University of West Bohemia
Faculty of Applied Sciences

Semantic Web Search Using Natural Language

Ivan Habernal

Doctoral Thesis
Submitted in partial fulfillment of the requirements
for a degree of Doctor of Philosophy
in Computer Science and Engineering

Supervisor: Professor Václav Matoušek
Department of Computer Science and Engineering

Plzeň, 2012

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Vyhledávání v Sémantickém webu použitím přirozeného jazyka

Ing. Ivan Habernal

disertační práce
k získání akademického titulu doktor v oboru
Informatika a výpočetní technika

Školitel: Prof. Ing. Václav Matoušek, CSc.
Katedra informatiky a výpočetní techniky

Plzeň 2012

Prohlášení

Předkládám tímto k posouzení a obhajobě disertační práci zpracovanou na závěr doktorského studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji tímto, ze tuto práci jsem vypracoval samostatně, s použitím odborné literatury a dostupných pramenů uvedených v seznamu, jenž je součástí této práce.

V Plzni dne 31. března 2012 Ing. Ivan Habernal

Abstract

This thesis presents a complete end-to-end system for the Semantic Web search using a Natural Language. The system is placed into the context of recent research in Information Retreival, Semantic Web, Natural Language Understanding, and Natural Language Interfaces. The key feature of Natural Language Interfaces is that users can search for the required information by posing their questions using natural language instead of e.g. filling web forms.

The developed system uses the Semantic Web technologies in both traditional and new forms. The idea of the Semantic Web has brought many interesting concepts into domain modeling and data sharing. Furthermore, the development in Natural Language Interfaces to Semantic Web has shown that bridging the gap between the Semantic Web and Natural Language Interfaces can uncover new research challenges.

The main contributions of this thesis are as follows. First, a unique formalism for capturing a natural language question semantics, based upon Semantic Web standards, was proposed. Second, the statistical model for the semantic analysis based upon supervised training was developed. Third, the evaluation of the fully functional end-to-end system with a real data and real queries was conducted.

The system was tested in the accommodation domain using real data acquired from the Web as well as the corpus of real queries in natural language. The thesis deals with both theoretical and practical issues that must be solved in a fully functional system. A complete work-flow is described, including preparation of data, natural language corpus, ontology design, annotation, semantic model and search.

Finally, a very detailed evaluation with promising results is presented and discussed. Special attention is also paid to open issues, such as a performance, a usability, a portability, or sources of a real Web data.

Abstrakt

Disertační práce popisuje kompletní systém pro vyhledávání v sémantickém webu použitím přirozeného jazyka. Systém je představen v kontextu výzkumu na poli Information Retrieval, sémantického webu, porozumění přirozenému jazyku a rozhraní využívajících přirozený jazyk. Hlavní výhodou rozhraní využívajících přirozený jazyk je možnost zadat otázku celou větou namísto vyplňování webových formulářů nebo použití pouze klíčových slov.

Vyvinutý systém využívá technologie sémantického webu tradičními i novými způsoby. Myšlenka sémantického webu vnesla mnoho zajímavých konceptů do modelování domén a sdílení dat napříč doménami. Navíc kombinace sémantického webu a rozhraní využívajích přirozený jazyk skýtá nové možnosti pro vylepšení uživatelského komfortu při vyhledávání.

Disertační práce má tyto hlavní přínosy. Zaprvé: byl navržen nový formalismum pro zachycení sémantiky otázky v přirozeném jazyce. Tento formalismum využívá technologií sémantického webu. Zadruhé: byl vyvinut statistický model pro sémantickou analýzu založený na strojovém učení. Zatřetí: systém byl otestován na reálných datech a reálných otázkách.

Systém byl testován na doméně pro vyhledávání ubytování. Data byla získána z reálných webových portálů stejně jako testovací otázky v přirozeném jazyce. Práce se zabývá teoretickými i praktickými problémy, které musí být ve funkčním systému vyřešeny. Je popsán celý postup získání dat, korpus otázek, návrh ontologií, anotace, sémantická analýza a vyhledávání.

Na závěr je provedeno velmi důkladné vyhodnocení funkčnosti systému. Pozornost je také zaměřena na otevřené problémy, např. výkon, použitelnost, přenositelnost na jinou doménu a jazyk a zdroje webových dat.

Contents

Acknowledgement

I would like to thank everyone who helped me.

Part I

Introduction

Chapter 1

Motivation and Introduction

Nowadays it is almost impossible to deal with the current extent of online data without a search engine. The possibility of finding a particular piece of information has been playing a crucial role in the Internet usability over the past decade. Major search engines can perform very fast and accurate search almost on the whole Web, providing simple user interfaces based upon keywords. Keyword-based search has proven to be very efficient on a collection of unstructured textual content, e.g. web pages. However, if users want to find information in a structured content, e.g. in a database, the basic keyword search fails.

A simple, yet sufficient solution on the Web can be provided by a form-based user interface. With this type of interface the user can typically combine keywords with some other restrictions, according to the specific domain structure. Form-based interfaces are user friendly in the sense that they do not require a prior knowledge of the underlying data structures from the user. The structure is typically shown as multiple forms or menus that allow further specification of the user request. Nevertheless, the form-based user interfaces are more complex than the straightforward keyword search.

One step beyond the above-mentioned traditional approaches are Natural Language Interfaces (NLI). The key feature of such interface is that users can search for the required information by posing their questions using natural language which allows formulating their information needs precisely. Since NLI can operate on both structured and unstructured content, it helps to unify the access to data and allows a new way of user experience.

The need of computer understanding natural languages has been playing a crucial role in many research fields in the few past decades. Some of the results have already been successfully transferred from the purely scientific proof-of-concept research into commercial sphere and industry. However, the combination of Natural Language Understanding (NLU) and Information

Retrieval (IR) brings a lot of new challenging tasks.

Another promising direction in the evolution of Web, the Semantic Web, has brought many interesting concepts into domain modeling and data sharing. Since its fundamentals are based upon very precise semantic description, many existing Web applications can benefit from incorporating Semantic Web technologies. Furthermore, the development in Natural Language Interfaces to Semantic Web (NLISW) has shown that bridging the gap between Semantic Web and Natural Language Interfaces can uncover new research challenges.

Inspired by the recent research, this thesis presents a complete Semantic Web Search using Natural Language (SWSNL) system. It uses the Semantic Web technologies in both traditional and new forms. It was tested in the accommodation domain using real data acquired from the Web as well as the corpus of real queries in natural language. It deals with both theoretical and practical issues that must be solved in a fully functional system. A complete work-flow is described, including preparation of data, natural language corpus, ontology design, annotation, semantic model and search. A very detailed evaluation with promising results is also presented.

1.1 Thesis Outline

The thesis is divided into two parts. The first part describes the current state of the art. Since the thesis topic covers a wide range of research directions, this survey is an attempt to present the most important up-to-date ideas from each research field. To the author's best knowledge, all fundamental subjects related to the thesis topic are covered. Still, the development in this area is very rapid and there is a chance that some new studies could have appeared very recently without author's attention.

The survey is divided into two chapters.

Chapter 2 is focused on the recent attempts to develop Natural Language Interfaces on structured data such as Semantic Web. It describes basic ideas of Semantic Web and ontologies. The main content of the chapter is an analysis of existing Natural Language Interface systems in the Semantic Web.

Chapter 3 deals with understanding of questions in natural language in general. The problems are presented from the perspective of recent development in spoken language understanding and it can bridge the gap between traditional approaches used in Natural Language Interfaces and recent advanced techniques for natural language understanding.

The second part continues with a thorough description of a development of a complete end-to-end Semantic Web Search system using a natural language.

Chapter 4 explores the possibilities of modeling domain data. It describes the target domain together with techniques required to deal with real Web data sources. A natural language corpus is also presented.

Chapter 5 describes formalism for capturing a natural language question semantics, based upon Semantic Web standards used in a unique way.

Chapter 6 proposes a statistical semantic model for analysing natural language questions.

Chapter 7 deals with semantic interpretation of a semantic annotation in order to perform a search on a particular knowledge base.

Chapter 8 thoroughly describes the evaluation of the SWSNL system. Open issues and the future work are then discussed in Chapter 9.

1.2 Discussion of the Terminology Used

The field covered by this thesis is fairly broad since it combines various research directions. Thus, it is important to discuss the terminology in advance in order to avoid any misunderstanding.

Natural Language Processing (NLP). NLP is a general term, mostly used to describe a family of tools and techniques dealing with processing of natural language.

Named Entity Recognition (NER). Named entities are defined as proper names and quantities of interest. In general, named entities relate to persons, organizations, location names as well as dates, times, percentages, and monetary amounts. The goal of NER is to identify named entities in a (plain) text (G. Zhou & Su, 2005).

Question Answering (QA). QA research is basically devoted to computer systems that understand a written natural language question and are able to present an answer, either in a natural language or in another form (Habernal, Konopík, & Rohlík, 2012).

Information Retrieval (IR). *"Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)."* (Manning, Raghavan, & Schütze, 2008, p. 1) Given this description, it is obvious that the term IR is very broad in its nature.

Spoken Human-Computer Dialogue. In general, spoken dialogue systems communicate with users in natural language. Such a system accepts spoken input and returns a spoken answer, forming a dialogue. A dialogue system should typically deal not only with questions but also with commands or answers (Jurafsky & Martin, 2008).

