ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Stochastic Semantic Analysis

PhD Study Report

Ivan Habernal

Technical Report No. DCSE/TR-2009-04
May, 2009

Distribution: Public

Abstract

In human-computer dialogue systems, the task of Spoken Language Understanding (SLU) system is to process the input acoustic utterance and transform it into a semantic representation. The goal of *semantic analysis* is to represent what the subject intended to say.

This thesis presented overview of the current state-of-the-art methods for statistical semantic analysis. In comparison to the expert based systems, the main advantage of stochastic approaches is the ability to train the model from data. Furthermore, systems based on statistical models can be easily ported to other domains. However, the amount of the annotation effort must be also taken into account when developing the stochastic semantic analysis system. In this thesis, fundamental stochastic models along with the training and evaluation of these models are described.

Copies of this report are available on
http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address:

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitní 8
30614 Pilsen
Czech Republic

Contents

Chapter 1

Introduction

The goal of Spoken Language Understanding system (SLU) is to extract the meaning from the natural speech. The SLU covers many subfields such as utterance classification, speech sumarization, natural language understanding (NLU) and information extraction.

In human-computer dialogue systems, the task of SLU system is to process the input acoustic utterance and transform it into a semantic representation. However, this task can be split into two parts – *automatic speech recognition* (ASR) and *semantic analysis*.

In this thesis, we will focus on semantic analysis. The purpose of a semantic analysis system is to obtain a context-independent semantic representation from a given input sentence.

Although there is a difference between SLU and NLU in the sense of the input (audio signal for SLU systems, text for NLU systems), we will not distinguish bewteen these terms so strictly. The main reason is that many of the presented approaches and systems are originally focused on the whole SLU system, even though they deal with semantic analysis and assume a textual representation on the input. A schematic example of SLU system is in Fig. 1.1.

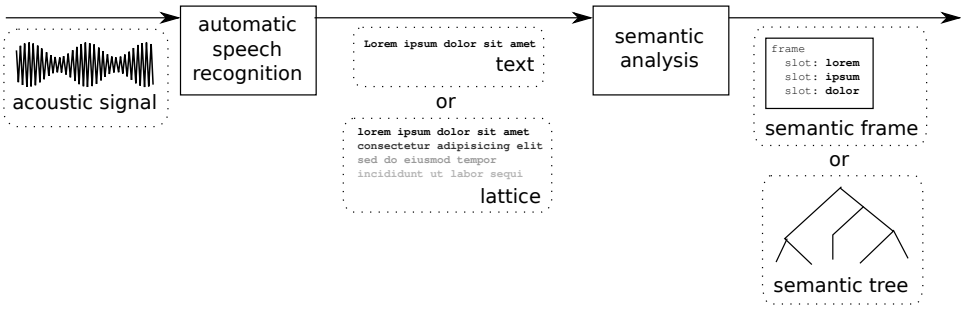

Figure 1.1: A schema of a SLU system.

Early SLU systems were mostly based on an *expert approach*, which means that the system is written entirely by the system designer (an expert). The *syntax-driven semantic analysis* [All95] uses a hand-written context-free grammar (CFG) for syntax. The first-order predicate calculus (FOPC) is used for meaning representation. Later, *semantic grammars* [JM00] were based on CFGs and described the semantic hierarchy rather then the language syntax.

Hovewer, the expert-based systems have a lot of limitations. The first disadvatage is very high cost of creating such system because the grammars must be written by an expert. Such system can also cover limited domain and it lacks portability.[1] Althought expert-based systems are still beeing used, recently the majority of semantic analysis systems is based on statistics. The main advantage of such systems is the ability to learn from data.

Thus, the state-of-the-art statistical methods for SLU are presented in this thesis.

1.1 Problem definition

As shown in Fig. 1.1, **the input** of the semantic analysis is an output from the ASR module. It can be either a plain text (a single sentence), a word lattice or n-best list, whter the sentences are ordered by its probabilites assigned by the ASR.

The output of the SLU system is a context-independent[2] semantic representation. There are two most commonly used representations of the semantics – *frame based* and *tree-based* representation.

In the frame-based SLU system, the semantic representation of an application domain can be defined in terms of *semantic frames.* Each frame contains several components called slots. The SLU system fills the slots with appropriate content. The meaning of an input sentence is an instantiation of the semantic frame. Some SLU systems do not allow a hierarchy of slots. In such case, the semantic representation is a *flat concept* representation (or *attribute-value pairs*).

Since the flat concept representation is simpler and it may result in simpler statistical model, the hierarchical representation (tree-based representation) is more expressive and it can deal with long-dependencies. Both models will be described and compared in the next chapter.

[1] *Porting* means adapting the system to different domain

[2] Does not depend neither on the history nor on the context.

1.2 Structure of the thesis

- Chapter 2 describes the sequential models for semantic analysis. These models use a non-hierarchical representation of the semantics.
- More complex models are presented in chapter 3, as well as other approaches for semantic parsing.
- Chapter 4 describes the evaluation of semantic analysis systems.
- Chapter 5 deals with data annotation and annotation methodology.
- An overview of existing systems and corpora is provided in chapter 6.
- The aims of the doctoral thesis are presented in chapter 7.

Chapter 2

Sequential models

2.1 Hidden understanding model

The Hidden understanding model (HUM) [SMSM97] was motivated by Hidden Markov Models (HMM), that have been succesful in speech recognition. Because of differences between speech recognition and language understanding, significant changes are required in the HMM methodology. [MBIS94] proposes the following requirements for hidden understaning systems:

- A system for expressing meaning
- A statistical system which is capable to capture associations between words and meaning
- A training algorithm for estimating the parameters of the model from annotated data
- An algorithm for performing the search for the most-likely meaning given a word sequence

The key requirement for a hidden understanding model are the properties of the meaning expression. It should be both precise and appropriate for automatic learning techniques. [MBIS94] requires a meaning representation that is:

Expressive. For all sentences that appear in the application, the formalism must be able to express the meaning.

Annotable. It must be possible to create annotations of large corpus with low human effort.

Trainable. The system must be able to train the parameters from training annotated examples.

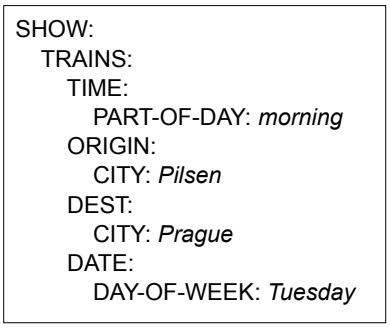

Figure 2.1: A frame-based representation for input sentence *"Show me morning trains from Pilsen to Prague on Tuesday"*.

Tractable. There must be an efficint algorithm for performing the search over the meaning space.

Fig. 2.1 shows the Frame based meaning representation. This representation is very simple but it has also a disadvatage. The problem is that generally there is no explicit alignment of frames to the the words from the sentence. Fig. 2.2 shows another representation – the tree based meaning representation as proposed in [SMSM97]. In such representation, the words are directly aligned in the structure. The cost is that tree based representation is more detailed and therefore it requires more human annotation effort.

Tree based representation

The main characteristic of tree structured representation is that semantic concepts appear as nodes of a tree. Some concepts can be attached to parent concept node. The leaves of the tree are directly mapped to the words from the sentence and they are called terminal nodes in [SMSM97] (or preterminal semantic concept in [You02]). Each word is mapped to its corresponding terminal node.

Such representation can be viewed as a product of parsing by a semantic grammar as well. But in HUM the relations between concepts are defined in an annotation schema and the relations are trained from annotated data.

Frame based representation

The frame based representation differs from the tree representation in word alignment. While tree based representation assigns all tree nodes to the words, in the frame based representation there can be words that are ommited. These missing terminals are said to be hidden, in the sense that

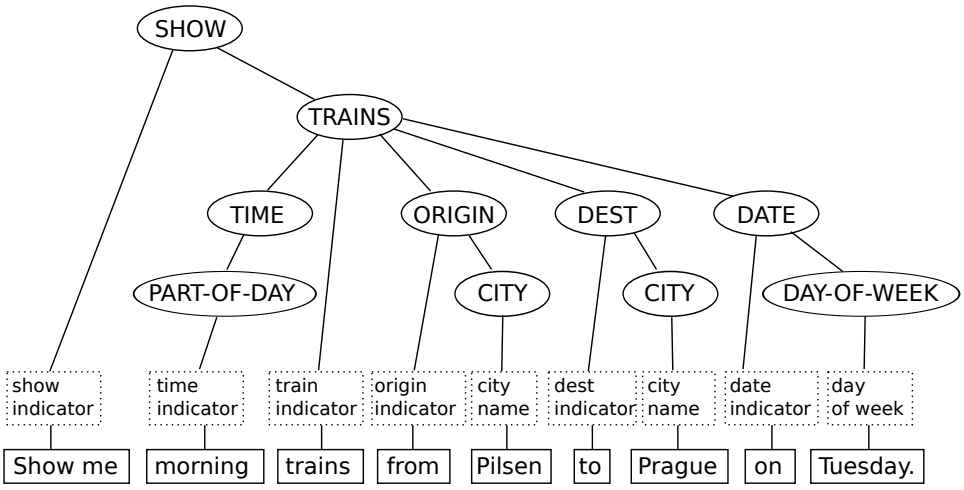

Figure 2.2: A possible tree-based semantic representation for input sentence *"Show me morning trains from Pilsen to Prague on Tuesday"*.

every word has to be aligned to some terminal node but the alignment is not given by the meaning frame.

