STOCHASTIC SEMANTIC ANALYSIS

Miloslav Konopík

University of West Bohemia, FAV, KIV
Univerzitní 8, 306 14 Pilsen, Czech Republic
Pilsen, CZECH REPUBLIC
E-mail: `konopik@kiv.zcu.cz`

Abstract: Speech is the most natural way of human communication. Therefore, there is an effort to incorporate speech control into human-computer interfaces. However, nobody likes the idea of remembering a large amount of specific commands and so the ability of understanding the meaning of an utterance is crucial for many speech-enabled computer systems.
This article describes a very promising method suitable for analysis of the meaning contained in an utterance. The described method extends Markov models so that every Markov state stores a semantic vector. This extension allows the model to capture hierarchical structures.

Keywords: semantic analysis, natural language understanding, semantic parsing

1. INTRODUCTION

This paper is part of the *COT-SEWing*[1] project. The purpose of the project is to develop a complex base of tools which will remove some of the typical barriers present in communication between human user and computer within the scope of internet access. One of the goals is to enable voice control of an internet application. The analysis of spoken input involves two main areas: voice recognition and utterance understanding. In this paper, we focus on the second area, on the utterance understanding.

Natural Language Understanding (NLU) is a process whereby a computer algorithm extracts the meaning of an utterance and embeds the meaning in the computer model of the world. **Semantic analysis** is the first step of the NLU process. The goal of semantic analysis is to represent what the subject intended to say in a way that would facilitate the process of interpretation [2].

A semantic analyzer of a spoken language system must be able to deal with spontaneous speech effects such as unconstrained formulations, ill formed expressions, repairs, false starts and unknown words. Due to grammatical problems of spoken input, syntax should not play significant role during utterance processing. The corpus of COT-SEWing project is in Czech language, therefore we try to adapt existing algorithms for semantic analysis to the Czech language.

[1]Complex Knowledge Base Tools for Natural Language Communication with the Semantic Web

[2]interpretation = reasoning about the meaning of the utterance; interpretation is explained further for example in (Konopík and Mouček, 2005)

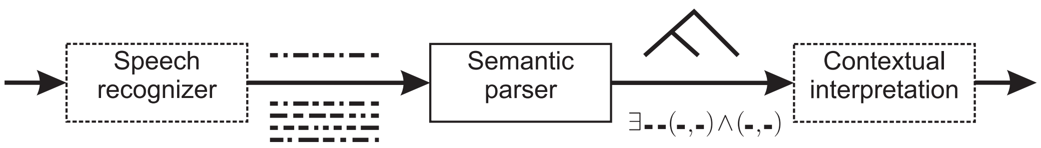

Fig. 1. The interaction of the semantic analysis system with other modules. The semantic parser input is indicated as the most likely utterance transcription (top) or a word lattice (bottom). The output is either a tree (top) or a logic representation (bottom)

A very promising method of semantic analysis is the Vector-state Markov model (see (Young, 2002)). When we train this model in an unsupervised manner then the model is called Hidden Vector-state Markov model (HVM). A very good description of HVM model can be found in (Yulan and Young, 2005). Our intension is to test this model in the COT-SEWing project conditions. That involves the issue of Czech language. The HVS model was originally proposed for English language. So, we try to adapt the HVS model to the Czech language.

The remainder of this paper is organized as follows. First, the problem is specified by the definition of the input and the output of the semantic analysis algorithm. Then the Vector-state Markov model is introduced. And finally, the unsupervised training of the Vector-state Markov model is explained.

This article assumes that the reader is familiar with basic terms of the analysis of language (parsing, parse tree, etc.). If it is not true, please see e.g. (Jurafsky and Martin, 2000) or (Konopík, 2006). (Konopík, 2006) is oriented towards stochastic semantic analysis while (Jurafsky and Martin, 2000) is much more general.

2. PROBLEM DEFINITION

This section specifies the problem of semantic analysis by the definition of the input that enters into the semantic analysis algorithm and the output that results from the algorithm.

The input is the orthographic transcription of an utterance. The form of the transcription can be either the most likely transcription of the utterance or even better a word lattice. Prosody or some nonverbal features may be included as well. Stochastic semantic analysis methods in particular profit from the presence of other information sources (prosody, etc.).