Spoken Language Understanding (SLU). SLU is a part of Spoken Human-Computer Dialogue system and its role is to robustly interpret the meanings of users' utterances. SLU implementation normally comprises of three main components: a speech recognizer, a semantic parser to extract the semantic information from the recognized utterance and a dialogue act decoder to determine the overall goal expressed by the utterance (He & Young, 2006b).

Natural Language Understanding (NLU). The meaning of the term NLU overlaps with SLU. Again, the role of NLU is to capture semantics of an input. It is assumed that the input is in textual form therefore a speech recognizer is not a part of a NLU system. However, some sources freely mix NLU and SLU, e.g. (He & Young, 2005).

Semantic Analysis. The same as NLU—the role of Semantic Analysis system is to represent the meaning of the given text input.

Natural Language Interface (NLI). Typically, the term NLI is used in the literature when a system can be accessed using (written) natural language. The system contains (mostly structured) information and, given the natural language question, it can find an appropriate answer.

Natural Language Interfaces to Databases (NLIDB). The same as NLI. The system holds information in a relational database.

Natural Language Interfaces to Semantic Web (NLISW). In this case of NLI, the information is stored in the form of ontology which plays an important role in the Semantic Web research field.

Semantic Web Search Using Natural Language (SWSNL). This is purely the author's choice of the thesis title. It is based upon previous literature survey and it also takes into consideration its most appropriate meaning. Although NLISW covers a very similar task, the most of the existing NLISW systems are designed to query the knowledge base in order to find also non-trivial information, such as counts of certain instances, structure of the knowledge base and others (see Chapter 2). On the contrary, SWSNL can be viewed as a special case of search engines that return a set of results according to natural language question.

Part II

State of the Art

Chapter 2

Natural Language Interfaces to Structured Data

Natural Language Interface (NLI) is a very general description of a user interface which allows to access data using a natural language. This term has been used mostly in the field of NLI to structured data, e.g. for accessing databases using NLI or NLI on the Semantic Web platform. Nevertheless, such a general characterisation of NLI can also cover Question Answering (QA) or a part of Spoken Human-computer Dialogue Systems (see next chapter). The terminology does not limit the scope of NLI explicitly, therefore there are some overlaps among the research fields (see the discussion on page 4).

As pointed out by (Kaufmann & Bernstein, 2007), querying structured data (or knowledge base, KB) using a domain-specific query language can be complicated for casual users[1]. Most of existing query languages such as SQL or SPARQL uses a precise syntax and they expect the user to be aware of the back-end data structure. Evidently, this allows to formulate exact queries by domain experts but on the other hand this is also a big obstacle for users. An evaluation conducted by (Kaufmann & Bernstein, 2007) compared four different interfaces to KB (namely: constructing SPARQL queries using graphical interface, controlled language interface, natural language interface, and keyword search) and it showed that NLI is the most appropriate interface to satisfy the user needs with just a little knowledge about the KB.

A short comment to the controlled language interfaces mentioned above: Since the understanding of *natural* language questions is still far from being solved, there have been attempts to put some constraints to the nature of the questions themselves by introducing Controlled Language (CL). CL is

[1] We will use simply the term *users*.

a subset of a natural language and it uses e.g. simplified syntax or limited vocabulary. An example of a recent CL system can be found in (Schwitter, 2010). However, we feel that CL is a dead branch in NLI due to effort that must be invested to design the CL and to train the users (which is not feasible in the case of e.g. freely accessible web-based QA systems).

This chapter presents the state of the art of NLI to a structured data along with its specifics and typical domains and applications. In the first section, the traditional NLI to databases and its history are briefly introduced. The second section then follows with an application of NLI in the Semantic Web environment. It also includes a brief introduction of some foundations of Semantic Web.

2.1 Natural Language Interfaces to Databases

Natural Language Interfaces to Databases (NLIDB) has been a well-studied research discipline since 1970's when i.e. *Rendezvous* system (Codd, 1974) or *Ladder* system (Hendrix, Sacerdoti, Sagalowicz, & Slocum, 1978) were developed. Both systems were tailored to a specific domain and were based upon hand-written semantic grammars. After gaining popularity in 80's (e.g. *Chat-80* (Warren & Pereira, 1982) or *Janus* (Weischedel, 1989), both based upon predicate logic and lambda calculus), in 90's NLIDBs were slightly loosing attention (Androutsopoulos, Ritchie, & Thanisch, 1995). However, some recent systems such as *Precise* (Popescu, Etzioni, & Kautz, 2003) or *LingoLogic* (Thompson, Pazandak, & Tennant, 2005) continued in the NLIDB development by incorporating recent methods and also by redefining expectations from such a system.

A brief description of NLIDB systems follows: The user writes a NL question, the system translates it into an SQL query, executes the query via the database engine and returns the result.

Since the research in NLIDB area has moved towards the Semantic Web during the past decade (Damljanović & Bontcheva, 2009), we will focus on that area in the next section.

2.2 Semantic Web and Ontologies

One of the key ideas of the Semantic Web is that semantic data can be shared among computers in the form of ontologies which would enable to create a kind of a *global database* (Berners-Lee, Hendler, & Lassila, 2001). Since its introduction a decade ago, Semantic Web has emerged into a set of standards and technologies (e.g. RDF, OWL, SPARQL—will be explained later in

more details), software tools (e.g. Semantic Web search engines, semantic repositories, and knowledge bases), and techniques and design concepts (e.g. domain modeling using ontologies).

The term *ontology* has many definitions in computer science varying from a quite abstract definition "Ontology is an explicit specification of a conceptualization" by (Gruber, 1993) to concrete ones, e.g. "Ontologies are (meta)data schemas providing a controlled vocabulary of concepts, each with an explicitly defined and machine processable semantics." (Maedche & Staab, 2001). From the SW perspective, the term ontology has been established as something with the following features (Spanos, Stavrou, & Mitrou, 2012):

- a set of strings that describe lexical entries for concepts and relations,
- a taxonomy of concepts with multiple inheritance,
- a set of non-taxonomic relations—described by their domain and range restrictions,
- a hierarchy of relations, i.e. a set of taxonomic relations,
- a set of axioms that describe additional constraints on the ontology and allow to make implicit facts explicit.

From a practical perspective, ontology refers to various levels of formal semantic description, from e.g. a glossary or a class taxonomy to a domain class model or even to a model with formal logic constraints populated by instances and their relations. See e.g. (Lassila & McGuinness, 2001) for a good introduction to ontologies.

As the Semantic Web area has developed from the a abstract idea, a few standards and technologies for dealing with ontologies have been created. First, the *RDF* (Resource Description Framework[2]) is a basic building block of the SW technology stack. The RDF model is a directed labelled graph consisting of *triplets* (or *statements*). Each triplet contains a *predicate* (which describes a relation and acts as an edge in the RDF graph) and a *subject* and an *object* (which are nodes of the graph). Both the subject and the object can be seen as a real-world entity and a good practice is to denote them using a URI[3]. The RDF has various exchange formats, e.g. XML-based RDF/XML. However, the RDF itself has no other special functionality—"RDF provides a way to express simple statements about resources, using named properties and values" (Manola & Miller, 2004).

[2] `http://www.w3.org/RDF/`

[3] Unified Resource Identifier

In order to provide a *type system* on the top of the RDF, the *RDF Schema* (RDFS) was proposed. It brings facilities for describing classes and properties. Both classes and properties are inheritable, thus they can form a taxonomy. Furthermore, RDFS allows to enrich property definition by constraints, cardinality and transitivity. Moreover, different classes can represent the same class, two classes can be disjoint or combined (using intersection or union). For details see `http://www.w3.org/TR/rdf-schema/`.

Adapting the advanced features of the RDFS, the Ontology Web Language (OWL) is built upon it. It further extends the abilities for capturing semantics, by e.g. introducing *individuals* (*instances* of classes), distinguishing between *datatype* and *object* properties, etc. There exist three OWL dialects distinguished by its expressivity[4]:

- OWL Lite supports hierarchical classification and constraint features.
- OWL DL supports all OWL language constructs, e.g. type separation.
- OWL Full is the most expressive dialect.

The key feature of OWL Lite and OWL DL is that it is based upon Description Logic (DL) and thus allows semantic reasoning (Horrocks, Patel-Schneider, & Harmelen, 2003). From a practical point of view, OWL is a de-facto standard for ontologies in the Semantic Web environment.

2.3 Architecture of NLISW

This section deals with a general architecture of NLI systems on the Semantic Web (NLISW). Despite some patterns in the architecture of existing systems can be found, many systems are very unique and follow such an architecture quite loosely or even not at all. Thus the classification of the NLISW systems can be seen from different perspectives. A schema in (Gao, Liu, Zhong, Chen, & Liu, 2011) demonstrates various kinds of NLISW systems according to their approaches to semantic processing instead of trying to put all systems into one architecture and discuss each component separately. However, we will keep a structure according to the particual processing modules. We will first discuss the ontology as a knowledge base for NLISW systems.