Statistical model

Let $\mathscr{M}$ be the meaning space and $\mathscr{V}$ the vocabulary. Then the problem of understanding can be viewed as recovering the most likely meaning structure $M \in \mathscr{M}$ given a sequence of words $W \in \mathscr{V}$:

$$\hat{M} = \underset{M}{\operatorname{argmax}}\, P(M|W) \tag{2.1}$$

Using Bayes rule, $P(M|W)$ can be rewritten into

$$P(M|W) = \frac{P(W|M)P(M)}{P(W)} \tag{2.2}$$

where $P(M|W)$ is *lexical realization model* and $P(M)$ is *semantic language model*. Since $P(W)$ does not depend on M, it can be ignored for computing maximal probability $P(M|W)$. There is an analogy with HMM because only words can be observed and the internal states of each of the two models are unseen and must be inferred from the words.

2.2 Flat-concept parsing

An semantic decoding approach inspired by HUM (section 2.1) is the *Finite State Tagger* [HY05] (or *flat-concept model* in [You02]). It assumes that each

word w from the sentence is labeled with a semantic concept c. For example the sentence *"What will be the weather in Pilsen tomorrow morning?"* might be decoded in bracket notation (see Chapter 5) as:

```
WEATHERREQ(weather) PLACE(Pilsen) DATE(tomorrow) TIME(morning)
```

Irrelevant words are labeled with a *dummy concept* and later discarded from the semantic annotation. The formal definition of the model follows.

Let $\mathscr{V}$ be the vocabulary and $W = (w_1, \ldots, w_T)$ the word sequence where $w_t \in \mathscr{V}$. Given the semantic concept space $\mathscr{T}$, we can assume that each word w_t is tagged with one semantic label concept c_t and the whole sequence is $C = (c_1, \ldots, c_t)$, where $c_t \in \mathscr{T}$. The FST model can then be described as follows:

$$P(W|C)P(C) = \prod_{t=1}^{T} P(w_t|w_{t-1} \ldots w_1, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \ldots c_t) \tag{2.3}$$

where $P(W|C)$ is the probability of generating word w_t given the word history $w_{t-1} \ldots w_1$ and corresponding current concept c_t. It is called *lexical model* (or *lexical realization model* in HUM). The *semantic model* $P(C)$ is the probability of generating current concept c_t given the concept history $c_{t-1} \ldots c_1$.

This model uses an unlimited history of words and concepts. In practical applications, the history is truncated to a limited size n and m. The model is now approximated by the following formula:

$$P(W|C)P(C) \approx \prod_{t=1}^{T} P(w_t|w_{t-1} \ldots w_{t-n+1}, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \ldots c_{t-m+1}) \tag{2.4}$$

For special case $n = 1$ and $m = 2$, we can rewrite the formula as:

$$P(W|C)P(C) = \prod_{t=1}^{T} P(w_t|c_t) \prod_{t=1}^{T} P(c_t|c_{t-1})$$

which becomes a conventional first order Markov model, where state transitions are modelled as concept bigram probabilities and words are modelled as unigram conditioned by the concept c_t.

An example of the model is shown in Fig. 2.3.

2.3 Conditional random fields

The SLU problem can be stated as a sequential supervised learning problem [NSP06]. The result of the classifier is the semantic label sequence which

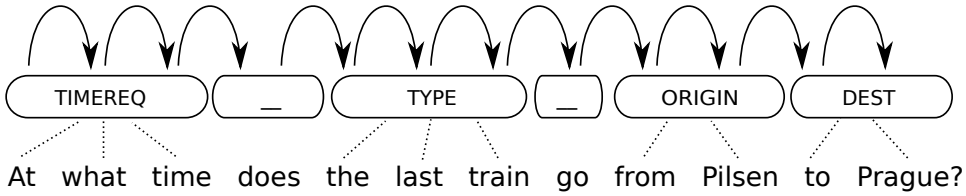

Figure 2.3: Discrete Markov model representation of semantics

can be transformed into the slot/value pairs. In such sequential processing, the hierarchy of the semantic representation can be defined by the slot description.

Let $\mathcal{D} = \{X^i, Y^i\}_{i=1,\ldots,N}$ be a set of N training examples where each example is a pair of sequences (X^i, Y^i). The $X^i = \langle x_1^i, \ldots, x_{T_i}^i \rangle$ features *vector sequence* and $Y^i = \langle y_1^i, \ldots, y_{T_i}^i \rangle$ is a *label sequence*. For example, X can be a sequence of words and Y can be the sequence of corresponding semantic labels. We also assume that X and Y are the same size. The goal of classifier h is to find the best probable semantic class sequence given the input vector:

$$\hat{Y} = \underset{Y}{\operatorname{argmax}}\, h(Y, X, \Lambda), \tag{2.5}$$

where $\Lambda = \lambda_k$ denotes the parameter vector of size K.

Linear-chain Conditional Random Fiels (CRFs) are conditional probability distributions over label sequences which are conditioned by the input sequence [WDA05], [RR07], [NSP06]. Formally, linear-chain CRFs are defined as follows:

$$p_\Lambda(Y|X) = \frac{1}{Z(X)} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, X), \tag{2.6}$$

where X is the random vector that is observed, Y is the vector we want to predict and Ψ is the local potential ([JL08], also called a *feature function* in [SP03]). $Z(X)$ is a normalization function which ensures that the probabilities of all state sequences sum up to one. Typically, Ψ consist of a set of feature functions f_k.

In general, the feature function $f_k(y_t, y_{t-1}, X)$ is an arbitrary linguistic function. Typically, a feature depends on the inputs around the given position i, although they may also depend on global properties of the input [JL08]. Formally, the feature can encode any aspect of a state transition $f_k(y_{t-1}, y_t)$ and the observation $f_k(y_t, x_t)$, centered at the position t.

For example, the feature function $f_k(y_t, x_t)$ can be a binary function which yields 1 iff $y_t =$ `'TOLOC.CITY NAME-B'` and current word is `'chicago'`. For higher values of λ_k the event occurs more likely.

Parameter estimation of linear-chaining CRFs is usually performed by conditional maximum log-likelihood [SP03]. To avoid overfitting, the penalization is used on a vector with a too large norm:

$$\mathcal{L}(\Lambda) = \sum_{i=1}^{N} \log P_\Lambda(Y^i|X^i) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2} \tag{2.7}$$

However, [SP03] tested other algorithms for CRF training, such as preconditioned conjugate-gradient, limited-memory quasi-Newton and voted perceptron.

Chapter 3

Stochastic semantic parsing

3.1 Prerequisites

Most of the stochastic semantic parsers, which are introduced in this chapter, need a preprocessing step [HY06b], [PMS+01], [WM06]. Since the terminology for this task is not stable, the problem can be called *shallow semantic parsing* [PWH+04], *lexical class analysis* [Kon09], *named entity recognition* [TKSDM03] or *NP-chunking* [NSP06]. In fact, these methods attempt to identify semanticaly meaninguful units (words or word groups) in the input sentence and assign a semantic label to them. This definition of semantic classes of words is necessary in order to obtain high coverage models from the given data [PMS+01]. There is a strong paralelism with the stochastic approach applied to the problem of tagging text; not only Part-of-speech tagging, but also detection of syntactic structures as noun phrases etc (*NP-chunking*). Depending on the approach, the results of this step may vary from a set of lexical classes to a lattice, where the semantic labels are assigned to the words with some probability.

Some algorithms dealing with this problem were described in the previous chapter. Namely, the *finite state tagger* (FST) in section 2.2 or the conditional random fields (CRF) in section 2.3. Some other approaches as *NP-chunking*, stochastic finite state transducers (SFST) or classifier based sequence labeling are presented in the sources (eg. [CRR08], [IP07]) in more detail, as well as the training methods.

