabilities. The probabilities help us to deal with the ambiguity. A probability of a rule transcription is assigned to each rule. The probabilities are computed from the training data. Using stochastic[2] parsing the resulting parse trees are returned with the overall probability of the parse. The parse tree with the highest

[1]by subset we mean that the annotation tree does not have to contain all branches from the scheme tree

[2]stochastic = using probability

probability is then the most likely to be the correct result. Now, we will explain the process of parsing, the training formulas and modification to the parsing algorithm to deal with probabilities more formally.

First, we start with training. In the previous section III-B it was explained how the annotations trees are transformed to context-free grammar rules. It was said that when a duplicate rule is found then it is discarded. Now, the rule is not simply discarded but instead it is counted how many times the rule occurred in the training data. Then we can estimate the conditional probability of a rule transcription given the left side of the rule using the MLE[3]:

$$P(N \rightarrow \alpha|N) = \frac{\text{Count}(N \rightarrow \alpha)}{\sum_{\gamma} \text{Count}(N \rightarrow \gamma)} \tag{1}$$

To deal with the probabilities the chart parser from previous section is modified in the following way. During parsing the probability of the so far created tree $P(T)$ is computed by:

$$P(T) = P(N \rightarrow A_1 A_2 ... A_k|N) \prod_i P(T_i) \tag{2}$$

where N is the top nonterminal of the subtree T, A_i are the terminals or non-terminals to which the N is being transcribed and T_i is the subtree having the A_i nonterminal on the top.

When the parsing is finished a probability is assigned to all resulting parse trees. The probability is then weighted by the prior probability of the theme and the maximum probability is chosen as the result:

$$\hat{T} = \arg\max_i P(S_i) P(T_i) \tag{3}$$

where $\hat{T}$ is the most likely parse tree and $P(S_i)$ is the probability of the starting symbol of the parse tree T_i.

The advantage of this parser is that it assigns probabilities to resulting parse trees. In the case of multiple outputs the most probable parse tree is selected as the result. However, this parser accounts for the lexical classes only and other words of the sentence are still ignored.

D. Word Context Parsing

Now we start looking at other words of sentences rather than looking on lexical classes only. For this purpose we need to extend both the training algorithm and the analysis algorithm.

The training phase shares the same steps with the training of the previous parser in section III-C. So, the node is transformed to the grammar rule and the frequency of rule occurrence is counted. However instead of going to the next node the context of the node is examined. Every node that is not a leaf has a subtree beneath. The subtree spans across some nonterminals. The context of the node is defined as the words before and after the span of the subtree (see figure 3). During training the frequency of the context and

[3]Maximum Likelihood Estimate

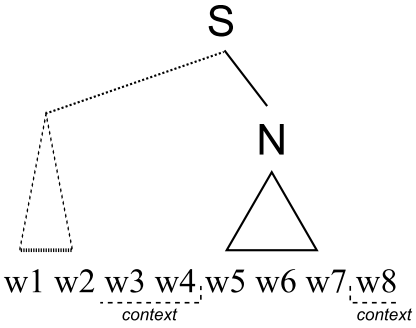

Fig. 3. Illustration of a context of a subtree.

a nonterminal (Count$(word, nonterminal)$) is counted. The probability of a context given a nonterminal is computed via MLE as follows:

$$P(w|N) = \frac{\text{Count}(w, N) + \lambda}{\sum_i \text{Count}(w_i, N) + \lambda V} \tag{4}$$

where λ is the smoothing constant, V is the estimate of the vocabulary size, w is the actual context word and w_i are all the context words of nonterminal N.

Additionally, to improve the estimate of the prior probability of the theme (the root node of the annotation) we add words to the estimate as well:

$$P(w|S) = \frac{\text{Count}(w, S) + \kappa}{\sum_i \text{Count}(w_i, S) + \kappa V} \tag{5}$$

where κ is the smoothing constant, w_i are the words of the sentence and S is the theme of the sentence (the theme constitutes the starting symbol after annotation tree transformation).