2.3.1 Knowledge Base and Ontologies

Basically, the main difference between general Question Answering (QA) and NLISW is the presence of a structured knowledge base with precisely

[4]Consult `http://www.w3.org/TR/owl-features/` for details.

described semantics in the latter one. Whereas QA must find an answer in a collection of unstructured data (typically text documents), NLISW has not only the ability to find the facts in the knowledge base but also the ability to inference new facts using semantic reasoning. As the main vehicle for knowledge base, ontologies are used widely (Damljanović & Bontcheva, 2009).

Let's show an example question "How many five-star hotels are there in Prague downtown?". Even if the KB does not contain this particular information, it can be answered using an inference mechanism. On the other hand, most of the open-domain QA systems rely on a large set of documents that (probably) contain the desired answer in some textual form and thus it would be rather impossible to answer this example question.

Given the above mentioned distinction, the main field of using the NLISW is question answering on *closed domains*. Thus the essential requirement of any NLISW application is not only the presence of an ontology but also its quality. In this context, *quality* can mean, for example, completeness (i.e. how much knowledge of the particular domain is covered by the ontology), validity (i.e. whether the ontology instances fulfill the semantic constraints of that ontology) or uniqueness of entities (i.e. whether the same real-world object share the same instance in the ontology), among others. These quality flaws can be avoided when the ontology is created from scratch but many real-world applications either reuse existing data sources or merge various ontologies and thus the ontology quality must be taken into account. See e.g. (Frank et al., 2007).

2.3.2 Question Understanding

Generally, almost every existing NLISW system uses a unique combination of NLP tools and techniques, that will be discussed later. However, tokenization, NER, or syntactic parsing belong to the set of widely used methods. Many of the depicted system follows the traditional processing chain consisting of *preprocessing*, *syntactic parsing*, and *semantic processing* as shown in e.g. (Allen, 1995).

In the preprocessing step, the popular NLP toolkit GATE[5] is used by *FreyA* (Damljanovic, Agatonovic, & Cunningham, 2010), *AquaLog* (Lopez, Uren, Motta, & Pasin, 2007), or *PowerAqua* (Lopez, Fernández, Motta, & Stieler, 2011). GATE provides tools for tokenization and entity matching (the so-called *OntoRoot Gazetter* used in i.e. FreyA). It also comes with a framework for context-free grammars called *JAPE grammars* that are used in AquaLog and PowerAqua. Moreover, *stemming* can be found in *Precise* (Popescu et

[5] `http://gate.ac.uk`

```
Tagging:
I/PRP need/VBP a/DT cheap/JJ accommodation/NN in/IN Boston/NNP downtown/NN ./.

Parse:
(ROOT
  (S
    (NP (PRP I))
    (VP (VBP need)
      (NP
        (NP (DT a) (JJ cheap) (NN accommodation))
        (PP (IN in)
          (NP (NNP Boston) (NN downtown)))))
    (. .)))

Typed dependencies:                 Typed dependencies, collapsed:
nsubj(need-2, I-1)                  nsubj(need-2, I-1)
root(ROOT-0, need-2)                root(ROOT-0, need-2)
det(accommodation-5, a-3)           det(accommodation-5, a-3)
amod(accommodation-5, cheap-4)      amod(accommodation-5, cheap-4)
dobj(need-2, accommodation-5)       dobj(need-2, accommodation-5)
prep(accommodation-5, in-6)         nn(downtown-8, Boston-7)
nn(downtown-8, Boston-7)            prep_in(accommodation-5, downtown-8)
pobj(in-6, downtown-8)
```

Figure 2.1: An example output produced by Stanford parser for input sentence "I need a cheap accommodation in Boston downtown."

al., 2003) or *NLP-reduce* (Kaufmann, Bernstein, & Fischer, 2007).

Many systems depend on a syntactic parser, namely *FreyA*, *Orakel* (Cimiano, Haase, Heizmann, Mantel, & Studer, 2008), *Panto* (C. Wang, Xiong, Zhou, & Yu, 2007), *Precise*, or the system introduced in (Frank et al., 2007). Two commonly used parsers are the Charniak parser (Charniak, 2000) and the Stanford parser (Klein & Manning, 2003). Figure 2.1 shows an example output from the Stanford parser given the testing sentence *"I need a cheap accommodation in Boston downtown."*

The need of syntax preprocessing makes the adaptation of the systems harder, especially to languages that lack state-of-the-art tools for syntactic analysis or that are primarily too complex for even processing their syntax automatically with reasonable results.

Models for Capturing Question Semantics

After preprocessing and/or parsing, the step of creating a semantic representation usually follows. The formalism for describing the semantics of the question is rather unique for each system as well as the algorithms creating such a representation. However, there are some common patterns in the used approaches.

Heuristic rules are a common approach for transforming the output of the previous step (e.g. a parse tree) to the semantic representation like in e.g. PANTO, AquaLog, PowerAqua, NLP-reduce, and FreyA.

Ontology concepts or triplets as a framework for capturing the question semantics like in e.g. FreyA, PANTO, AquaLog, and PowerAqua.

2.3.3 Knowledge Base Querying and Answer Representation

Once the semantic formalisation of the input question is obtained, it can be transformed into a particular query language and then executed against the KB. Two popular query languages in the Semantic Web are SPARQL (Prud'hommeaux & Seaborne, 2008) and ReSQL (Broekstra & Kampman, 2003). The transformation into the query language is quite straightforward in many NLISW systems and it is mostly rule-based.

The result of querying the KB is a sub-graph of the ontology RDF graph (Prud'hommeaux & Seaborne, 2008). This means that the required information is stored in the triplet form and it should be transformed into a more human-readable output. However, this step is system-specific and depends on the desired usability of a particular system—the answer can vary from simple triplet representation to a full-sentence answer.

2.4 Evaluation Metrics and Testing Datasets

Whereas in established NLP branches (e.g. ASR or NER, among others) a standard testing datasets together with evaluation criteria are available, there is a lack of such dataset in the NLISW field. This is a fundamental issue since it is not possible to compare the performance of various systems. A deep exploration of the existing systems and their evaluation uncovers large inconsistencies in the performance evaluation. As pointed out by (Damljanović & Bontcheva, 2009), more attention should be paid to creating a standard evaluation set. Two typical problems related to the evaluation are:

- There is no widely accepted agreement on what is a *good* result. This is a very vague expression as it stands for e.g. a recall of correct questions (in Orakel), a number of answered questions (in Aqualog), or even a number of queries generated as an output (in Panto).
- Each system operates on different ontology and the ontologies vary in their size and complexity.

One of the widely used ontologies for the NLISW evaluation is the *Mooney: geography*. A deep quantitative analysis of this dataset was performed by (Cimiano & Minock, 2010). The dataset contains information about the U. S. geography, such as cities, rivers, mountains, countries, etc. Originally, the dataset was in Prolog (consisting of about 1000 Prolog facts), later it was converted into OWL by (Cimiano & Minock, 2010). The question corpus consists of 880 user questions with relation to the U. S. geography and it was collected from undergraduate students of the authors of (Tang & Mooney, 2001).

Although many NLISW systems used this corpus for the evaluation (e.g. Panto, Querix, NLP-reduce), the authors of (Cimiano et al., 2008) point out some interesting real-world issues. First, many systems 'cheated' in the evaluation on this dataset by hard-coding the meaning of adjectives and superlatives. Second, many systems claim to be portable but they require at least a hand-crafted lexicon to map the queries to the ontology.

2.5 Overview of Existing Systems

This section sketches an overview of some existing NLISW systems. The list is not intended to be exhaustive. Basically, it is focused on systems that have a significant impact in the field given their relevance as found in the literature.

2.5.1 PowerAqua and AquaLog

A very recent system called PowerAqua (Lopez et al., 2011) is a state-of-the-art ontology-based QA system which overcomes traditional NLISW systems by managing multiple ontology sources and scalability. Since its NL processing module remains the same as in the previous system AquaLog (Lopez et al., 2007), we will further discuss the AquaLog system.

AquaLog claims to be a portable NLISW system which handles user queries in a natural language (English) and returns answers inferred from a knowledge base. It consists of various components, namely: a linguistic component for transforming the natural language input into query triplets, a relation similarity service for generating ontology-compliant triplets, and a learning component to improve the system capabilities over time and incorporate a particular user-specific jargon. The architecture can be characterised as a cascade model where each component is fed with the output of the previous one. Basically, AquaLog provides portability on different domains which is only limited by the given ontology. Its authors suggest that only a little tuning must be performed before adapting the system to another domain.

However, an examination of the system uncovers many practical limitations. Firstly, the linguistic component is heavily dependent on syntax. The input query is processed by the GATE libraries, namely the tokenizer, the sentence splitter, the POS tagger, and the VP chunker. Furthermore, the question types are recognized by a set of hand-written JAPE grammars, that are based upon regular expressions. The principal disadvantage of using the JAPE grammars is that they can deal only with a subset of a natural language and, thus, complicate porting of the system.

Secondly, the relation of terms and their corresponding ontology concepts must be defined manually in advance. Practically, this means that the terms "who", "where" and "when" correspond to ontology terms "person/organisation", "location" and "timeposition", respectively. Furthermore, the so-called pretty names, that are alternatives of an instance, must be also listed. A textual (or WordNet) similarity between the query terms and the ontology is also assumed.

Thirdly, the transformation of query-triplets into ontology-triplets is based upon hard-coded heuristic rules. Furthermore, AguaLog can only deal with questions that generate a maximum of two triplets.