3.2 Probabilistic semantic grammars

The flat-concept parser described in section 2.2 has some limitations on its expression ability. The most important drawback is that the model does not allow to capture any hierarchical structure which groups corresponing concepts into a covering semantical concept. The long-distance dependecy problems (see Fig. 3.1) also cannot be well described by the FST model.

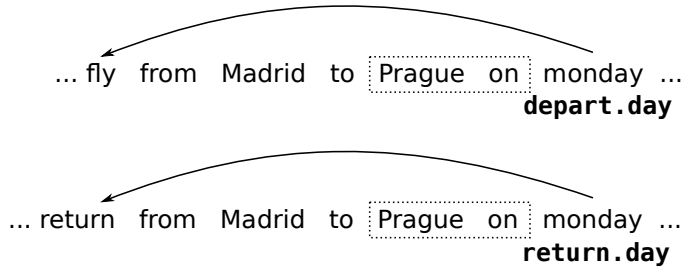

Figure 3.1: An illustration of the long-distance dependency problem of flat models using limited history.

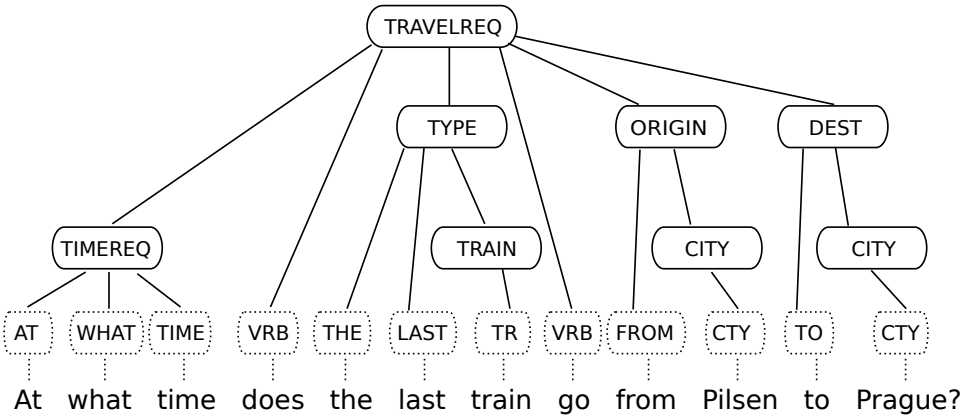

Figure 3.2: Probabilistic context-free grammar for semantics

One possible solution is to use a more complex model, which can be e.g. the *probabilistic context-free grammar* (PCFG) model. An example is shown in Fig. 3.2.

The lexical model $P(C|W)$ can be obtained by using the above mentioned FST model (or by the other models from chapter 2). However, the probabilities of the semantic model $P(C|S_s)$ are computed recursively and they are complex.

Let C be the decoded semantic tree, $P(c, i, j)$ be a probability that the concept c covers sub-concepts (preterminal nodes) from indices i to j. For N preterminal concepts $c_1 \dots c_N$, the probability is $P(C) = P(s, 1, N)$, where s is the root concept of the semantic tree. Let the $P(c \to c_1 \dots c_Q)$ be the probability that the concept c directly generates the sequence of concepts $c_1 \dots c_Q$. Then the *inside-outside* probability is recursively formulated as:

$$P(c,i,j) = \sum_{Q \le j-i+1} \sum_{C_1^Q \in \{c*\}} \sum_{I_0^Q \in \{(i \ldots j)*} P(c \to c_1 \ldots c_Q) \prod_{q=1}^{Q} P(c, I(q-1), I(q)) \tag{3.1}$$

where I_0^Q represents a set of Q integers which splits the sequence $c_i \ldots c_j$ into Q subsequences such that $I(0) = i$ and $I(Q) = j$. The previous formula can be applied for any unrestricted branching, however, in the case of binary branching it is considerably simplified to:

$$P(c,i,j) = \sum_{c_l,c_r} \sum_{t=1}^{i-1} P(c \to c_l c_r) P(c_l, i, t) P(c_r, t+1, j) \tag{3.2}$$

Now, the formula 3.2 is the *inside probability* of the Inside-Outside algorithm which is a modification of the EM algorithm for parameter estimation.

If the fully annotated corpus is available, the parameter estimation is simplified. In this case, the probabilities of the parse path (a sequence of concepts C_1^Q) are computed as:

$$P(C_1^Q) = \prod_{q=1}^{Q} \begin{cases} P(c_q | c_{q-1}, c*) & \text{if } c* \text{ is in semantic model} \\ P(w_q | w_{q-1}, c*) & \text{if } c* \text{ is in lexical model} \end{cases} \tag{3.3}$$

However, the PCFG models suffer from a variety of theoretical and practical problems, such as normalization problems. Also, the recursive nature makes them computationally intractable [You02].

3.3 Vector-state Markov model

The 1-st order Markov model in *FST* was described in section 2.2. Although this model is mathematically simple and easy to train, its major disadvantage is the lack of ability to capture the hierarchy of semantics. This may lead to long dependency problems etc. On the other hand, the *PCFG* introduced in the previous section seems to be a too complex model, especially for training and estimation of the probabilities.

To improve the simple 1-st order model, the *vector-state Markov model* is proposed in [HY05]. It is still a discrete HMM, but each state is actually a stack of pushdown automata with a limited size. This stack consists of semantic concepts. Thus, there is an equivalecy between a PCFG and vector-state Markov model with unlimited stack depth. It is shown in Fig. 3.3. This model is *right branching*. It means that the branches of the semantic tree grow in left-to-right direction.

The probability of a sematnic parse tree $\mathbb{C}$ (which is actually a set of stack of concepts) given the input sentence W can be formalised as follows:

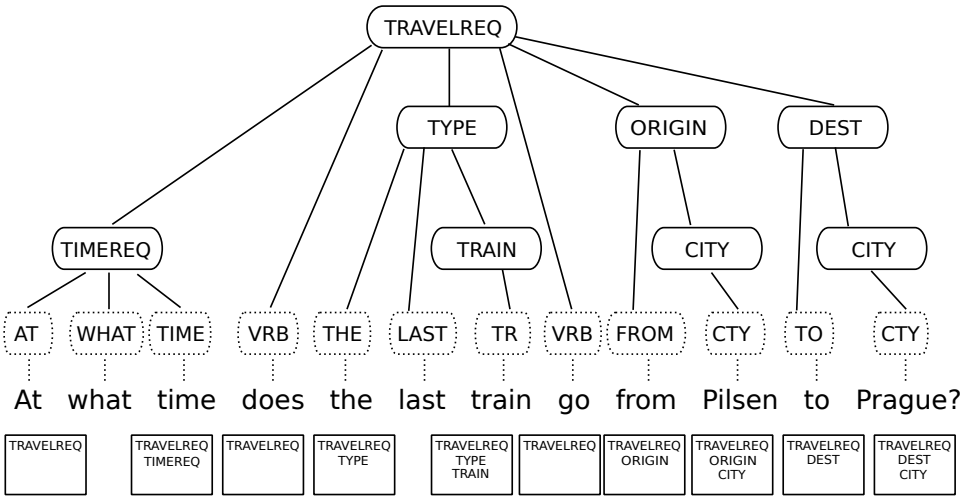

Figure 3.3: A vector-state model and its equivalent tree representation

$$P(N, \mathbb{C}, W) = \prod_{t=1}^{T} P(n_t | W_{1\ldots t-1}, \mathbb{C}_{1\ldots t-1}) \cdot P(c_t[1] | W_{1\ldots t-1}, \mathbb{C}_{1\ldots t-1}, n_t) \cdot \\ \cdot P(w_t | W_{1\ldots t-1}, \mathbb{C}_{1\ldots t}), \tag{3.4}$$

where

- $W_{1\ldots t-1}$ is the word history up to $t-1$,
- $\mathbb{C}_{1\ldots t}$ denotes the history of concepts $(\mathbf{c}_1 \ldots \mathbf{c}_t)$. Each vector state $\mathbf{c}_t$ at position t is a vector (or stack) of D_t semantic concept labels where D_t is the stack depth. In more detail, $\mathbf{c}_t = (c_t[1], c_t[2], \ldots c_t[D_t])$ where $c_t[D_t]$ is the root concept and $c_t[1]$ is the preterminal concept (the concept immediately above the word w_t),
- n_t is the number of stack pop operations and takes values in the range of $\langle 0, \ldots, D_{t-1} \rangle$
- $W_{1\ldots t-1}, \mathbb{C}_{1\ldots t-1}$ denotes the previous parse up to position $t-1$,
- $c_t[1]$ is the new preterminal semantic concept at position t assigned to word w_t.