The output is the context-independent meaning of the utterance in a suitable meaning representation. The requirements about the output of semantic analysis are stated in (Jurafsky and Martin, 2000). In short, the result of semantic parsing is required to support the interpretation that follows the process of semantic analysis (see fig. 1).

3. VECTOR-STATE MARKOV MODEL

The key feature of the 1st order Markov model (see e.g. (Rabiner, 1989)) is that the current state at time t holds all of the information needed to account for the observation at time t and the transition to a new state at time $t+1$. It is this property which gives the model its mathematical simplicity. The Vector-state Markov Model improves the basic model by extension of the state space to record broader context. Every Markov state is extended to contain a vector of semantic concepts. The vector of concepts is utilized to capture the hierarchic structure of an utterance.

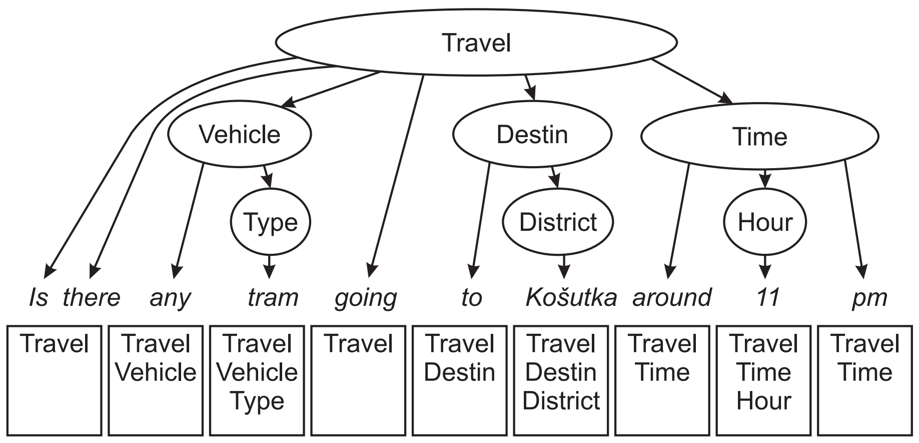

Fig. 2. Example of a right branching tree and corresponding vector state sequence. Borrowed from .

The information in a parse tree relating to any single word can be stored as a vector of node names starting from the preterminal and ending in the root node. For instance, see fig. 2 and the state associated with the word "tram". This state contains a semantic vector [Type, Vehicle, Travel]. The complete parse tree can be represented by a sequence of semantic vectors (see bellow in the fig. 2). If the nodes in a parse tree are all distinct from each other, there is a one-to-one mapping between semantic vector sequences and a parse tree. The semantic vector has indeed the structure of a stack because new nodes are put at the end of the vector and the last nodes are taken first from the vector while parsing a sentence.

The parsing of a sentence works within Vector-state Markov model in the following way: the parser goes left to right, each time a word is processed, a transition based on current state is triggered. Every transition is constrained to take the form of stack shift (a number of nodes are removed) followed by a push of one or more nodes. Please note, when the transitions have exactly the form of removing one node followed by pushing of one node, we get an ordinary 1st order Markov model.

Given an unlimited stack, any PCFG[3] formalism can be converted into a Vector-state Markov Model. The problem, however, is that in doing so the state space rapidly becomes huge. Thus, the form of transitions can be limited to alleviate the model complexity. Following transition limitation was proposed in (Young, 2002): a stack shift followed by pushing at most one node. The effect of this on parse tree of semantic concepts is to make it right branching. The example in fig. 2 was made in this manner.

