Hybrid Semantic Analysis System – ATIS Data Evaluation

Ivan Habernal and Miloslav Konopík

University of West Bohemia, Department of Computer Sciences,
Univerzitní 22, CZ - 306 14 Plzeň, Czech Republic
{habernal,konopik}@kiv.zcu.cz

Abstract. In this article we show a novel method of semantic parsing. The method deals with two main issues. First, it is developed to be reliable and easy to use. It uses a simple tree-based semantic annotation and it learns from data. Second, it is designed to be used in practical applications by incorporating a method for data formalization into the system. The system uses a novel parser that extends a general probabilistic context-free parser by using context for better probability estimation. The semantic parser was originally developed for Czech data and for written questions. In this article we show an evaluation of the method on a very different domain – ATIS corpus. The achieved results are very encouraging considering the difficulties connected with the ATIS corpus.

Keywords: semantic analysis, semantic parsing, spoken language understanding, ATIS

1 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. Semantic analysis is a process whereby the computer representation of the sentence meaning is automatically assigned to an analyzed sentence.

Our approach to semantic analysis is based upon a combination of expert methods and stochastic methods (that is why we call our approach a hybrid semantic analysis). We show that a robust system for semantic analysis can be created in this way. During the development of the system an original algorithm for semantic parsing was created. The developed algorithm extends the chart parsing method and context-free grammars. Our approach is based upon the ideas from the Chronus system [1] and the HVS model [2].

At first the hybrid semantic analysis method is described in this article. Then we test how the method can be adapted to a domain (ATIS corpus, English data, spoken transcriptions) which is very different from the original data (LINGVOSemantics corpus, Czech data, written questions). The last part of the article shows the results achieved on both domains and it compares our results with a state-of-the-art semantic analysis system [15].

2 Related Work

A significant system for stochastic semantic analysis is based on HVS model (hidden vector-state model) [2]. The system was tested on the ATIS and DARPA corpora, recently the system was also used for semantic extraction from bioinformatics corpus Genia [13]. The first model traning was based on MLE (maximum likelihood estimation), however, the discriminative training has also been proposed. According to our knowledge, the system presented in [13] achieved the state-of-the-art performance on the ATIS corpus.

An extension of the basic HVS Parser has been developed in the work of [16]. The improvement is achieved by extending the lexical model and by allowing left-branching. The system was tested on Czech human-human train timetable corpus and it is public available.

Scissor (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations) is another system which uses the syntactic parser enriched with semantic tags, generating a semantically augmented parse tree. Since it uses the state-of-art syntactic parser for English, the Collin's parser, we suppose, that it can not be easily adapted to other languages.

In [18], the generative HMM/CFG composite model is used to reduce the SLU slot error rate on ATIS data. Also a simple approach to encoding the long-distance dependency is proposed. The core of the system is based conditional random fields (CRF) and the *previous slot context* is used to capture non-local dependency. This is an effective and simple heuristic but the system requires a set of rules to determine whether the previous slot word is a filler or a preamble. Thus, it is difficult to port the system to other domain.

3 Semantic Representation

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

The described semantic representation formalism uses the same principle as it was originally described in [2]. The formalism is very advantageous since it does not require annotation of all words of a sentence. It makes it suitable for practical applications where the provision of large scale annotation training data is always complicated.

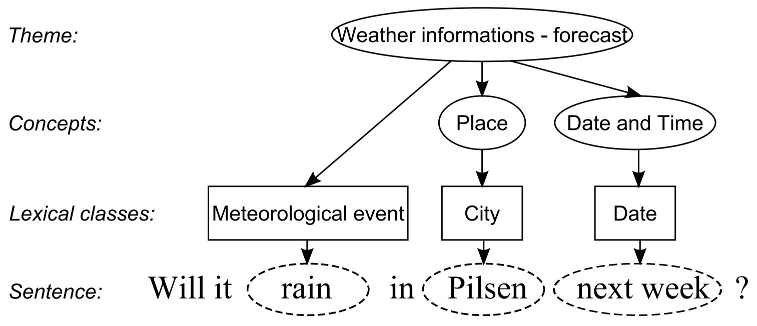

Fig. 1. An example of a semantic annotation tree.