Each transition of the model is restricted to the following operations, which corresponds with the probabilities from Eq. 3.4: (i) pop n_t concept

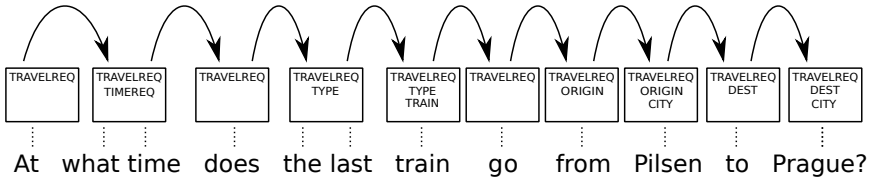

Figure 3.4: A vector-state Markov model

labels from the stack, (i) generating new preterminal concept and (iii) generating a word. Having the n_t which defines the number of semantic concept popped off the stack, the transition from position $t-1$ to t given preterminal concept c_{w_t} for word w_t can be described as:

- pushing the concept to the stack

$$c_t[1] = c_{w_t}, \tag{3.5}$$

- copying the previous stack to the current stack, skipping n_t concepts which have been popped off at the position t

$$c_t[2 \dots D_t] = c_{t-1}[(n_t + 1) \dots D_{t-1}], \tag{3.6}$$

- adujsting the stack depth at position t

$$D_t = D_{t-1} + 1 - n_t \tag{3.7}$$

Depending on value of n_t, the stack can grow as follows. For $n_t = 0$ the stack grows by one semantic concept (no concept has been popped off). The case $n_t = 1$ corresponds to replacing a preterminal concept with new concept. And for $n_t > 1$ the stack is reducing its size by popping off more concepts.

The general model, described in Eq. 3.4, depends on unlimited history of concepts and words. In [HY05] the history is truncated – only the previous semantic concept stack is used and the word history is ignored. The equations are then approximated by

$$\begin{aligned} P(n_t|W_{1\dots t-1}, \mathbb{C}_{1\dots t-1}) &\approx P(n_t|\mathbf{c}_{t-1}) && (3.8)\\ P(c_t[1]|W_{1\dots t-1}, \mathbb{C}_{1\dots t-1}, n_t) &\approx P(c_t[1]|c_t[2 \dots D_t]) && (3.9)\\ P(w_t|W_{1\dots t-1}, \mathbb{C}_{1\dots t}) &\approx P(w_t|\mathbf{c}_t). && (3.10) \end{aligned}$$

The vector-state markov model is shown in Fig. 3.4.

3.4 Hidden vector-state Markov model

In the previous section, the basic vector-state Markov model was introduced. For training such model, a fully annotated corpora with aligned preterminal concepts is required. Then the training is simply a matter of counting of events and a model smoothing. The *hidden vector-state Markov model* is based only on unaligned abstract annotation. It means, that each training sentence is annotated with a semantic concept hierarchy but the association between the preterminal concept layer and the words is not captured (this is, actually, an equivalent of HMM in ASR, where the observations are seen but the states of the model are hidden).

In the next section, two approaches to train such model are introduced. The first one is based on MLE and estimating the model parameters using the Expectation-Minimization algorithm (EM). The second one uses discriminative training. However, there are some prerequisites for both methods. First, there must be a sort of *a priori* knowledge of the domain – the *lexical classes* (see section 3.1). Second, an abstract semantic annotation must be provided for each sentence. These annotations are made by human annotators (see chapter 5).

3.4.1 MLE training of the HVS model

The purpose of training the HVS-based parser is to find the model parameter set $\lambda = \{\mathbb{C}, N\}$ which will result in the maximal likelihood of the training data. This is done by maximizing some objective function $R(\lambda)$. Most commonly used parameter estimation is maximum likelihood estimation (MLE). Given the set of observations $\mathbb{W} = \{W_1, \ldots W_I\}$, the objective function can be formulated as

$$R(\lambda) = f_{ML}(\lambda) = \log \prod_{r=1}^{I} P(W_r, \lambda) = \sum_{r=1}^{I} \log P(W_r, \lambda) \tag{3.11}$$

MLE maximizes the likelihood of the training data. The λ must be recomputed to obtain the model parameters that best explains the data:

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \sum_{r=1}^{I} \log P(W_r, \lambda) \tag{3.12}$$

The training of HVS-based model by MLE is an iterative process. There exists a reestimation formula $f(\cdot)$ such that if $\hat{\lambda} = f(\lambda)$, then the objective function in next step is $R(\hat{\lambda}) > R(\lambda)$. The λ reaches the local maximum when $R(\hat{\lambda}) = R(\lambda)$. A well-known alogorithm for the training is the Baum-Welch algorithm.

The reestimation formulas are:

$$P^*(n|\mathbf{c}') = \frac{\sum_t P(n_t = n, \mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda^k)}{\sum_t P(\mathbf{c}_{t-1} = \mathbf{c}', W|\lambda^k)}, \tag{3.13}$$

$$P^*(c[1]|c[2 \dots D]) = \frac{\sum_t P(\mathbf{c}_t, W|\lambda^k)}{\sum_t P(c_t[2 \dots D] = c[2 \dots D]|W, \lambda^k)}, \tag{3.14}$$

$$P^*(w|\mathbf{c}) = \frac{\sum_t P(\mathbf{c}_t = \mathbf{c}, w_t = w|\lambda^k)}{\sum_t P(\mathbf{c}_t = \mathbf{c}, W|\lambda^k)}. \tag{3.15}$$

MLE makes numbers of assumption which cannot be reached in practice: the global likelihood maximum can be found, the observations are from a known family of distributions and the training data are unlimited. Thus, it is not guaranteed that the MLE trained model will yield optimal results.

3.4.2 Discriminative training of the HVS model

As described in section 3.4.1, the MLE is used for generative statistical training where only the correct models are taken into account during parametr estimation. [ZH09] proposes a method for training of the generative model based on discriminative optimization criterion. That means that not only the likelihood for correct models should be increased, but also the likelihood for incorrect models should be decreased as well.

Accorging to equation 3.4, semantic parsing models are trained to have a maximal likelihood $P(C)$. However, the correlation between higher $P(C)$ and better performance is not perfect.

Let's assume a sentece W with the appropriate set of all possible semantic parse trees of size M and most likely semantic parse tree $\hat{C} = C_j$. The discriminant function is $\log P(W, C)$ and depends on sentence W and the model parameters. A *parse error measure* can be defined as:

$$d(W) = -\log P(W, C_j) + \log \left[\frac{1}{M-1} \sum_{i=1, i \neq j}^{M} P(W, C_i)^\eta \right]^{\frac{1}{\eta}} \tag{3.16}$$

where η is a positive number and is used to select competing semantic parses. For $\eta = 1$, the competing semantic parse term (the second term from Eq. 3.16) becomes an average of all competing parse tree probabilities. When $\eta \to \infty$, the term is $\max_{i, i \neq j} P(W, C_i)$, which is the score of the top competing semantic parse tree result. The resulting $d(W) \leq 0$ implies a correct decision, $d(W) > 0$ implies a classification error.

The loss function is defined as:

$$\ell(W) = \text{sigmoid}(d(W)) \tag{3.17}$$

where sigmoid function is used to normalize $d(W)$ in range $(0, 1)$ and is defined as:

$$\text{sigmoid}(x) = \frac{1}{1 + \mathrm{e}^{-\gamma x}} \tag{3.18}$$

where γ is a constant that controlls the slope of the sigmoid function. The empirical loss is expressed as:

$$L_0(\lambda) = \frac{1}{I} \sum_{j=1}^{I} \sum_{i=1}^{M} \ell_i(W_j, \lambda) = \int \ell(W, \lambda) \mathrm{d}P_I \tag{3.19}$$

where I is the number of samples from the training set $\{W_1, \ldots, W_I\}$ and P_I is the empirical discrete probability measure defined on the training data set. The expected loss is defined as:

$$L(\lambda) = E_W\{\ell(W, \lambda)\} \tag{3.20}$$

In discriminative HVS model training, the goal is to minimize the expected loss over the training samples. The model is trained to separate the correct parse from the incorrect parses. The trained model is then used to parse the training sentences again and the training procedure repeats. This approach is based on the generalized probabilistic descend algorighm [ZH09].

The model parameters are updated sequentially by the formula:

$$\lambda^{k+1} = \lambda^k - \epsilon^k \nabla \ell(W_i, \lambda^k) \tag{3.21}$$

where ϵ^k is the step size and ∇ is the gradient.