The generative process (eq. 1) associated with this model consists of three steps for each word position t: (a) choose a value n_t which represents a number of removed nodes; (b) select a preterminal node c_{w_t} for word w_t; (c) select a word w_t. As with the PCFG model, the probability distribution $P(W, \mathbb{C}, N)$ can be decomposed either top-down or bottom-up. The top-down decomposition is presented in 1:

$$P(W, \mathbb{C}, N) = \prod_{t=1}^{T} \overbrace{P(n_t|W_1^{t-1}, C_1^{t-1})}^{\text{(a)}} \overbrace{P(c_t[d_t]|W_1^{t-1}, C_1^{t-1}, n_t)}^{\text{(b)}} \underbrace{P(w_t|W_1^{t-1}, C_1^{t})}_{\text{(c)}} \tag{1}$$

[3]PCFG = Probabilistic Context-Free Grammar, see e.g. (Konopík, 2006) – section 3.3.2

where $\mathbb{C}$ is matrix whose columns are vector stacks $\vec{c}$; d_t denotes the length of the stack $\vec{c}_t$ ($d_t = |\vec{c}_{t-1}| - n_t + 1$); $c_t[d_t]$ is a new preterminal symbol on the top of the stack $\vec{c}_t$.

The eq.1 is too complex to be used directly. Instead, it has to be approximated. In the version of the Vector-state Markov model discussed in (Young, 2002), the components of eq. 1 are approximated by:

$$P(n_t|W_1^{t-1}, C_1^{t-1}) \approx P(n_t|\vec{c}_{t-1}) \tag{2}$$
$$P(c_t[d_t]|W_1^{t-1}, C_1^{t-1}, n_t) \approx P(c_t[d_t]|c_t[1..d_t - 1]) \tag{3}$$
$$P(w_t|W_1^{t-1}, C_1^{t}) \approx P(w_t|\vec{c}_t) \tag{4}$$

where $c_t[1..d_t - 1]$ denotes stack elements 1,2,..,$d_t - 1$.

Then, by substitution 2 – 4 in 1 we get the 5. This formula is being used in practical applications.

$$P(W, \mathbb{C}, N) \approx \prod_{t=1}^{T} P(n_t|\vec{c}_{t-1})P(c_t[d_t]|c_t[1..d_t - 1])P(w_t|\vec{c}_t) \tag{5}$$

The supervised training is relatively simple and consists in events counting, normalization and smoothing. The unsupervised training is discussed in the next section.

Unlike the PCFG model, this probabilistic model is well-suited to left-right decoding. Since each partial path covering word sequence W_1^t contains exactly the same number of probabilities, paths can be compared directly without normalization.

The Vector-state Markov model extends the HMM model by expanding each state to encode the stack of a push-down automaton. This allows the model to encode hierarchical context. This model stands in between the HMM and PCFG model. With the limited stack it covers regular languages only (as well as an HMM). However, by extending the capacity of the stack it approaches the context-free languages. This model was proposed to be simple and thus reliable and to cover natural language at the same time. The formerly described transition limitation covers right-branching languages only. In (Yulan and Young, 2005), it is claimed that majority of English sentences are right-branching. Still, it has to be examined, whether this statement is valid in Czech language. If not, an appropriate modification has to be applied.

4. HIDDEN VECTOR-STATE MODEL

The authors of (Yulan and Young, 2005) claim that the provision of fully annotated data is not realistic in practice, thus the supervised training is not possible. So they developed a method of unsupervised training of the Vector-state Markov Model. However, the fully unsupervised training is not possible in such a complex model. Some prior knowledge has to be present. The prior knowledge and the training equations are described further in this section. The model and the method of training together create the Hidden Vector-state Model (HVM).

The prior knowledge in HVM consists of the two following parts:

- A set of domain specific lexical classes.
- Abstract semantic annotation for each utterance.

1. I want to go Chicago to arrive around 11am.
2. I need arrive about noon in New York.
3. I have to be in Boston at 10am.
4. Find flights arriving in Dallas mid-morning.

```
TRAVELREQ(
  TOPLACE(
    CITY,
    TIMESPEC(TIME)
))
```

Fig. 3. An abstract annotation of utterances carrying the same meaning.

Lexical classes typically group proper names into hypernym[4] class. For example, in a flight information system, typical classes might be CITY = {Boston, Denver, New York, ...}, AIRPORTS = {Dulles, ...}, etc. These domain specific classes can be usually extracted from the domain database schema. The generic classes (covering times, dates, etc.) may be also included in domain specific classes. Given these classes, all words in the training set are replaced with corresponding lexical class when possible. This reduces the size of the model.