4 Data

This section describes two corpora used for training and testing. The Czech corpus (LINGVOSemantics corpus) was used at the beginning for the development of the method. The English corpus (ATIS) was used to find out whether the designed method is universal and can be successfully used for a different corpus. The second reason for using the ATIS corpus is to compare our method with the state-of-the-art system that has been also tested on the ATIS corpus.

4.1 LINGVOSemantics corpus

The data used during the development are questions to an intelligent Internet search engine. The questions are in the form of whole sentences because the engine can operate on whole sentences rather than just on keywords as usual. The questions were obtained during a system simulation. We asked users to put some questions into a system that looked like a real system. In this way we obtained 20 292 unique sentences. The sentences were annotated with the aforementioned semantic representation (Section 3). More information about the data can be found in [11].

An example of the data follows (*How warm will it be the day after tomorrow?*):

WEATHER(Jaká EVENT(teplota) vzduchu bude DATETIME(DATE(pozítří))?)

4.2 ATIS corpus

One of the commonly used corpora for testing of semantic analysis systems in English is the ATIS corpus. It was used for evaluation in e.g. [2], [3], [4] and [5]. The original ATIS corpus is divided into several parts: ATIS2 train, ATIS3 train, two test sets etc. [6].

The two testing sets NOV93 (448 sentences) and DEC94 (445 sentences) contain the annotation in the semantic frame format. This representation has the same semantic expressive ability as the aforementioned semantic tree representation (Section 3). Each sentence is labeled with a goal name and slot names with associated content.

The corpus does not contain any fixed training set. Originally in [2], 4978 utterances were selected from the context independent training data in the ATIS2 and ATIS3 corpora and abstract semantics for each training utterance were derived semi-automatically from the SQL queries provided in ATIS3.

At this point we have to thank Y. He for sharing their data. It allowed us to test our system on the same testing data that uses their state-of-the-art system for semantic analysis. However, deep exploration revealed that the training data are specially tailored for the HVS model. The data were in the form of HVS model stacks and the conversion from stacks to proper trees was ambiguous and difficult. However, we still were able to use the test data (the test data are stored in the semantic frame format) and the plain sentences from the training data.

Instead of trying to convert the training data from HVS stacks or obtaining the original SQL queries and converting them we decided to annotate a part of the ATIS corpus using the abstract semantic annotation (see section 3). Using the methodology described in [13] we have initially created an annotation scheme[1] from the test data. In the first step, 100 sentences from ATIS2 train set were manually annotated. Thereafter the system was trained using this data. Another set of sentences was automatically annotated and then hand-corrected (this incremental methodology of annotation is called *bootstrapping*). In total, we annotated 1400 random sentences from both ATIS2 and ATIS3 training set.

5 System Description

The system consists of three main blocks (see Figure 2). The *preprocessing* phase prepares the system for semantic analysis. It involves sentence normalization, tokenization and morphological processing. The *lexical class analysis* is explained in Section 5.1 and the *probabilistic parsing* is explained in Section 5.2.

5.1 Lexical Class Identification

The lexical class identification is the first phase of the semantic analysis. During this phase the lexical classes (see Section 3) are being found in the input sentence.

The lexical class identification consists of two stages. First, several dedicated parsers are run in parallel. During this stage a set of lexical classes are found. In the second stage the lexical classes are stored in a lattice. Then the lattice is converted into possible sequences of lexical classes. Only the sequences that contain no overlapping classes are created.

[1] An annotation scheme is a hierarchical structure (a tree) that defines a dominance relationship among concepts, themes and lexical classes. It says which concepts can be associated with which super-concepts, which lexical classes belong to which concepts and so on. More in [13].

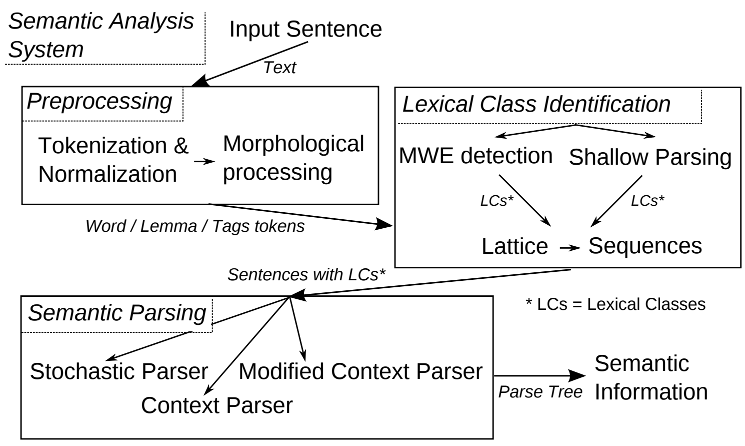

Fig. 2. The structure of our semantic analysis system.