The update formulas (see section 3.4.1 for comparison) can be then defined as:

$$\begin{aligned} (\log P(n|\mathbf{c}'))^* = \log P(n|\mathbf{c}') - \epsilon\gamma\ell(d_i)(1 - \ell(d_i)) \times \\ \times \left[-I(C_j, n, \mathbf{c}') + \sum_{i, i \neq j} I(C_i, n, \mathbf{c}') \frac{P(W_i, C_i, \lambda)^\eta}{\sum_{i, i \neq j} P(W_i, C_i, \lambda)^\eta} \right] \end{aligned} \tag{3.22}$$

$$\begin{aligned} (\log P(c[1]|c[2 \ldots D]))^* = \log P(c[1]|c[2 \ldots D]) - \epsilon\gamma\ell(d_i)(1 - \ell(d_i)) \times \\ \left[-I(C_j, c[1], c[2 \ldots D]) + \sum_{i, i \neq j} I(C_i, c[1], c[2 \ldots D]) \frac{P(W_i, C_i, \lambda)^\eta}{\sum_{i, i \neq j} P(W_i, C_i, \lambda)^\eta} \right] \end{aligned} \tag{3.23}$$

$$(\log P(w|\mathbf{c}))^* = \log P(w|\mathbf{c})) - \epsilon\gamma\ell(d_i)(1-\ell(d_i))\times$$
$$\times \left[-I(C_j, w, \mathbf{c}) + \sum_{i,i\neq j} I(C_i, w, \mathbf{c}) \frac{P(W_i, C_i, \lambda)^\eta}{\sum_{i,i\neq j} P(W_i, C_i, \lambda)^\eta} \right] \tag{3.24}$$

where

- $I(C_i, n, \mathbf{c}')$ is the number of operations of popping up n semantic tags from the stack at the current vector state $\mathbf{c}'$ in the C_i parse tree,
- $I(C_i, c[1], c[2 \dots D])$ denotes the number of operations of pushing the semantic concept $c[1]$ at the current vector state $c[2 \dots D]$ in the C_i parse tree,
- and $I(C_i, w, \mathbf{c})$ is the number of times of emittin the word w at the state $\mathbf{c}$ in the parse tree C_i.

The full derivations of the updating formulas can be found in [ZH09].

3.4.3 Extended HVS parser

[Jur07] introduced some extensions to the base HVS semantic parser, namely the left-right-branching parsing and input parametrization.

Left-right-branching parsing

The original semantic model of HVS parser (see Eq. 3.4) allows to push only one concept $c_t[1]$ to the stack. To enable either pushing one concept or no concept, a new hidden variable *push* is inserted into the model. The resulting semantic model is then formulated as:

$$P(C) = \prod_{t=1}^{T} P(n_t|\mathbf{c}_{t-1}) \cdot P(push_t|\mathbf{c}_{t-1}) \cdot \begin{cases} P(\mathbf{c}_t = \mathbf{c}_{t-1}|\mathbf{c}_{t-1}) = 1 & \text{if } push_t = 0 \\ P(c_t[1]|c_t[2, \dots D]) & \text{if } push_t = 1 \end{cases} \tag{3.25}$$

where $push_t$ takes values 1 for pushing a new concept to the stack or 0 for pushing no concept to the stack (keeping the concept vector unchanged from the previous vector state $t-1$). The implementation of such model adds the interim stack c_t^i.

Another modification of the base HVS model described in [Jur07] is the posibility of pushing two concepts at the same time. It has been experimentaly verified that pushing more than two concepts does not significantly affect the results. Moreover, this limitation keeps the model simple.

In such extension, the $push_t$ variable takes values from $\{0, 1, 2\}$ and the semantic model formula is defined as:

$$P(C) = \prod_{t=1}^{T} P(n_t|\mathbf{c}_{t-1}) \cdot P(push_t|\mathbf{c}_{t-1}) \cdot \\ \cdot \begin{cases} P(\mathbf{c}_t = \mathbf{c}_{t-1}|\mathbf{c}_{t-1}) = 1 & \text{if } push_t = 0 \\ P(c_t[1]|c_t[2,\dots D]) & \text{if } push_t = 1 \\ P(c_t[1]|c_t[2,\dots D]) \cdot P(c_t[2]|c_t[3,\dots D]) & \text{if } push_t = 2 \end{cases} \quad (3.26)$$

Eq. 3.27 contains an approximation of the probability of pushing two concepts at the same time to improve the model robustness:

$$P(c_t[1,2]|c_t[3\dots D]) \approx P(c_t[1]|c_t[2,\dots D]) \cdot P(c_t[2]|c_t[3,\dots D]) \quad (3.27)$$

Input Parametrization

In the basic HVS model, the input W is in a form of word sequence. However, [SJM07] proposed to extend the input with some additional information such as lemma or morphological tags.

Using the lemmatized input means that the input word w_i is replaced by its lemma. The reduction of the vocabulary consequently reduces the number of parameters to be estimated and it improves the model robustness. Nevertheless, the discrimination ability of such model is decreased.

Additionaly, the full morhological analysis results can be used to avoid the lower model discriminability by lemmatization. Thus, the *input feature vector* $\mathbf{f}_t$ is introduced:

$$\mathbf{f}_t = f_t[1], \dots f_t[N_F] \quad (3.28)$$

where N_F is the number of the features. The feature vector contains i.e. $f_t[1] =$ *lemma*, $f_t[2] =$*morph. tag*, etc. Using this vector, the lexical model of the HVS parser can be generalized as:

$$P(W|S) = \prod_{t=1}^{T} P(\mathbf{f}_t|\mathbf{c}_t) \quad (3.29)$$

To achieve higer model robustnes, the following approximation is used in [Jur07]:

$$P(\mathbf{f}_t|\mathbf{c}_t) \approx \prod_{j=1}^{N_F} P(f_t[j]|\mathbf{c}_t) \quad (3.30)$$

Thus, the lexical model including feature vectors is formulated as:

$$P(W|S) = \prod_{t=1}^{T} \prod_{j=1}^{N_F} P(f_t[j]|\mathbf{c}_t) \quad (3.31)$$

3.5 Context-based Stochastic Parser

The semantic parser presented in [Kon09] is a hybrid stochastic semantic parser. The training data consists of annotated sentences, where the preterminal concepts (or *lexical classes*) are aligned to the input words (this annotation methodology is similar to vector-state parser introduced in section 3.3). The model parameters are then estimated using MLE:

$$P(N \to \alpha|N) = \frac{\text{Count}(N \to \alpha)}{\sum_\gamma \text{Count}(N \to \gamma)} \tag{3.32}$$

where $N \to \alpha$ means that in the data, the non-terminal N is rewritten to α. Moreover, the word context of each tree node is taken into account. The context is defined as the words before and the words after the span of a subtree. Then the probability of a context given a nonterminal is estimated by MLE as:

$$P(w|N) = \frac{\text{Count}(w, N) + \lambda}{\sum_i \text{Count}(w_i, N) + \lambda V} \tag{3.33}$$

where w is the current context word of nonterminal N, w_i are all context words, λ is the smoothing constant and V is the estimate of the vocabulary size. For the estimation of the *theme* probability (in this model the *theme* is the root concept of the sematnic tree), there is an additional formula:

$$P(w|S) = \frac{\text{Count}(w, S) + \kappa}{\sum_i \text{Count}(w_i, S) + \kappa V} \tag{3.34}$$

where S is the root concept (the *theme*), w_i are the words of the sentence and κ is the smoothing constant.

Once the model is trained, the parser performs two steps. First, the shallow parsing algorithms (see section 3.1) are used to identify lexical classes. In this system, the shallow parser is based on CFG for generic lexical classes such dates, time, numbers, etc. and on the vocabulary methods for proper names etc. Second, a stochastic bottom-up chart parser is used to create parse trees and to compute probabilites as follows:

$$P(T) = \sum_i P(w_i|N)P(N \to A_1 \dots A_k|N) \prod_j P(T_j), \tag{3.35}$$

where N is the top nonterminal of the subtree T, $A_1 \dots A_k$ are terminals or nonterminals which are expanded from N and T_j is a subtree having the nonterminal A_i on the top.

Then the best parse is selected using the best probability:

$$P(\hat{T}) = \underset{i}{\text{argmax}}\, P(S_i) \prod_j P(w_j|S)P(T_j) \tag{3.36}$$

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

3.6 Other approaches to semantic parsing

3.6.1 Clustering approach

[HY06a] proposed a novel algorithm for semantic decoding in SLU systems. This approach differs from previously mentioned either rule-based or purely statistical systems. Both systems treat semantic decoding as a classical parsing problem. An alternative would be to treat the decoding as a straightforward pattern recognition problem.

This approach requires relatively few training data (less then the HVS model, sec. 3.3) and the data are annotated only at sentence level. The key idea is to cluster sentences into classes and then assign a single semantic annotation to each class. In the decoding process, the input sentence is assigned to the class which it most closely matches.