2.5.2 ORAKEL

ORAKEL (Cimiano et al., 2008) is an ontology-based question answering system. It accepts English factoid questions and translates them into first-order logic forms. This conversion uses full syntax parsing and a compositional semantics approach. ORAKEL can be ported into another domain but such a porting requires a domain expert to create a domain-dependent lexicon. The lexicon is used for an exact mapping from natural language constructs onto ontology entities.

A big drawback of ORAKEL's approach is the necessity of creating the lexicon manually. Another one is that the system can neither handle ungrammatical questions nor deal with unknown words. This makes the system unusable in any real-world environment.

2.5.3 FREyA

The FREyA system (Damljanovic et al., 2010) is a NLISW system that combines syntactic parsing with ontology reasoning. It derives parse trees of English input questions and uses heuristic rules to find a set of potential ontology concepts (for mapping from question terms onto ontology concepts) using GATE and OntoRoot Gazetteer (Cunningham et al., 2011). The primary source for the question understanding is the ontology itself. If the

system encounters ambiguities, a clarification dialogue is offered to the user. The potential ontology concepts retrieved from the question analysis are then transformed into SPARQL (presumably using some rules, this step is not described clearly). The system has been tested on Mooney: Geography dataset of 250 questions with results comparable to other systems evaluated on this domain.

2.5.4 PANTO

A portable QA system called PANTO (C. Wang et al., 2007) is based upon the off-the-shelf statistical parser StandfordParser and integrates tools like WordNet and various metrics algorithms to map the NL question terms to an intermediate representation called *QueryTriplets*. This semantic description is then mapped onto the so-called *OntoTriplets* that are connected to entities from the underlying ontology. This step involves a set of 11 heuristic mapping rules. Finally, OntoTriplets are represented as SPARQL queries. The PANTO architecture also contains a *Lexicon* which helps with matching NL words to ontology classes, relations, and instances.

The main idea of transforming a NL question into triplets in PANTO is based upon the author's observation that two nominal phrases from a parse tree are expected to be mapped onto a triplet in the ontology.

The system was evaluated on the Mooney dataset. The output of PANTO was compared to the manually generated SPARQL queries. One of the drawbacks of the system is its dependence on the full syntax parsing.

2.5.5 QACID

The ontology-based QA system called QACID introduced in (Ferrández, Izquierdo, Ferrández, & Vicedo, 2009) covers a cinema/movie domain and its target language is Spanish. It consists of two main components, the *user query formulation database*, and *textual-entailment engine*. Whereas the former component serves mainly for development and system training purposes, the latter one is intended for an unknown query processing. As a KB, an OWL cinema ontology was used. This ontology consists of a few classes and relations and it has been later populated with instances. This populated ontology served as a source for creating a lexicon (which is, in fact, a simple mapping between terms and their ontology concepts, e.g. "Real movie name" and "[MOVIE_CLASS]"). The core of the QACID system is a database of query formulations. The database contains a set of 54 clusters, each cluster represents one type of question and it has a representative query pattern which was derived from a set of training data. Each cluster is also associated to one SPARQL query.

When processing an unknown question, the system replaces all words that appear both in the sentence and in the lexicon with their ontology concepts. Then the textual entailment module tries to match the input question to a particular cluster from the query formulation database (this is actually a classification of the question). In this task, the question is treated as a bag-of-words and various lexical metrics are used for this classification. The entailment module also takes into account ontological attributes and relations of concepts identified in the question.

As pointed out in the QACID evaluation, the system is not able to answer unknown ontology concepts. In other words, if the user poses a question using terms that are not present in the lexicon, the system fails. Another errors are encountered when the query formulation database does not contain an appropriate cluster for the given question (this is a problem similar to out-of-coverage questions). Moreover, all the given question examples contain only one semantic concept and thus it is not clear whether the system is able to cope with more difficult questions with more ontology relations (see e.g. Mooney Geography on page 14 for examples of such questions). We suspect that the approach based upon the bag-of-words textual entailment would fail in this task. Finally, the system is not capable of handling temporal questions which is, from our point of view, somewhat startling in the case of a QA system on the cinema domain (definitely this would be a handicap when deploying this system to the real-world usage).

2.5.6 QUETAL QA

A domain-restricted QA system called QUETAL (Frank et al., 2007) expoits a robust semantic analysis on a hybrid NLP architecture. The system was developed to answer NL questions on two domains: the Nobel prizes and information about Language Technology Institute. For these two domains, the ontologies were converted from existing ontologies and merged with other more general ontologies (e.g. SUMO for the Nobel prizes). Unfortunately, the article shows only some concepts from the ontologies and presents them in a textual way thus it is not possible to uncover the complexity of the ontologies. The authors also admit that the source LT ontology did not respect domain and range properties.

Question analysis is performed by the HPSG (Head-driven Phrase Structure Grammar; see e.g. (Pollard & Sag, 1988) or (Pollard & Sag, 1994)) syntactic and semantic parser with a support of the Heart of Gold architecture[6] which provides shallow NER. The HPSG parser uses grammars for English and German and the output of the parser is a formalism called Minimal

[6]Heart of Gold (HoG) is an open source middleware for combining shallow and deep NLP components (last release from 2008), `http://heartofgold.dfki.de/`

Recursion Semantics which is a flat, non-recursive semantic representation format (Copestake, Flickinger, Pollard, & Sag, 2005). This question representation is then transformed into the so-called *proto queries* that are suitable for querying a particular KB. The system was evaluated on a set of 100 English questions in the Nobel prize domain only.

There is one notable limitation of the system. To adapt it to another language there must exist a wide-coverage HPSG grammar for that language

2.5.7 Other related work

QUICK: Instead of supporting full NL questions, QUery Intent Constructor for Keywords (QUICK) system guides the users in constructing a semantic query from keywords (Zenz, Zhou, Minack, Siberski, & Nejdl, 2009). After typing the keywords, the system analyses them according to the underlying ontology and it allows the user to choose the required semantic query in a clarification dialogue. The performance was tested using two ontologies (a movie database and a song database) and 175 queries consisting of 2-5 keywords. Although QUICK is not a fully-fledged NLI system, it can be viewed as a trade-off between the simple keyword search and the NL question systems.

A good overview of other state-of-the art keyword based semantic search systems is shown in (Fazzinga & Lukasiewicz, 2010).

CINDI: The system in (Stratica, Kosseim, & Desai, 2005) translates English questions into SQL queries and it is based upon syntactic analysis and simple hand-written templates. It was tested on a digital library domain but the authors provide very poor evaluation.

An example of the controlled language NLIDB is introduced in (Hallett, 2006). This system does not require (which means, actually, does not allow) the user to input NL questions but it assists in composing the query, starting with editing a basic query frame; the frames are automatically inferred from the database structure. The system was tested using GEOBASE with average results.

Chapter 3

Statistical Methods for Natural Language Understanding*

One of the key components of any system dealing with a natural language input is a Natural Language Understanding (NLU) component. This chapter explores the state of the art of NLU as viewed primarily from the Spoken Language Understanding (SLU) perspective.

Chapter Outline

- Section 3.2 focuses on sequential models for semantic analysis. These models use a non-hierarchical representation of semantics.
- Models capturing hierarchical semantic dependencies are discussed in section 3.3, followed by some other approaches to semantic parsing in section 3.4.
- Section 3.5 describes the evaluation of semantic analysis systems.
- An overview of existing systems and corpora is provided in section 3.6.

NLU on the Semantic Web. It is worth mentioning that NLU and Semantic Web have emerged from different historical and technical background. While NLU has played a crucial role since the first human-computer dialogue applications, the main idea of Semantic Web was to enable a formalised exchange of the web data and to make it accessible both for humans

*This chapter is an edited and updated version of author's previous technical report (Habernal, 2009).

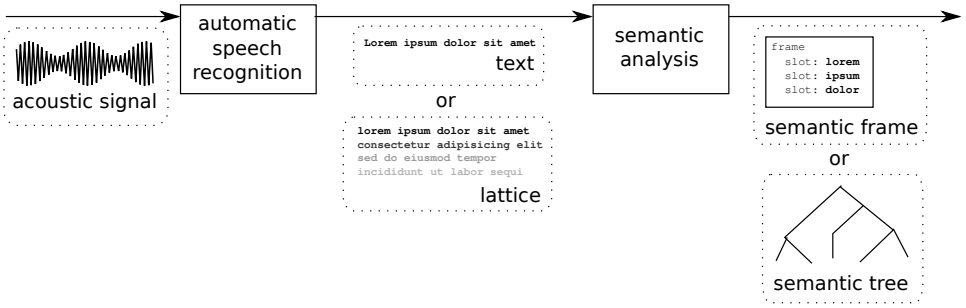

Figure 3.1: A schema of a SLU system.

and computers. Nevertheless, many recent Semantic Web applications require some sort of semantic understanding of the user question in order to return an answer from a knowledge base.

Before we present a thorough overview of the existing NLU techniques, we will draw one important conclusion. Although NLISW requires almost the same functionality in terms of understanding the question, many of the state-of-the-art NLISW systems are based upon approaches from the 1990's—namely syntax or hand-written heuristic rules (as shown in section 2.3.2). However, the research in the NLU field has made a tremendous movement towards statistical or hybrid methods and machine learning approaches in the past decade.