During the first step the lexical classes are being found as individual units. We found in our data two groups of lexical classes:

1. Proper names, multi-word expressions, enumerations.
2. Structures (date, time, postal addresses, ...).

To analyze the first group we created a database of proper names and enumerations (cities, names of stations, types of means of transport etc). Since a lexical class can consist of more than one word it is necessary to look for multiple word expressions (MWEs). To solve the search problem effectively a specialized searching algorithm was developed.

The main ideas of the searching algorithm are organizing the lexical classes in the trie structure [12] and using parallel searching. The algorithm ensures that all lexical classes (possibly consisting of more words) are found during one pass with a linear complexity $O(n)$ (where n is the number of letters of the sentence). The algorithm is explained in [11] in details.

For the analysis of complicated phrases, such as dates, numbers, time, etc., the LINGVOParser [9] was developed. It is an implementation of a context-free grammar parser. The parser has some features suitable for semantic analysis. It uses the so-called *active tags*. Active tags contain processing instructions for extracting semantic information (for more information see [9]).

Another feature of the LINGVOParser is the one we call *partial parsing*. Turning the partial parsing on causes the parser to scan the input sentence and build the partial parse trees wherever possible (a standard parser usually requires to parse the whole input from the start to the end). Partial trees do not need to cover whole sentences and they are used to localize the lexical classes described by a grammar.

During the second stage the lexical classes found in the first stage are put into a lattice. Then the lattice is walked through and the sequences of lexical classes that do not overlap are created. The result of the algorithm is the sequences of lexical classes. The algorithm uses the dynamic programming to build the sequences effectively.

The active tags used in the LINGVOParser allow the system to formalize the content of lexical classes. By formalizing we mean for example the transformation of date time information into one unified format. We can also transform spoken number into their written forms etc. This step is crucial for practical applications where it is required to express the same type of information in the same way [11].

5.2 Semantic Parsing

In the previous section the process of finding lexical classes was described. In this section we will presume that the lexical classes are known and the semantic tree is being built. The structure of the tree is shown in Figure 1. The task of the parsers described here is to create the same trees as in the training data.

Stochastic Parser The parser works in two modes: training and analysis. The training phase requires annotated training data (see Section 4). 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 side 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, identical rules are created during this process. The rules are counted and conditional probabilities of rule transcriptions are estimated:

$$P(N \to \alpha|N) = \frac{\mathrm{Count}(N \to \alpha)}{\sum_\gamma \mathrm{Count}(N \to \gamma)} \ , \tag{1}$$

where N is a nonterminal, α and γ are strings of terminal and nonterminal symbols.

The analysis phase is in no way different from standard stochastic context-free parsing. The sentence is passed to the parsing algorithm. The stochastic variant of the active chart parsing algorithm (see e.g. [7]) is used. The lexical classes identified in the sentence are treated as terminal symbols. 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.

During parsing the probability of the so far created tree $P(T)$ is computed by:

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

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.

Context Parser The context parser looks at other words of the sentence rather than looking at lexical classes only. For this purpose it was necessary 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 5.2. The node is thus transformed into the grammar rule and the frequency of the 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 terminals. The context of the node is defined as the words before and after the span of the subtree. During training the frequency of the context and a nonterminal (Count($word, nonterminal$)) are 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:

$$P(T) = \sum_i P(w_i|N)P(N \to A_1A_2...A_k|N) \prod_j P(T_j) \ . \tag{6}$$

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.

Modifications for Morphologically Rich Languages We tried to further improve the performance of parsing algorithms by incorporating features that consider the specific properties of the Czech language. The Czech language is a morphologically rich language [8] and it has also a more flexible word order than for example English or German. To deal with specific properties of the Czech language *lemmatization* and *ignoring word order* features were incorporated to the Context Parser.