Given a set of N training sentences $\{S_1, \ldots, S_N\}$, the Y-clustering algorithm groups the sentences into Y classes. From each class, one sentence is selected as so called *template*. The basic algorithm is similar to K-means clustering.

1. *Initialization.* Randomly select Y different sentences as the initial templates $\{T_1, \ldots, T_Y\}$.

2. *Clustering.* Assing each sentence S_i to the closest template m^*

$$m^* = \underset{m}{\operatorname{argmax}}\{d(S_i, T_m)\} \qquad \text{for} \quad m = 1, \ldots, Y \tag{3.37}$$

   where $d(S, T)$ is a distance between sentence S and template T.

3. *Template regeneration.* For each class, select new template $T = S^T$ which yelds the highest within-class similarity $\mathcal{D}$:

$$S^T = \underset{S_i}{\operatorname{argmax}}\{\mathcal{D}(S_i)\} \qquad \text{for} \quad i = 1, \ldots, H \tag{3.38}$$

   where H is the number of sentences in the class and $\mathcal{D}(S_i)$ is the total similarity between sentence and all members of the class, i.e.

$$\mathcal{D}(S) = \sum_{h=1}^{H} d(S, S_h) \tag{3.39}$$

4. If the termination criteria is not satisfied, return to Step 2. For instance, the process can be terminated if the newly created classes are same as the classes from the previous iteration.

When implementing the algorithm described above, two main issues must be solved: the distance measurement and the method of controlling the generated classes.

The similarity between sentences depends on the words within. Every word is assigned a *saliency* (Eq. 3.40) to ensure that the words with key information are weighted more heavily than the less relevant words. The saliency of the word represents how important the word is for distinguishing classes.

Assume that the class contains H sentences and α sentences contain the word w. Additionaly, assume that there are M classes and among them β classes containing the word w. Then the saliency of the w in that class is defined as

$$\mathcal{I}(w) = \sqrt{\frac{\alpha}{H}\left(1 - \frac{\beta}{M}\right)} \tag{3.40}$$

The value of $\mathcal{I}(w)$ ranges from 0 to 1. It is actually a form of mutual information between the within class frequency and inter-class frequency. Higher saliery means that the word will ocure more frequently in that class and less frequently in other ones.

For computing similarity between a sentence S and the template T, the DTW algorithm is used for aligning on indetical words. Given this alignment, assume L words in the sentence S, K words in the template T with saliency $\mathcal{I}_i$ for $i = 1, \ldots, K$, then the similarity is given by:

$$d(S,T) = \sqrt{\frac{J}{L}\left(\frac{\sum_{j=1}^{J}\mathcal{I}(w_j)}{\sum_{i=1}^{K}\mathcal{I}_i}\right)} \tag{3.41}$$

Controlling the process of generating classes by setting a similarity threshold and the parameters of clustering are described more precisely in [HY06a].

Semanic decoding using the Y-clustering approach is performed in the following steps: First, each sentence in the corpus is preprocessed in such a way that any lexical class is replaced by its class name. Then the clustering is performed to generate the templates. Each template then has the associated slot/values given by the position in the template. For example, the sentence

```
Show me flights from London to Paris
```

has following template:

```
Show me flights from city_name to city_name
Slots/Values: FROMLOC.CITY = T(5)
              TOLOC.CITY = T(7)
```

Decoding the sentence uses the same algorithm as training (DTW, computing similiarity) and the most similar template is then chosen as the corresponding semantic class. Finally, the slots are filled with the realizations from the sentence.

3.6.2 Kernel-based statistical methods

In traditional machine learning methods, the input is represented by a set of features (feature vector). But some more complicated input structures, such as trees, cannot be easily expressed by such feature vectors, because the structural relations are lost when the data are reduced to a set of features. To aviod this, all possible sequences of features must be taken into account, which makes the the algorithm computationally expensive.

An alternative to feature-based methods are the kernel-based methods [KM06]. A kernel is a similarity function K over the domain X which maps a pair of objects $x, y \in X$ to their similarity score $K(x, y)$, ranging from 0 to ∞.

[Kat09] defines a kernel between two strings as the amount of common subsequences between them. Formally, let s be a string from the alphabet Σ and $|s|$ the length of the string and a substring is a continuous subset $s[i \dots k]$. We call a subsequence u of s, if there exists an index sequence $\mathbf{i} = (i_1 i_2 \dots i_{|u|})$ with $1 \leq i_1 \leq \cdots \leq i_{|u|} \leq |s|$ and it is written as $s[\mathbf{i}]$ for short.

The *span* is the distance between the first and the last index of $\mathbf{i}$, such $span(i) = i_{|u|} - i_1 + 1$. The function $\Phi_u(s)$ is a number of multiple unique index sequences $\mathbf{i}$ of string s and it is formulated as:

$$\Phi_u(s) = \frac{1}{\lambda^{|u|}} \sum_{\mathbf{i}:s[\mathbf{i}]=u} \lambda^{span(\mathbf{i})} \tag{3.42}$$

where $\lambda \in (0, 1\rangle$ is a decay factor. Finally, the kernel between two strings is defined as:

$$K(s, t) = \sum_{u \in \Sigma^*} \Phi_u(s)\Phi_u(t) \tag{3.43}$$

and normalized kernel to remove any bias due to different string lengths is then formulated as:

$$K_{normalized}(s, t) = \frac{K(s, t)}{\sqrt{K(s, s)K(t, t)}} \tag{3.44}$$

The semantic parsing is then performed by using the notion of a semantic derivation of an NL sentence. In [Kat09], the task is presented as finding the most probable semantic derivation of an NL sentence which is determined using an extended version of Earley's algorithm for context-free grammar parsing. The kernel-based SVM (support vector machines) are used as the classifiers for computing the probability of the parse tree derivation.

Chapter 4

System evaluation

4.1 Evaluation techniques

Although the SLU/NLU systems mostly do not appear as standalone applications and they are incorporated into e.g. dialog systems, there is a reasonable need of independent evaluation of the system itself. Having a semanticaly annotated corpus made by human annotators, we can compare it to the restults obtained from the semantic analysis system using some criteria. Therefore, various types of measures are introduced in this section.

4.1.1 Exact match

The simplest criterion is the exact match criterion. Let C_R be the *reference* parse tree (made by human annotator) and let C_P be the tree produced by the parser. Then the exact match criterion is a binary function resulting 1 iff the C_P matches exactly the C_R and 0 otherwise. The exact match means that both trees has the same structure and labels of concepts and the preterminal concepts cover the same positions of words in the input sentence.

Since the output of the parser can differ only e.g. in one concept or even in preterminal concept positions, such tree is automatically discarted with 0. Therefore we need a more finer measure which would penalize the output depending on its distance from the reference parse tree.

The *semantic accuracy* presented in [Jur07] is actually the exact match measure, but it does not take into account the word alignment of the preterminal concepts and uses only the concept structure. Then the accuracy is computed as

$$SAcc = \frac{E}{N} \tag{4.1}$$

where E is the number of hypothesis semantics that exactly match and N is the number of all recognized semantics.

4.1.2 PARSEVAL

The standard measures used in PARSEVAL are *precission* (P), *recall* (R) and *F-measure* (F). Let *correct concept* be a concept which spans the same words both in reference and hypothesis tree. The the values are computed as follows:

$$P = \frac{\#\text{ of correct concepts in } C_P}{\#\text{ of concepts in } C_P} \tag{4.2}$$

$$R = \frac{\#\text{ of correct concepts in } C_P}{\#\text{ of concepts in } C_R} \tag{4.3}$$

$$F = \left(\frac{\alpha}{P} + \frac{1-\alpha}{R}\right)^{-1} \tag{4.4}$$

The parameter $\alpha \in \langle 0, 1\rangle$ is used to distribute the preference between the recall and the precision.

4.1.3 Tree edit distance

The tree edit distance algorithm computes the minimum numbers of substitions, insertions and deletions required to transform one tree into another. For semantic analysis system outputs its able to measure only distance between abstract semantic trees without the relation to the words of input sentence (therefore it is called *concept accuracy* in [Jur07]). The accuracy is then defined as:

$$CAcc = \frac{N - S - D - I}{N} \tag{4.5}$$

where N is number of concepts in the reference semantics tree, S is the number of substitutions, D is the number of deletions and I is the number of insertions.

Chapter 5

Data annotation

All semantic analysis systems presented in this thesis heavily depend on annotated data. Annotated data are data enriched with some additional semantic information, such as *abstract annotation* [HY05], semantic classes of words [You02] or abstract annotation aligned to the words of the input sentence [MGJ+09], as shown in Fig. 5.1.