3.1 Introduction to NLU

The goal of a Natural Language Understanding system (NLU) is to extract the meaning from a natural speech. Although there is a difference between SLU and NLU in the sense of the input (an audio signal for SLU systems, a text for NLU systems), we will not distinguish between these terms so strictly. The main reason is that many of the discussed approaches and systems are originally focused on the whole SLU system, even though they deal with semantic analysis and assume a textual representation as the input. See also the terminology discussion in Section 1.2. A schematic example of a SLU system is in Fig. 3.1.

3.1.1 Basic Approaches

Early NLU systems were mostly based upon an *expert approach*, which means that the system was written entirely by a system designer (an expert). The *syntax-driven semantic analysis* (Allen, 1995) uses a hand-

written context-free grammar (CFG) for syntactic analysis. The first-order predicate calculus (FOPC) is used for meaning representation. Later, *semantic grammars* (Jurafsky & Martin, 2008) were based upon CFGs and described the semantic hierarchy rather than the language syntax.

However, the expert-based systems have a lot of limitations. The first disadvantage is a very high cost of creating such a system because the grammars must be written by an expert. Such system can also cover a limited domain and it lacks portability.[1] Although expert-based systems are still being used, recently the research has moved towards systems based upon machine learning and statistical models. The main advantage of such systems is an ability to learn from data.

3.1.2 Semantic Representation

The output of a NLU system is a context-independent[2] semantic representation. A commonly used representation of the semantics is the *frame based* representation.

In the frame-based NLU system, the semantic representation of an application domain can be defined in terms of *semantic frames.* Each frame contains several components called slots. The NLU system fills the slots with appropriate content. The meaning of an input sentence is an instantiation of the semantic frame. Some NLU systems do not allow a hierarchy of the 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.

3.2 Sequential models

3.2.1 Hidden understanding model

The Hidden understanding model (HUM) (Schwartz, Miller, Stallard, & Makhoul, 1997) was motivated by Hidden Markov Models (HMM), that have been successful in speech recognition. Because of differences between speech recognition and language understanding, significant changes are required in the HMM methodology. (Miller, Bobrow, Ingria, & Schwartz, 1994) proposes the following requirements for the HUM-based systems:

[1] *Porting* means adapting the system to different domain.
[2] Does not depend either on the history or on the context.

- A system for expressing the meaning.
- A statistical system which is capable to capture associations between words and their meaning.
- A training algorithm for estimating the parameters of the model from an 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 representation. It should be both precise and appropriate for automatic learning techniques. (Miller et al., 1994) requires a meaning representation which 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 a large corpus with low human effort.

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

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

Statistical HUM

Let $\mathscr{M}$ be the meaning space, $\mathscr{V}$ the vocabulary, and $W = (w_1, \ldots, w_T)$ the word sequence where $w_t \in \mathscr{V}$. Then the problem of understanding can be viewed as recovering the most likely meaning structure $M \in \mathscr{M}$ given a sequence of words $W \in \mathscr{V}$:

$$\hat{M} = \underset{M}{\operatorname{argmax}}\, P(M|W) \tag{3.1}$$

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

$$P(M|W) = \frac{P(W|M)P(M)}{P(W)} \tag{3.2}$$

where $P(M|W)$ is the *lexical realization model* and $P(M)$ is the *semantic language model*. Since $P(W)$ does not depend on M, it can be ignored in computing the 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.

3.2.2 Flat-concept parsing

A semantic decoding approach inspired by HUM (section 3.2.1) is the *Finite State Tagger* (He & Young, 2005) (or *flat-concept model* in (Young, 2002)). It assumes that each word w from the sentence is labelled with a semantic concept c. For example the sentence *"What will be the weather in Pilsen tomorrow morning?"* might be decoded in bracket notation as:

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

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

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:

$$P(W|C)P(C) = \prod_{t=1}^{T} P(w_t|w_{t-1} \ldots w_1, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \ldots c_t) \tag{3.3}$$

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 a *lexical model* (or a *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:

$$P(W|C)P(C) \approx \prod_{t=1}^{T} P(w_t|w_{t-1} \ldots w_{t-n+1}, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \ldots c_{t-m+1}) \tag{3.4}$$

For the special cases $n = 1$ 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 the state transitions are modeled as concept bigram probabilities and the words are modeled as unigrams conditioned by the concept c_t.

An example of the model is shown in Fig. 3.2.

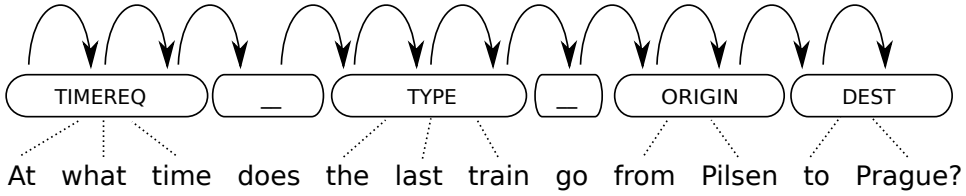

Figure 3.2: Discrete Markov model representation of semantics

3.2.3 Conditional Random Fields

The NLU problem can be also stated as a sequential supervised learning problem (Nguyen, Shimazu, & Phan, 2006). The result of the classifier is a semantic label sequence which 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. For example a two-level hierarchical slot can be considered as one flattened slot simply by concatenating two concepts.

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 of the same size. The goal of a classifier h is to find the most probable semantic class sequence given the input vector:

$$\hat{Y} = \underset{Y}{\operatorname{argmax}}\, h(Y, X, \Lambda), \tag{3.5}$$

where $\Lambda = \langle \lambda_1 \ldots \lambda_K \rangle$ denotes the parameter vector of size K.

Linear-chain Conditional Random Fiels (CRFs) are conditional probability distributions over label sequences that are conditioned by an input sequence (Y.-Y. Wang, Deng, & Acero, 2005), (C. Raymond & Riccardi, 2007), (Nguyen et al., 2006). Formally, linear-chain CRFs are defined as follows:

$$p_\Lambda(Y|X) = \frac{1}{Z(X)} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, X), \tag{3.6}$$

where X is the random vector that is observed, Y is the vector we want to predict and Ψ is a local potential ((Jeong & Lee, 2008), also called a *feature function* in (Sha & Pereira, 2003)). $Z(X)$ 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. In most cases a feature depends on the inputs around the given position i, although they may also depend on global properties of the input (Jeong & Lee, 2008). 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 if $y_t =$ `'TOLOC.CITY NAME-B'` and the current word is `'chicago'`. For higher values of λ_k the event occurs more likely.

3.3 Stochastic Semantic Parsing

3.3.1 Preprocessing

Most of the stochastic semantic parsers, that are introduced in this chapter, need a preprocessing step (He & Young, 2006b), (Pla, Molina, Sanchis, Segarra, & García, 2001), (Wong & Mooney, 2006). Since the terminology for this task is not stable, the problem can be called *shallow semantic parsing* (Pradhan, Ward, Hacioglu, Martin, & Jurafsky, 2004), *lexical class analysis* (Konopík & Habernal, 2009), *named entity recognition* (Tjong Kim Sang & De Meulder, 2003) or *NP-chunking* (Nguyen et al., 2006). In fact, these methods attempt to identify semantically meaningful units (words or word groups) in an input sentence and assign semantic labels to them. This definition of semantic classes of words is necessary in order to obtain high coverage models from a given data (Pla et al., 2001). There is a strong parallelism with the stochastic approach applied to the problem of text tagging. 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 3.2.2 or the conditional random fields (CRF) in section 3.2.3.

3.3.2 Probabilistic semantic grammars

The flat-concept parser described in section 3.2.2 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 corresponding concepts into a covering semantic concept. The long-distance dependency problems (see Fig. 3.3) also cannot be well described by the FST model. One possible solution is to use a more complex model, which can be e.g. the *probabilistic context-free grammar* (PCFG) model.

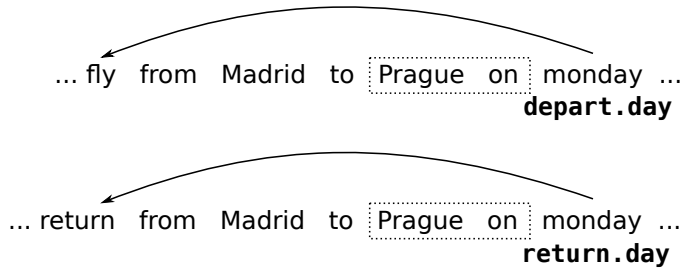

Figure 3.3: An illustration of the long-distance dependency problem of flat models using limited history.

The lexical model $P(C|W)$ can be obtained by using the above mentioned FST model (or by the other models from section 3.2). 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 a $P(c \to c_1 \dots c_Q)$ 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:

$$P(c, i, j) = \sum_{Q \leq j-i+1} \sum_{C_1^Q \in \{c*\}} \sum_{I_0^Q \in \{(i \dots j)*\}} P(c \to c_1 \dots c_Q) \prod_{q=1}^{Q} P(c, I(q-1), I(q)) \tag{3.7}$$

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

$$P(c, i, j) = \sum_{c_l, c_r} \sum_{t=1}^{i-1} P(c \to c_l c_r) P(c_l, i, t) P(c_r, t+1, j) \tag{3.8}$$

Now, the formula 3.8 is the *inside probability* of the Inside-Outside algorithm which is a modification of the EM algorithm for parameter estimation. 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 (Young, 2002).