6 Performance Tests

6.1 Results on the LINGVOSemantics corpus

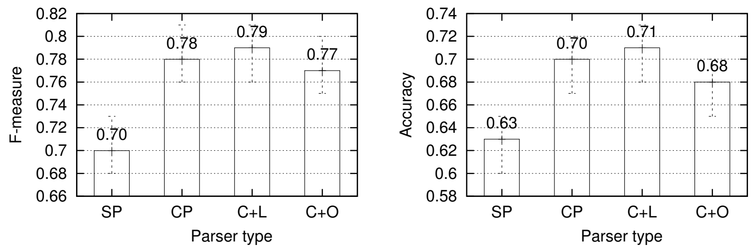

Fig. 3. Results of Semantic Parsers. *SP* = Stochastic Parser, *CP* = Context Parser, *C+L* = Context + Lemma, *C+O* = Context + word Order. Confidence intervals are computed at confidence level: $\alpha = 95\%$.

Figure 3 shows the results for the LINGVOSemantics corpus (Section 4.1). The figure shows performance of the base line parser (Stochastic Parser – section 5.2), the novel parser (Context Parser – section 5.2) and the modifications of the Context Parser (section 5.2).

The results are measured in the *accuracy* and *f-measure* metrics. Both the metrics use the slot-value pairs for computation. The slot is the name of a slot and its path in the slot hierarchy and the value is the value of the slot. Accuracy and F-measure are standard metrics for multiple outputs (one semantic tree or semantic frame consists of more slot-value pairs) and the formulas are defined in [2]. We use the same metrics as in [2] for sake of mutual comparison of our results and the results in [2].

6.2 Results on the ATIS corpus

The adaptation of the system to the ATIS corpus consisted of two steps. First, an appropriate English context-free grammar covering the date, time and numbers was created for the LINGVOParser (see section 5.1). Aditionally, the multiword expression identificator was re-trained using the data from ATIS `*.tab` files that

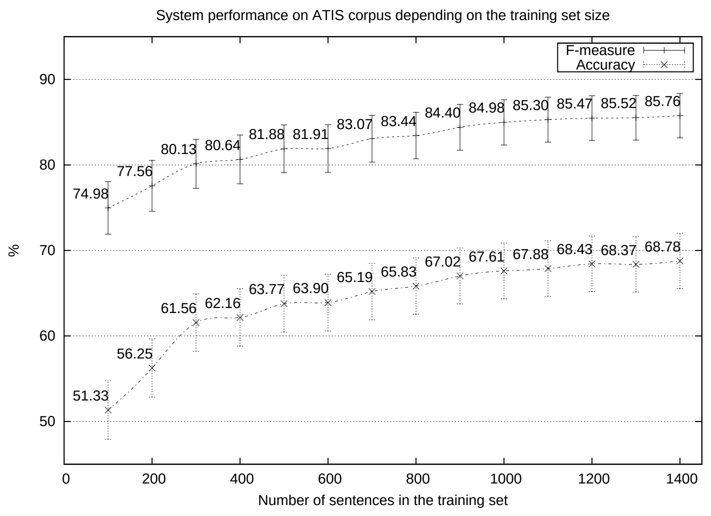

Fig. 4. System performance on the ATIS corpus with various amount of training data. Confidence intervals are computed at confidence level: $\alpha = 95\%$.

contain cities, airports, etc. Second, the semantic parser was trained using the training set described in section 4.2.

Figure 4 shows the results on the ATIS corpus depending on training data size. The training sentences were chosen randomly, ten times from each set. The best result achieved on 1400 training sentences was 85.76%. When compared to [2] (89.28%) or [15] (91.11%), we must consider that in [2] and in [15] their systems were trained on a larger training set (4978 utterances). The reasons that we used a smaller set are explained in section 4.2.

However, we have discovered a significant amount of inconsistencies in the ATIS test set. It contains ambiguities in semantic representation (e.g. two same slots for one sentence), multiple goals, or interpreted data in slots (e.g. airport names which do not appear in the sentence at all).[2] Thus, we think that the performance testing on this corpus is affected by the testing set inconsistency and the objectivity of the evaluation is compromised.