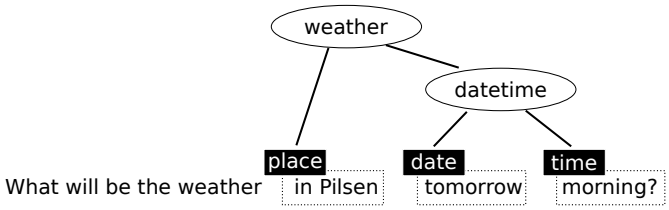

Figure 5.1: An example of abstract semantic annotation with aligned lexical classes.

The annotations are created by human *annotators*. Given a certain annotation methodology, the annotator uses his or her semantic and domain knowledge and associates an annotation to each sentence from the training data. Nevertheless, it is very common that two annotators create a different annotation for the same sentence (one of the reason can be e.g. semantic ambiguity of such sentence). To achieve more independency on the approach of one annotator, the *inter-annotator agreement* measuring is often used [Kon09].

Since the annotation process is very time demanding, some additional steps can be performed while creating the annotated corpora. A methodology described in [Kon09] is called *bootstrapping*. This is an iterative process of training unsupervised or parital-supervised models. At the begining, only a small part of the data is annotated by the human. Using this data, the initial model is trained. Then the trained model runs over the remaining

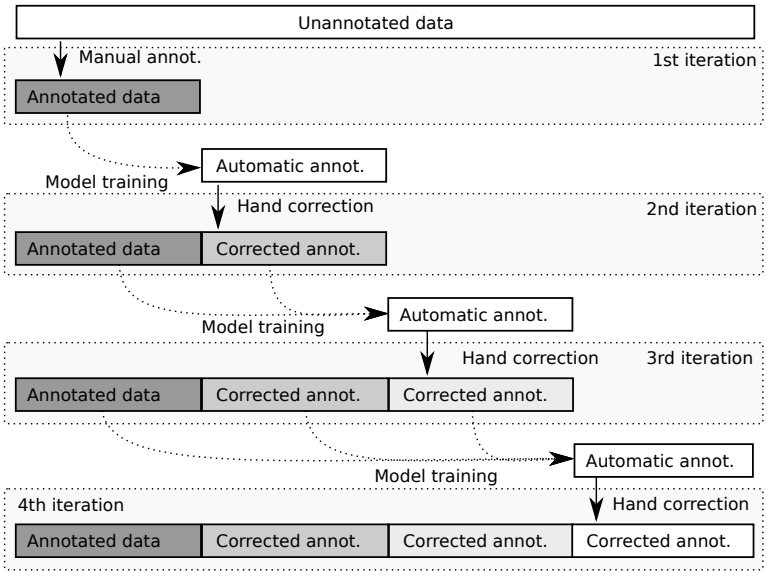

Figure 5.2: An example of bootstrapping annotation methodology.

data, from which a part is corrected by human annotators. The model is then re-trained again and the process continues. An example is shown in Fig. 5.2. Another automatic semanic annotation approch is presented in the LUNA project [MMG08]. In this project, the rule-based methods are used for identification of proper names and simple semantic concepts. However, this preprocessing depends on language and domain.

There are some recommendations for creating the annotated data [Kon09]. The most important requirement is very detailed annotation manual and a very simple annotation methodology (but still, the methodology must be able to capture all required semantic relations in the data). Moreover, an efficient annotation tool is required to decrease the time of annotating.

There are some common methodologies used for semantic annotation. Obviously, the final format heavily depends on the target system, but there are still some common features. The *bracketing notation* is used e.g. in [SMSM97] or [MBIS94]. For example, the input sentence *"I need to go to Pilsen tomorrow morning"* would have the following semantic annotation:

```
TRAVELREQ(TOPLACE(
    CITY,
    DATETIME(DATE, TIME))
)
```

This corresponds to the abstract semantic annotation in [You02] because there is no specific connection between the concepts from the annotation

and the words within the sentence. However, such annotaion can be easily extended to the aligned semantic annotation by asigning the concepts to the words, ie:

```
TRAVELREQ(TOPLACE(
    CITY(Pilsen),
    DATETIME(DATE(tomorrow), TIME(morning)))
)
```

Apparently, the bracket notation is just a formalism of describing a tree structure. Although creating such annotation for the sentence is a straightforward solution, it still requires more annotator effort (ie. checking the bracket pairing etc.). Therefore, some systems use more efficient tools for creating semantic annotation to avoid the syntax invalidity etc. [IM07].

Chapter 6

Existing systems and corpora

In this chapter, an overview of existing NLU systems and corpora is presented. Instead of a comphrehensive survey, some well documented and commonly referenced systems has been chosen.

6.1 Corpora

6.1.1 ATIS

The ATIS (Air Travel Information Service) corpus contains a travel information data. There are more versions of the ATIS corpus: ATIS-2 and ATIS-3. Originally, these corpora contained only transcriptions from spoken language. The abstract semantic annotations were derived semi-automatically using the SQL queries provided in ATIS-3. A set of 30 domain-specific lexical classes were extracted from the ATIS database. For creating the test set, postprocessing was required to extract relevant slot/values and transform them into a compatible format [HY06b].

6.1.2 DARPA

The DARPA Communication data contain utterance transcriptions and the semantic parse results from the rule-based Phoenix parser [WI94]. The abstract semantic annotation produced by the parser were hand-corrected. The data consists of 38408 utterances total. After removing context dependend utterances such "Yes", "No, thank you", etc., the final set consists of 12702 utterances. The corpus is publicly available without the semantic annotations.

6.1.3 Other corpora

The LUNA project is an attempt to create a multilingual spoken language understanding module. The corpus contains semantic annotation for three

languages (French, Italian an Polish). The Polish corpus was collected from the Warsaw City Transportation information center where people can get various information related to routes and time schedules of public transport etc. An automatic annotation process of this corpus is described in [MMG08]. The Italian part of this project consists of spontaneous dialogues recorded in the call center of the help desk facility of the Consortium for Information Systems of Piemont (CSI Piemonte), thus its domain is focused on computer related dialogues. The active annotation of this corpus is presented in [CRR08].

The MEDIA project aims to evaluate SLU approches in French language. The domain is targeted at hotel room booking and related tourist information. The corpus was recorded using WOZ[1] system simulation. The semantic annotation procudure is described in [BMRA+05].

6.2 Existing systems

6.2.1 HVS Parser

A hidden vector-state model (see section 3.3) was presented in [HY05]. The system was tested on the ATIS and DARPA corpora, recently the system was also used for semantic extraction from bioinformatics corpus Genia [ZH09]. The first model traning was based on MLE (see section 3.4.1), however, the discriminative training has also been proposed (see section 3.4.2).

Extended HVS Parser

An extension of the basic HVS Parser has been developed in the work of [Jur07]. The improvement is achieved by extending the lexical model and by allowing left-branching (see section 3.4.3). The system was tested on Czech human-human train timetable corpus and it is public available.

6.2.2 Scissor

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. It uses the state-of-art syntactic parser for English, the Collin's parser [Col97]. The parse tree consists of syntactic nodes augmented with semantic concepts. Some concepts (reffered to as predicate) take an ordered list of arguments from sub-concepts. Also, *null* concepts, which do not correspond to any semantic information, are introduced.

[1] WOZ = Wizard-Of-Oz. In this way, the user/speaker believes he or she is talking to the machine, whereas in fact he is talking to a human being who simulates the behaviour of the dialogue server.

The training of the system is performed in the following steps. First, the corpus is parsed by Collin's parser. Then the semantic labels for individual words are added to the POS nodes in the tree. Finally, the rest of semantic concepts is added in the bottom-up manner. This semantic annotation must be done manually and heavily depends on the domain knowledge and requires a robust syntactic parser. The system is described in detail e.g. in [GM05], [RM06] or [Moo07].

6.2.3 Wasp

Wasp (Word Alignment-based Semantic Parsing) is based on statistical machine translation (SMT). The method is trained on corpora consisting of a natural language data and appropriate meaning representation language (MRL). Wasp requires no prior knowledge of the natural language syntax, although it assumes that an unambiguous, context-free grammar (CFG) of the target MRL is available [WM06]. First, the words are aligned to corresponding semantic concepts using the GIZA++ system [ON03]. Then the probabilistic parser and synchronous CFG is used for creating parse tree.