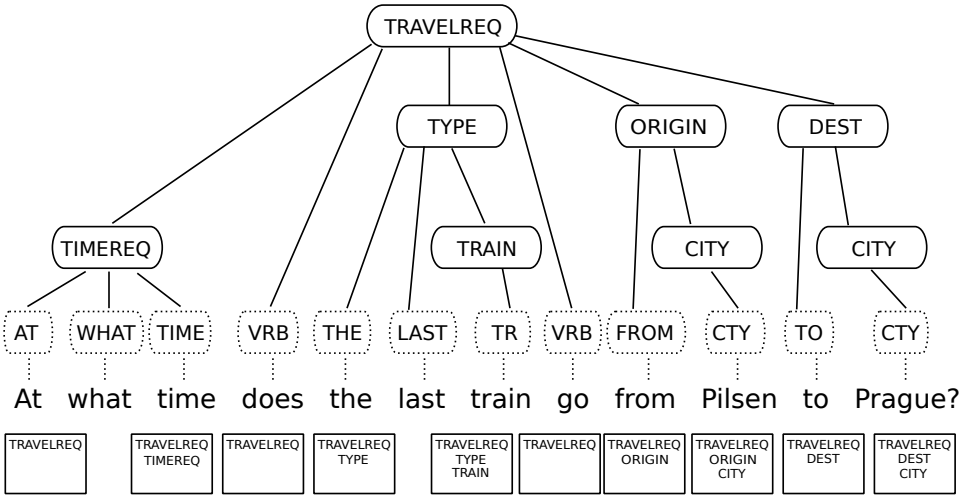

Figure 3.4: A vector-state model and its equivalent tree representation

3.3.3 Vector-state Markov model

The 1st order Markov model in *FST* was described in section 3.2.2. Although this model is mathematically simple and easy to train, its major disadvantage is a lack of ability to capture the hierarchy of semantics. This may lead to long dependence 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 1st order model, the *vector-state Markov model* is proposed in (He & Young, 2005). It is still a discrete HMM but each state is actually a stack of push-down automata with a limited size. This stack consists of semantic concepts. Thus, there is an equivalence between the PCFG and the vector-state Markov model with an unlimited stack depth. It is shown in Fig. 3.4. This model is a *right branching*. It means that the branches of the semantic tree grow in left-to-right direction.

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

$$P(N, \mathbb{C}, W) = \prod_{t=1}^{T} P(n_t|W_{1\ldots t-1}, \mathbb{C}_{1\ldots t-1}) \cdot P(c_t[1]|W_{1\ldots t-1}, \mathbb{C}_{1\ldots t-1}, n_t) \cdot \\ \cdot P(w_t|W_{1\ldots t-1}, \mathbb{C}_{1\ldots t}), \tag{3.9}$$

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 a 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 gains 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 the word w_t.

Each transition of the model is restricted to the following operations that correspond with the probabilities from Eq. 3.9: (i) pop n_t concept labels from the stack, (ii) generate a new preterminal concept, and (iii) generate a word. Having the n_t which defines the number of semantic concepts popped out of the stack, the transition from position $t-1$ to t given the preterminal concept c_{w_t} for the word w_t can be described as:

- pushing the concept onto the stack

$$c_t[1] = c_{w_t}, \tag{3.10}$$

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

$$c_t[2 \ldots D_t] = c_{t-1}[(n_t + 1) \ldots D_{t-1}], \tag{3.11}$$

- adjusting the stack depth at position t

$$D_t = D_{t-1} + 1 - n_t \tag{3.12}$$

Depending on the 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 out). The case $n_t = 1$ corresponds to replacing a preterminal concept with a new concept. And for $n_t > 1$ the stack reduces its size by popping out more concepts.

The general model, described in Eq. 3.9, depends on unlimited history of concepts and words. In (He & Young, 2005) the history is truncated—only the previous semantic concept stack is used and the word history is ignored. The equations are then approximated by

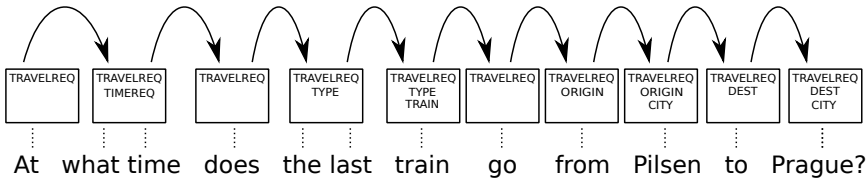

Figure 3.5: A vector-state Markov model

$$P(n_t|W_{1\ldots t-1}, \mathbb{C}_{1\ldots t-1}) \approx P(n_t|\mathbf{c}_{t-1}) \quad (3.13)$$

$$P(c_t[1]|W_{1\ldots t-1}, \mathbb{C}_{1\ldots t-1}, n_t) \approx P(c_t[1]|c_t[2\ldots D_t]) \quad (3.14)$$

$$P(w_t|W_{1\ldots t-1}, \mathbb{C}_{1\ldots t}) \approx P(w_t|\mathbf{c}_t). \quad (3.15)$$

The vector-state Markov model is shown in Fig. 3.5.

3.3.4 Hidden Vector-state Markov model

In the previous section, the basic vector-state Markov model was introduced. For training such model, a fully annotated corpus with aligned preterminal concepts is required. Then the training is simply a matter of counting events and smoothing the model. The *hidden vector-state Markov model* is based only upon an 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 upon MLE[3] and the model parameters are estimated using the Expectation-Maximization (EM) algorithm. 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 3.3.1). Second, an abstract semantic annotation must be provided for each sentence. These annotations are made by human annotators.

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 maximum likelihood of the training data. This is done by maximizing some objective function $R(\lambda)$. Most commonly used parameter estimation

[3]Maximum Likelihood Estimation.

is maximum likelihood estimation (MLE). MLE makes numbers of assumptions that 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 is unlimited. Thus, it is not guaranteed that the MLE-trained model will yield optimal results.

Discriminative training of the HVS model. As described earlier, the MLE is used for generative statistical training where only the correct models are taken into account during parameter estimation. (D. Zhou & He, 2009) proposes a method for training of the generative model using a discriminative optimization criterion. That means that not only the likelihood of correct models should be increased but also the likelihood of incorrect models should be decreased as well.

In discriminative HVS model training, 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 upon the generalized probabilistic descend algorithm (D. Zhou & He, 2009).

Extended HVS parser

(Jurčíček, 2007) introduced some extensions to the basic HVS semantic parser, namely the left-right-branching parsing and the input parametrization.

Left-right-branching parsing. The original semantic model of the HVS parser (see Eq. 3.9) allows to push only one concept $c_t[1]$ onto the stack. To enable either pushing one concept or no concept, a new hidden variable *push* is inserted into the model.

Another modification of the basic HVS model described in (Jurčíček, 2007) is the possibility of pushing two concepts at the same time. It has been experimentally verified that pushing more than two concepts does not affect the results significantly. Moreover, this limitation keeps the model simple.

Input Parametrization. In the basic HVS model, the input W is in a form of a word sequence. However, (Svec, Jurčíček, & Müller, 2007) 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 num-

ber of parameters to be estimated and it improves the model robustness. Nevertheless, the discrimination ability of such model decreases.

3.3.5 Context-based Stochastic Parser

The semantic parser proposed in (Konopík, 2009) 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 the vector-state parser introduced in section 3.3.3). The model parameters are then estimated using MLE:

$$P(N \to \alpha|N) = \frac{\text{Count}(N \to \alpha)}{\sum_\gamma \text{Count}(N \to \gamma)} \tag{3.16}$$

where $N \to \alpha$ means that in the data, the non-terminal N is rewritten to α. Moreover, a 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 the context given a non-terminal is estimated by MLE as:

$$P(w|N) = \frac{\text{Count}(w, N) + \lambda}{\sum_i \text{Count}(w_i, N) + \lambda V} \tag{3.17}$$

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

$$P(w|S) = \frac{\text{Count}(w, S) + \kappa}{\sum_i \text{Count}(w_i, S) + \kappa V} \tag{3.18}$$

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

Once the model is trained, the parser performs two steps. First, the shallow parsing algorithms (see section 3.3.1) are used to identify lexical classes. In this system, the shallow parser is based upon CFG for generic lexical classes such as dates, time, numbers, etc., and upon the vocabulary methods for proper names, etc. Second, a stochastic bottom-up chart parser is used to create parse trees and to compute probabilities as follows:

$$P(T) = \sum_i P(w_i|N)P(N \to A_1 \dots A_k|N) \prod_j P(T_j), \tag{3.19}$$

where N is a top non-terminal of the sub-tree T, $A_1 \dots A_k$ are terminals or non-terminals that are expanded from N and T_j is a sub-tree having the non-terminal A_i on the top.

Then the best parse is selected using the highest probability:

$$P(\hat{T}) = \underset{i}{\operatorname{argmax}} P(S_i) \prod_j P(w_j|S)P(T_j) \tag{3.20}$$

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

3.4 Other approaches to semantic parsing

Clustering approach

A novel algorithm for semantic decoding in SLU systems was proposed in (He & Young, 2006a). This approach differs from either rule-based or purely statistical systems mentioned previously. 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 small training data (less then the HVS model, sec. 3.3.3) and the data is annotated only at sentence level. The key idea is to cluster sentences into classes and then assign to a single semantic annotation to each class. In the decoding process, the input sentence is assigned to the class to which it most closely matches. The detailed model description can be found in (He & Young, 2006a).