The analysis algorithm is the same as in the previous parser but the probability from formula 2 is reformulated to consider the context:

$$P(T) = \sum_i P(w_i|N) P(N \rightarrow A_1 A_2 ... A_k|N) \prod_j P(T_j) \tag{6}$$

Then the best parse is selected using context sensitive prior probability:

$$P(\hat{T}) = \arg\max_i P(S_i) \prod_j (P(w_j|S) P(T_i) \tag{7}$$

where S_i is the starting symbol of the parse tree T_i and w_j are the words of the analyzed sentence.

This parser provides a reasonable performance (see section IV) however there is still some unused potential. The conditional probabilities are computed in a word order dependent way. There is also the independence assumption of stochastic context free grammars (see [6], section 12.2) that may be too strong. Stochastic models can be tuned and so on. For further discussion see section V-B.

IV. RESULTS

This section describes the performance of the parsers introduced above. It was tested on 9 annotation schemes from 3 themes (weather forecast, accommodation, public transport

TABLE I. The parser performance

Parser	Training data	Cross validation
Annotation scheme	0.37	0.37
Chart parsing	0.62	0.59
Stochastic Chart Parser	0.64	0.62
Word Context Parser	0.87	0.77

information). The total number of tested sentences was 1344. The performance is given by the accuracy measure. The accuracy is computed as the ratio between the number of correct results (C) and the number of tested sentences (T): $\frac{C}{T}$. The result is considered to be correct only if the result is identical to the semantic annotation contained in the training/testing data.

There are two results given. First all sentences were used for both training and testing. Second results are computed by the *cross validation* technique. The pool of annotated sentences is split to the training and testing set and the resulting performance is computed. Then the pool is split again but in the different way so that that the new training set does not share any sentences that were contained in the previous set and again the results are computed. The process of splitting and performance computation is continued until all sentences are used for testing. Afterward the obtained results are averaged. In this way the complete data can be used for both training and testing without violating the principle that data used for training must not be used for testing. In our case 10% of sentences was used for testing and 90% sentences for training.

A. Results discussion

The first parser (Parsing according to the annotation scheme) does not use training data at all and yet it reached accuracy of 37%. For this result it can be concluded that the identified lexical classes are strong information source.

The second parser (The Chart Parser) introduced training from data. Results show a big jump in the performance caused by the usage of the data.

The improvement brought by introducing statistics to parsing (Stochastic Chart Parsing) is not so significant. The reasons why the use of statistics did not meet our expectation in performance could be only guessed. We think that due to the independence assumption the probability estimates are not predictive enough.

The last parser (Word Context Parsing) uses the context to improve the stochastic prediction. It can be concluded that the context is another strong information source for semantic parsing.

The price for the increased performance of the Word Context Parser is a slightly more complicated parsing algorithm but it still has the same computational complexity as other parsers - $O(n^3)$. Since it uses all words of training data the size of the model is proportional to the size of the training data. In the case of large training data the trained model can be larger and a pruning algorithm should be used.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this article we introduced several methods for semantic parsing. The article shows the influence of training data and context to semantic parsing. The most advanced parser provides very promising results. However it is necessary to stress that the provided results assume perfect (100 % accuracy) recognition of lexical classes. In reality it is not the case and the results with imperfectly recognized lexical classes are lower.

B. Future Works

Many improvements could be done. We expect some interesting results from the implementation of the HVS parser [2].

There is a lot of opportunity in tuning the parsers. Particularly different methods of smoothing the probability estimates can be used. Other information sources can be introduced. We will try to condition the probabilities to alleviate the independence assumption of stochastic grammars.

Obtaining more data is of course always necessary and it is the never ending process.

VI. ACKNOWLEDGMENTS

This work was supported by grant no. 2C06009 Cot-Sewing.

REFERENCES

[1] I. Habernal, "Lexical class analysis," M.S. thesis, University of West Bohemia, 2007.

[2] H. Yulan and S. Young, "Semantic processing using the hidden vector state model," *Computer speech & language*, 19(1):85–106, 2005.

[3] S. Young, "The statistical approach to the design of spoken dialogue systems," Tech. Rep. CUED/F-INFENG/TR.433, Cambridge University Engineering Department, 2002.

[4] S. Miller, D. Stallard, R. Bobrow, and R. Schwartz, "A fully statistical approach to natural language interfaces.," in *In Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, 1996.

[5] J. Allen, *Natural Language Understanding*, Benjamins / Cummins Publishing Co., Inc., Redwood City, CA, USA, second edition, 1995.

[6] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.