6.2.4 Krisp

Krisp (Kernel-based Robust Interpretation for Semantic Parsing) uses support vector machines with string kernels to build semantic parsers that are more robust in the presence of noisy training data. The string kernels are introduced in section 3.6.2. In particular, Krisp uses the string kernel to classify substrings in a natural language sentence. Like Wasp, it uses production rules in the MRL grammar to represent semantic concepts, and it learns classifiers for each production. Learning the parameters of the system is an iterative process, in each step the positive and negative examples are collected for training SVM. During semantic parsing, Krisp uses these classifiers to find the most probable semantic derivation of a sentence [KM06].

6.2.5 Other systems

The system for *semantic role labeling* described in [GJ00] indetifies the semantic roles. The output of the system is a sentence with labeled smenatic roles such as *agent* or *patient*, or more domain-specific semantic roles such *speaker*, *topic*, etc. Although the system applies statistical techniques, the main disadvantage of this system is the use of full syntactic parse, which is not suitable for free-word-order languages with very complicated grammars, such as the Czech langauge.

Another system for stochastic semantic analysis presented in [BMB05] uses a scheduling data corpus, the English Spontaneous Speech Task (ESST).

This is an appointment scheduling task where two people speaking different languages try to schedule a meeting. This approach is very similar to flat-concept parser (see section 2.2) because it uses first-order HMM, but the training stage depends again on the rule-based parser. However, a more general method for context definition has been proposed in this approach. Instead of introducing data-dependent context classes, the *contextual observations* are introduced.

The LingvoSemantics system is based on context-based stochastic parser introduced in section 3.5. The corpus consists of 20292 written user queries focused on internet search in natural Czech language (weather forecast, bus and train information, etc.). The semantic formalism is described by an *annotation scheme* which captures the possible dependences among semantic concepts in the abstract annotation. The training data were annotated according to these schemes. The project is developed in Java under GPL licence. Also the annotation editor is freely available.

In [WDA05], 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.

Chapter 7

Conclusion and future work

This thesis presented overview of the current state-of-the-art methods for statistical semantic analysis. In comparison to the expert based systems, the main advantage of stochastic approaches is the ability to train the model from data. Furthermore, systems based on statistical models can be easily ported to other domains. However, the amount of the annotation effort must be also taken into account when developing the stochastic semantic analysis system.

The baseline system for the dissertation thesis is the work of [Kon09]. However, there is a considerable possibility to advance the methods for semantic analysis in order to improve the results of the system. Namely, it can be e.g. better methods for lexical class analysis, incorporating negative examples into the statistical model or incorporate discriminative training. The task is summarised below.

7.1 Aims of the PhD thesis

- Continue in the work of [Kon09], study the system, obtain additional semantically annotated data using the methodology from chapter 5 and measure the performance of the system.
- Compare the system with another existing systems. Explore the possibilites to obtain a standard semantic corpus (ie. ATIS) and evaluate the system using this data.
- Propose and evaluate novel methods in order to improve the semantic analysis system. Focus on robust processing of faulty data. Consider specific properties of the Czech language.
- Experiment with the use of the developed system for semantic web searching using the natural language.

Bibliography

[All95] James Allen. *Natural language understanding (2nd ed.)*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1995.

[BMB05] Christiane Beuschel, Wolfgang Minker, and Dirk Bühler. Hidden markov modeling for semantic analysis – on the combination of different decoding strategies. *Internation Journal of Speech Technology*, 8(3):295–305, 2005.

[BMRA+05] H. Bonneau-Maynard, S. Rosset, C. Ayache, A. Kuhn, D. Mostefa, and the Media consortium. Semantic annotation of the French Media dialog corpus. In *Eurospeech 05, Lisbon*, 2005.

[Col97] Michael Collins. Three generative, lexicalised models for statistical parsing. In *In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, 1997.

[CRR08] Kepa Joseba Rodriguez Christian Raymond and Giuseppe Riccardi. Active annotation in the luna italian corpus of spontaneous dialogues. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA).

[GJ00] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. In *ACL*, 2000.

[GM05] Ruifang Ge and Raymond J. Mooney. A statistical semantic parser that integrates syntax and semantics. In *In Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 9–16, June 2005.

[HY05] Yulan He and Steve Young. Semantic processing using the hidden vector state model. *Computer Speech & Language*, 19(1):85–106, 2005.

[HY06a] Yulan He and S. Young. A clustering approach to semantic decoding. In *9th International Conferennce on Spoken Language Processig (Interspeech 2006 — ICSLP)*, pages 17–21, Pittsburgh, USA, Sept 2006.

[HY06b] Yulan He and Steve Young. Spoken language understanding using the hidden vector state model. *Speech Communication*, 48(3-4):262–275, 2006.

[IM07] Habernal I. and Konopik M. Jaae: the java abstract annotation editor. In *In INTERSPEECH-2007*, pages 1298–1301, 2007.

[IP07] E. Iosif and A. Potamianos. A soft-clustering algorithm for automatic induction of semantic classes. In *Interspeech-07*, pages 1609–1612, Antwerp, Belgium, August 2007.

[JL08] Minwoo Jeong and Gary Geunbae Lee. Practical use of non-local features for statistical spoken language understanding. *Computer Speech and Language*, 22(2):148–170, April 2008.

[JM00] Daniel Jurafsky and James 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.

[Jur07] F. Jurčíček. *Statistical approach to the semantic analysis of spoken dialogues.* PhD thesis, University of West Bohemia, Faculty of Applied Sciences, 2007.

[Kat09] Rohit J. Kate. *Learning for Semantic Parsing with Kernels under Various Forms of Supervision.* PhD thesis, Department of Computer Sciences, University of Texas at Austin, August 2009.

[KM06] Rohit J. Kate and Raymond J. Mooney. 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. Association for Computational Linguistics.

[Kon09] M. Konopík. *Hybrid Semantic Analysis.* PhD thesis, University of West Bohemia, Faculty of Applied Sciences, 2009.

[MBIS94] Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz. Hidden understanding models of natural language. In *Proceedings of the 32nd annual meeting on Association*

for Computational Linguistics, pages 25–32, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[MGJ+09] F. Mairesse, M. Gasic, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young. Spoken language understanding from unaligned data using discriminative classification models. In *In ICASSP 2009, International Conference on Acoustics, Speech, and Signal Processing, Taipei*, 2009.

[MMG08] Agnieszka Mykowiecka, Malgorzata Marciniak, and Katarzyna Glowińska. Automatic semantic annotation of polish dialogue corpus. In *TSD '08: Proceedings of the 11th international conference on Text, Speech and Dialogue*, pages 625–632, Berlin, Heidelberg, 2008. Springer-Verlag.

[Moo07] Raymond J. Mooney. Learning for semantic parsing. In *Computational Linguistics and Intelligent Text Processing: Proceedings of the 8th International Conference, CICLing 2007*, pages 311–324, Berlin, Germany, February 2007. Springer.

[NSP06] Le-Minh Nguyen, Akira Shimazu, and Xuan-Hieu Phan. Semantic parsing with structured svm ensemble classification models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 619–626, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[ON03] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[PMS+01] Ferran Pla, Antonio Molina, Emilio Sanchis, Encarna Segarra, and F. García. Language understanding using two-level stochastic models with pos and semantic units. In *TSD '01: Proceedings of the 4th International Conference on Text, Speech and Dialogue*, pages 403–409, London, UK, 2001. Springer-Verlag.

[PWH+04] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA, 2004.

[RM06] Ruifang Ge Raymond and J. Mooney. Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL*,

pages 263–270, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[RR07] Ch. Raymond and G. Riccardi. Generative and discriminative algorithms for spoken language understanding. In *Interspeech-07*, pages 1605–1608, Antwerp, Belgium, August 2007.

[SJM07] J. Svec, F. Jurčíček, and L. Müller. Input parameterization of the hvs semantic parser. *Lecture Notes in Artificial Intelligence*, pages 415–422, 2007.

[SMSM97] R. Schwartz, S. Miller, D. Stallard, and J. Makhoul. Hidden understanding models for statistical sentence understanding. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 2*, page 1479, Washington, DC, USA, 1997. IEEE Computer Society.

[SP03] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[TKSDM03] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

[WDA05] Ye-Yi Wang, Li Deng, and Alex Acero. An introduction to statistical spoken language understanding. *IEEE Signal Processing Magazine*, 22(5):16–31, 2005.

[WI94] Wayne Ward and Sunil Issar. Recent improvements in the cmu spoken language understanding system. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 213–216, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[WM06] Yuk Wah Wong and Raymond J. Mooney. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[You02] Steve Young. The statistical approach to the design of spoken dialogue systems. Technical report, University of Cambridge: Department of Engineering, Cambridge, UK, 2002.

[ZH09] Deyu Zhou and Yulan He. Discriminative training of the hidden vector state model for semantic parsing. *IEEE Trans. on Knowl. and Data Eng.*, 21(1):66–77, 2009.