Kernel-based statistical methods

In traditional machine learning methods, the input is represented as 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 is reduced to a set of features. To avoid this, all possible sequences of features must be taken into account, which makes the algorithm computationally expensive.

An alternative to the feature-based methods are the kernel-based methods (Kate & Mooney, 2006). 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 ∞. (Kate, 2009) defines a kernel between two strings as the amount of their common sub-sequences.

Semantic parsing is then performed by using the notion of a semantic derivation of a NL sentence. In (Kate, 2009), the task is described as finding the most probable semantic derivation of a 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 classifiers for computing the probability of the parse tree derivation.

3.5 Evaluation of NLU Systems

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

3.5.1 Exact match

The simplest criterion is the exact match criterion. Let C_R be the *reference* parse tree (made by a human annotator) and let C_P be the tree produced by the parser. Then the exact match criterion is a binary function returning 1 if the C_P matches exactly the C_R and 0 otherwise. The exact match means that both trees have 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 in e.g. one concept or even in preterminal concept positions, such tree is automatically discarded with 0. Therefore we need a finer measure which would penalize the output depending on its distance from the reference parse tree.

3.5.2 PARSEVAL

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

$$P = \frac{\#\text{ of correct concepts in } C_P}{\#\text{ of concepts in } C_P} \tag{3.21}$$

$$R = \frac{\#\text{ of correct concepts in } C_P}{\#\text{ of concepts in } C_R} \tag{3.22}$$

$$F = \left(\frac{\alpha}{P} + \frac{1-\alpha}{R}\right)^{-1} \tag{3.23}$$

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

3.5.3 Tree edit distance

The tree edit distance algorithm computes a minimum number of substitutions, insertions and deletions required to transform one tree into another. For semantic analysis system outputs it is possible to measure only the distance between abstract semantic trees without the relation to the words of the input sentence (therefore it is called *concept accuracy* in (Jurčíček, 2007)). The accuracy is then defined as:

$$CAcc = \frac{N - S - D - I}{N} \tag{3.24}$$

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

3.6 Existing corpora

In this section, an overview of existing NLU systems and corpora is presented. Instead of a comprehensive survey, some well documented and commonly referenced data sets and systems have been chosen.

3.6.1 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 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, a post-processing was required to extract relevant slots/values and transform them into a compatible format (He & Young, 2006b).

Although ATIS has been used for development and evaluation in many SLU systems, it is now over twenty years old (Price, 1990). There are several important issues that raise a question whether ATIS is a corpus suitable for evaluating a modern SLU application. For example, as pointed out by (Tur, Hakkani-Tur, & Heck, 2010), *"ATIS corpus is highly artificial and utterances are mostly grammatical and without disfluencies"*. Furthermore, many inconsistencies in the ATIS testing data set were discovered by (Habernal & Konopík, 2010).

3.6.2 DARPA

The DARPA Communication data contains utterance transcriptions and semantic parse results from the rule-based Phoenix parser (Ward & Issar, 1994). The abstract semantic annotations produced by the parser were hand-corrected. The data consists of 38408 utterances in total. After removing context dependent utterances such as "Yes", "No, thank you", etc., the final set consists of 12702 utterances. The corpus is publicly available without the semantic annotations.

3.6.3 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, and Polish). The Polish corpus was collected from the Warsaw City Transportation information center where people can get various information related to routes and timetables of public transport, etc. An automatic annotation process of this corpus is described in (Mykowiecka, Marciniak, & Glowińska, 2008). 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 so-called *active annotation* of this corpus is described in (C. Raymond, Rodriguez, & Riccardi, 2008).

The MEDIA project aims to evaluate SLU approaches to French language. The domain is targeted at hotel room booking and related tourist information. The corpus was recorded using a WOZ[4] system simulation. The semantic annotation procedure is described in (Bonneau-Maynard, Rosset, Ayache, Kuhn, & Mostefa, 2005).

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

3.7 Existing systems

3.7.1 HVS Parser

A hidden vector-state model (see section 3.3.3) was presented in (He & Young, 2005). The system was tested on the ATIS and the DARPA corpora, recently the system was also used for semantic extraction from the bioinformatics corpus Genia (D. Zhou & He, 2009). The first model training was based upon MLE (see section 3.3.4), however, a discriminative training has been also proposed.

An extension of the basic HVS Parser was developed in the work of (Jurčíček, 2007). The improvement is achieved by extending the lexical model and by allowing left-branching. The system was tested on a Czech human-human train timetable corpus.

3.7.2 Scissor

Scissor (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations) is another system which uses a syntactic parser enriched with semantic tags, generating a semantically augmented parse tree. It uses the state-of-the-art syntactic parser for English, the Collins parser (Collins, 1997). The parse tree consists of syntactic nodes augmented with semantic concepts. Some concepts (referred to as predicates) take an ordered list of arguments from sub-concepts. *Null* concepts, that do not correspond to any semantic information, are introduced as well.

The training of the system is performed in the following steps. First, the corpus is parsed by Collins 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 a 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 in e.g. (Ge & Mooney, 2005), (R. G. Raymond & Mooney, 2006) or (Mooney, 2007).

3.7.3 Wasp

Wasp (Word Alignment-based Semantic Parsing) is based upon statistical machine translation (SMT). The method is trained on corpora consisting of a natural language data and an 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 (Wong & Mooney, 2006). First, the words are aligned to corresponding semantic concepts using the GIZA++ system (Och

& Ney, 2003). Then a probabilistic parser and a synchronous CFG are used to create the parse tree.

3.7.4 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 3.4. In particular, Krisp uses the string kernel to classify sub-strings in a natural language sentence. Like Wasp, it uses production rules in the MRL grammar to represent semantic concepts. Learning the parameters of the system is an iterative process, in each step positive and negative examples are collected to train the SVM. During semantic parsing, Krisp uses these classifiers to find the most probable semantic derivation of a sentence (Kate & Mooney, 2006).

3.7.5 Other systems

Another system for stochastic semantic analysis introduced in (Beuschel, Minker, & Bühler, 2005) 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 the flat-concept parser (see section 3.2.2) because it uses the first-order HMM but the training stage depends again on a rule-based parser. However, a more general method for context definition was proposed in this approach. Instead of introducing data-dependent context classes, *contextual observations* are introduced.

In (Y.-Y. Wang et al., 2005), a generative HMM/CFG composite model is used to reduce the SLU slot error rate on the ATIS data. Also a simple approach to encode the long-distance dependence is proposed. The core of the system is based upon conditional random fields (CRF) and the *previous slot context* is used to capture non-local dependence. This is an effective and simple heuristics 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.

Part III

Natural Language-Based Semantic Web Search System

Chapter 4

Target Domain

This chapter describes the domain on which the SWSNL (Semantic Web Search using Natural Language) system was developed and tested. It starts with a discussion of the domain selection, continues with describing the process of obtaining data from users and concludes with testing data preparation.

4.1 Domain Requirements

As mentioned in section 3.1, *portability* plays a crucial role in the development of SWSNL system. A high degree of portability positively affects the costs required for adapting the system to another domain. On the contrary, if the system is tailored to a particular domain, it tends to perform better.

Our SWSNL system is domain independent.[1] Nevertheless, in order to develop and evaluate a full end-to-end system, an appropriate domain must be chosen.

The target domain should meet the following requirements:

- There must be a knowledge base (in any form) that contains the desired domain information.
- A testing corpus of natural language questions must exist for the particular domain.[2]
- For each question from the corpus, correct "answers" from the KB must be known.

[1] To a certain degree; see Chapter 7 dealing with semantic interpretation.

[2] In our SWSNL system the corpus is used for training as well; see chapter 6.

Obviously, these requirements are not exclusive to our SWSNL system. They can be applied to the evaluation of any NLISW system as discussed in sections 2.4 and 2.5. Furthermore, in order to prove the usability of the SWSNL system in a real-world scenario, we added few more requirements.

- There must be a promising commercial potential. It makes no sense to deal with simple or artificial domains or domains that have already been successfully deployed into commercial sector.
- The domain is currently accessible mainly via a form-based search. Users would benefit from a NL search.
- The NL question corpus must be as close to the real-world scenario as possible.

We also decided to focus on the Czech language at the first place. Due to its high degree of inflection and relatively free word order, it makes the task more challenging. However, it should be easy to adapt the system to English.

In sections 2.4 and 2.5 various corpora of NL questions and ontologies were introduced. Unfortunately, none of the existing corpora and KB meets our requirements as stated above:

- Lack of commercial potential or even solving non-existing problems (e.g. systems tested on wine ontology, Nobel prize ontology, digital library domain).
- Unrealistic NL question corpus (i.e. Mooney Geoquery[3]).
- Some domains are already parts of commercial solutions (e.g. public transportation domains).
- The vast majority of NL question corpora is in English.

The impossibility to re-use an existing KB/corpus has two effects. The disadvantage is that the performance of our SWSNL system cannot be directly compared with other existing systems. However, a brand new corpus/KB allows to explore new possibilities in SWSNL and thus go beyond the state of the art.