7 Conclusions

This article described the hybrid semantic parsing approach. The tests performed on ATIS data show that we almost reached the performance of the state-of-the-art system on a reduced training data set. It is probable that by fine-tuning the system and by annotating the full training set, the system could be capable of reaching the state-of-the-art performance. We, however, consider the results

[2] The examples are shown at http://liks.fav.zcu.cz/mediawiki/index.php/Interspeech_2010_Paper_Attachment

sufficient to prove that the hybrid approach with the context parser is worth of further development.

To compare our system with a very similar system (described in [15]) it can be concluded that a significant progress was made in two areas. Firstly, the annotation methodology was improved. It is is now faster and more fault-proof. Secondly, our system is prepared to be used in practical applications by using data formalization. In [15] the lexical classes are automatically learned without the possibility of data formalization. We however use a hybrid approach where the data formalization is used. In the near future we are going to publish papers on the results achieved under real conditions.

Acknowledgements. This work was supported by grant no. 2C06009 Cot-Sewing. We would like to thank Mrs. Yulan He for sharing the ATIS semantic corpus.

References

1. Pieraccini, R., Tzoukermann, E., Gorelov, Z., Levin, E., Lee, C-H., Gauvain, J-L.: Progress Report on the Chronus System: ATIS Benchmark Results. *In Proc. of the workshop on Speech and Natural Language*, 1992.
2. He, Y., Young, S.: Semantic processing using the Hidden Vector State model. Computer Speech and Language, Volume 19, Issue 1, 2005, 85–106.
3. Iosif E., Potamianos A.: A soft-clustering algorithm for automatic induction of semantic classes. In *Interspeech-07*, pages 1609–1612, Antwerp, Belgium, August 2007.
4. Jeong M., Lee G.: Practical use of non-local features for statistical spoken language understanding. Computer Speech and Language, 22(2):148–170, April 2008.
5. Raymond Ch., Riccardi G.: Generative and discriminative algorithms for spoken language understanding. In *Interspeech-07*, pages 1605–1608, Antwerp, Belgium, August 2007.
6. Deborah A. Dahl, et al., ATIS3 Test Data, Linguistic Data Consortium, Philadelphia, 1995
7. Allen, J.: *Natural Language Understanding.* Benjamin/Cummings Publ. Comp. Inc., Redwood City, California, 1995.
8. Grepl M., Karlík P.: Skladba češtiny, Olomouc, Czech Republic, 1998.
9. Habernal, I., Konopík, M.: Active Tags for Semantic Analysis, In TSD '08: Proceedings of the 11th international conference on Text, Speech and Dialogue, 69–76, ISBN 978-3-540-87390-7, 2008
10. Jurafsky, D., Martin, J.: *Speech and Language Processing.* Prentice Hall, 2000.
11. Konopík, M.: Hybrid Semantic Analysis, PhD thesis, University of West Bohemia, 2009
12. Knuth, D.: *The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition.* Addison-Wesley, 1997. ISBN 0-201-89685-0.
13. Habernal I., Konopík M.: Semantic Annotation for the LingvoSemantics Project. In Proceedings of the 12th international Conference on Text, Speech and Dialogue. Lecture Notes In Artificial Intelligence, vol. 5729. Springer-Verlag, 2009, Berlin, Heidelberg, 299-306. ISBN 978-3-642-04207-2.

14. Konopík M., Habernal I.: Hybrid Semantic Analysis. In Proceedings of the 12th international Conference on Text, Speech and Dialogue. Lecture Notes In Artificial Intelligence, vol. 5729. Springer-Verlag, 2009, Berlin, Heidelberg, 307-314. ISBN 978-3-642-04207-2.
15. He Y., Zhou D.: Discriminative training of the hidden vector state model for semantic parsing.*IEEE Trans. on Knowl. and Data Eng.*, 21(1):66–77, 2009.
16. Jurčíček F.: Statistical approach to the semantic analysis of spoken dialogues. PhD thesis, University of West Bohemia, Faculty of Applied Sciences, 2007.
17. Kate J., Mooney R. J.: Using string-kernels for learning semantic parsers, In ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pages 913–920, Morristown, NJ, USA, 2006.
18. Wang Y., Deng L., Acero A.: An introduction to statistical spoken language understanding. IEEE Signal Processing Magazine, 22(5):16–31, 2005.