The *abstract semantic annotation* lists a set of applicable semantic concepts with the dominance relationship between them for each training utterance. However, it does not take any account of word order or attempt to annotate every part of the utterance. Every particular utterance that has the same database SQL query should have the same abstract semantic annotation (for example, see figure 3).

The training of this model involves *preprocessing*, *parameter initialization* and *parameter re-estimation*.

The preprocessing prepares the data for training (mostly filtering). Parameters are initialized by so-called *flat-start* whereby all model parameters are initially made identical.

The parameter re-estimation is defined in (Yulan and Young, 2005) as follows: Let the complete set of model parameters be denoted by λ, EM-based parameter estimation aims to maximize the expectation of $L(\lambda) = \log P(N, \mathbb{C}, W|\lambda)$ given the observed data and current estimates. To do this, the 6 expression has to be maximized with respect to $\hat{\lambda}$:

$$\sum_{N,\mathbb{C}} P(N, \mathbb{C}|W, \lambda) \log P(N, \mathbb{C}, W|\hat{\lambda}) \tag{6}$$

Substituting eq. 5 into eq. 6 and differentiation lead to the following re-estimation formulae:

$$P(n_t|\vec{c}) = \frac{\sum_t P(n_t = n, \vec{c}_{t-1} = \vec{c}|W, \lambda)}{\sum_t P(\vec{c}_{t-1} = \vec{c}|W, \lambda)} \tag{7}$$

$$P(c_t[d_t]|c_t[1..d_t - 1]) = \frac{\sum_t P(\vec{c}_t = \vec{c}|W, \lambda)}{\sum_t P(c_t[1..d_t - 1] = c[1..d_t - 1]|W, \lambda)} \tag{8}$$

$$P(w_t|\vec{c}_t) = \frac{\sum_t P(\vec{c}_t = \vec{c}|W, \lambda)\delta(w_t = w)}{\sum_t P(\vec{c}_t = \vec{c}|W, \lambda)} \tag{9}$$

where $\delta(w_t = w)$ is one iff the word at time t is w, otherwise it is zero.

The key components of the above re-estimation formulae are the likelihoods $P(n_t = n, \vec{c}_{t-1} = \vec{c}|W, \lambda)$, $P(\vec{c}_t = \vec{c}|W, \lambda)$ and $P(c_t[1..d_t - 1] = c[1..d_t - 1]|W, \lambda)$. These likelihoods can be

[4]hypernym relates a more general term with a lexeme, e.g. vehicle is hypernym of car, for more detail see (Jurafsky and Martin, 2000) – chapter 16

efficiently calculated using the forward-backward algorithm. The formulae for the forward and backward probabilities (α, β) are defined in (Young, 2002) in detail.

The (Young, 2002) also defines the formulae for model adaptation. Regardless of how much training data is available, a semantic decoder will perform badly if the training data does not well-represent the test data. Model adaptation can be used to increase the performance of semantic decoder.

The experiments conducted on ATIS corpus and DARPA Communicator Travel Data in (Yulan and Young, 2005) are very promising. F-measure of automatic and human annotations was around 88%. Such a result is very good in the area of semantics.

5. CONCLUSION

This paper describes the HVS model and explains the training of this model.

The HVS model solves the problem of the need to collect a large quantity of fully annotated tree-bank data. It proves that the Vector-state Markov model is constrained enough to be trained by partially unsupervised training whilst at the same time it is able to capture hierarchical dependencies.

ACKNOWLEDGEMENTS

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

REFERENCES

Jurafsky, Daniel and James H. Martin (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR. Upper Saddle River, NJ, USA.

Konopík, Miloslav (2006). Stochastic semantic parsing. Technical Report DCSE/TR-2006-01. University of West Bohemia in Pilsen.

Konopík, Miloslav and Roman Mouček (2005). An alternative way of semantic interpretation. In: *Text, Speech and Dialogue*. Vol. 3658/2005. Springer Berlin / Heidelberg. pp. 348–355.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**(2), 257–286.

Young, Steve (2002). The statistical approach to the design of spoken dialogue systems. Technical Report CUED/F-INFENG/TR.433. Cambridge University Engineering Department.

Yulan, He and Steve Young (2005). Semantic processing using the hidden vector state model. *Computer speech & language* **19**(1), 85–106.