Finally, the *accommodation domain* was chosen. First, searching for a suitable accommodation is still far from being solved in any NLI system. Thus,

[3]In author's opinion, in a real-world search system users would not search for an information like *"What is the largest city in states that border California?"* or *"What is the highest point in the state with the capital Des Moines?"*.

it has a good commercial potential (comparing to other domains discussed earlier; this is the author's opinion). Second, the domain is not trivial and a fully functional end-to-end system must deal with many practical issues, that are sometimes ignored in the existing systems but that are essential in any real-world application.

In the rest of this chapter, the process of collecting a NL question corpus and creating a KB will be depicted.

4.2 Natural Language Question Corpus

This section describes a process of collecting a corpus consisting of NL questions in the Czech language on the accommodation domain. Whereas all existing NLI/QA systems expect the question to be a single sentence, we decided to go beyond the state of the art by allowing the users to ask their questions using one or more sentences to formulate their needs more precisely.

We gathered the user questions using a Facebook page[4]. The motivation was based upon our previous experience with obtaining data for NLI systems (Habernal & Konopík, 2009). It showed us that asking only i.e. students from a certain course can negatively affect the data. We observed that first few questions from one student were "real" questions—such questions are on-topic and they ask for some real data (i.e. train connections, train types, etc. in the train domain or, i.e. cheap accommodation in certain city in the accommodation domain). Starting with the 4^{th} or 5^{th} quesion from the same student, the quality of questions decreased and became poor for further usage in the corpus. The student mostly posed off-topic or simply silly questions (i.e. whether some hotel serves a certain type of beer, etc.). We think that even if the assignment for devising the questions was clear, at some point the students decided to test the abilities of the NLI system by asking "hard" questions instead of trying to find some real and useful information.

Thus, we claim that it is very important to choose a proper way of obtaining the NL corpus for an evaluation. Table 4.1 lists few corpora from the related work. Given this statistics, we suspect that many systems may suffer from the similar problem as we did and that their evaluation datasets are likely to be artificial.

We started the Facebook campaign with a clear description of the "virtual" SWSNL system. In other words, it says: *"There is a search engine that operates on accommodation domain in the Czech Republic. Try to find an*

[4] `http://www.facebook.com/pages/Vyhledávání-v-přirozeném-jazyce/112739178813727`

Corpus	Source and quantity
(Ruiz-Martínez et al., 2009)	4 Ph.D. students, 20 questions each
(Gao et al., 2011)	students (93 questions); manually created questions (77 questions)
(Ferrández et al., 2009)	10 unspecified users, total 540 questions; 10 user (high school students, administrative assistants), 10 questions each
(Frank et al., 2007)	100 questions, system developers
(Cimiano et al., 2008)	24 users (computer scientists from academic and industry), 10 questions each
(Tang & Mooney, 2001)	1000 questions from undergraduate students from the authors' department

Table 4.1: Sources of the evaluation questions in the related work.

accommodation that you really want by describing it using natural language instead of keywords. You are not limited to a single sentence. Just keep in mind that a human must be able to find the accommodation for you." A few diverse examples followed. At that time, we had no back-end database nor knowledge base thus we did not limit the users to ask questions that can be found in any existing accommodation search portal. To avoid poor quality of the corpus, as described previously, we asked each user to pose one to three questions.

After two weeks of quite intensive Facebook sharing and recommending to other people via "friends", we gathered 72 questions. This was less than we had expected but the whole campaign was based upon spontaneity and we didn't want to push anyone to participate in it. After discarding three really off-topic questions we had a corpus consisting of 69 questions.

Note that the participants were not asked for finding any answer to their questions. The main goal was to collect just the questions at this step.

4.2.1 NL Question Corpus Statistics

Table 4.2 shows the distribution of question lengths in terms of number of sentences used. Although there was no limit to the question length, most of the users asked using a single sentence only.

In the table 4.3, some interesting statistics follow. The average number of questions per user might be an important variable affecting the corpus quality, see the beginning of this section. Also the average question length

	1 sentence	**2 sentences**	**3 sentences**
number of questions	38	23	8

Table 4.2: Number of sentences per question.

Questions with missing diacritics	3
Average question length (words)	24.9
Number of unique participating users	42
Average number of questions per user	1.54

Table 4.3: Various NL question corpus statistics.

shows that the questions are relatively long.

The length of the questions is also important from an another point of view. The questions contain lot of information (see the examples in the next section). This makes the corpus very atypical, compared to the existing corpora (section 3.6).

4.2.2 Question Examples

Here are few examples of the NL questions in Czech with their English translation[5].

- *Ráda bych se ubytovala v Karlštejně s výhledem na hrad a královské vinice, nejlépe s polopenzí, parkováním a úschovnou kol, pro 2 lidi od pátku do neděle.* — I'd like to have an accommodation for two people in Karlštejn with a good view of the castle, including half board, a parking lot, and a bike rental from Friday till Sunday.
- *Jsme dva a chceme strávit jarní prázdniny v malebném hotýlku nebo apartmánu v Beskydech, blízko modré sjezdovky, předpokládáme lyžárnu, vyžadujeme polopenzi. Bazén, pingpong, wellness a podobné legrácky potěší, ale nejsou nezbytné.* — Two persons want to spend the spring holidays in a cute hotel or apartment in the Beskydy mountains. We require a ski-locker room and full board. Swimming pool, a ping-pong table and some wellness will be a pro but that's not necessary.
- *Hledám levné víkendové ubytování pro 3 odoby v Jindřichově Hradci nebo blízkém okolí, v blízkosti půjčovny jízdních kol nebo s možností uskladnění vlastních, pokoj nejlépe s krbem a možností připojení na internet, vlastní sociální zařízení podmínkou.* — I'm looking for a

[5]The translation is not 100% precise.

cheap weekend accommodation for 3 persons in the area of Jindřichův Hradec, near to a bike-rental store or with a bike locker-room. Room including a fireplace, internet and a private bathroom.

- *Hledám hotel na pořádání mezinárodní konference dne 25.10.. Hotel musí mít sál pro minimálně 100 lidí, wifi pokrytí, možnost uspořádání večeře, a minimálně 10 dvůjlůžkových pokojů. Upřednostňuji hotel s vlastní prezentační technikou* — I'm looking for a hotel suitable for hosting an international conference on October 25. We require a large hall (100 people at least), wifi, banquet, conference equipment. 10 double-bed rooms are minimum.

4.3 Knowledge Base and Domain Ontology

In order to develop a fully functional SWSNL prototype application, the underlying KB is essential. As already pointed out in section 2.3.1, the KB is a module that stores the complete knowledge about the covered domain. In the Semantic Web, ontologies are the main vehicle for domain modeling.

In our experiment on the accommodation domain, we had no such ontology during collecting the testing NL question corpus (see the previous section). On one hand, we did not restrict the user questions to comply with a structure or content of a particular domain model. On the other hand, there was a risk that the KB would not contain the answers to the collected NL questions.

After few unsuccessfull attempts to cooperate with companies running commercial accommodation web search portals, it became obvious that there is no easy way of obtaining an accommodation database (at least in the Czech Republic). This problem will be also discussed in section 9.1.5. The only way how to get a real database with accommodation possibilites was to use web data mining techniques.

4.3.1 Pattern-based Web Information Extraction

As a source website for data mining, the portal `hotelypenziony.cz` was chosen. The site provides a large database of various types of accommodation (from cottages to luxury hotels). All entries are categorised in one or more sections (by accommodation type or by tourist region) denoted by a URL prefix. Unfortunately, due to an inappropriate web structure, the same accommodation can appear at various URLs. Almost all entries have some location information, such as GPS coordinates or an exact address. Furthermore, each accommodation page contains some information about facilities,

prices, etc. but these fields are not mandatory. Some entries are enhanced with textual descriptions, like e.g. a site or a restaurant description.

Apparently, the website has a relational database on the back-end. However, we decided to put the extracted information into an ontology. The motivation was the following.

1. Ontology can be searched using a reasoner which can infer some facts that are not expressed explicitly.
2. Ontology can capture *both* the domain model and data instances.
3. Ontologies are an essential feature of Semantic Web and thus we wanted to test it in a real-world scenario.

The first point cannot be easily achieved using a standard relational database. An ontology, consisting of triplets, can contain e.g. symmetrical or transitional properties forming an unconstrained graph structure. Moreover, for example *transitive closures* cannot be defined in SQL[6] (Libkin & Wong, 1997). The second reason follows the basic idea of Semantic Web that data can be shared using ontologies with well-defined semantics and an easy-to-follow domain model for both computers and humans.

Crawling. The entire site was downloaded using an in-house crawler written in Groovy. Traditional crawlers failed due to a bad site structure design which caused an infinite link generation (the so-called *traps*).

Information extraction. A pattern-based information extraction tool was developed in order to extract the structured content from the crawled web pages. The extracted data were then transformed into the corresponding ontology instances and properties.

4.3.2 Domain Ontology

The domain ontology structure was designed according to the extracted information, see Figure 4.1.

The ontology structure is more or less "flat". It has one important class `Accommodation` with many object and data properties. The reason for this design comes from the structure of the website (the information source), where one accommodation page is equivalent to one instance of the ontology class `Accommodation`.

[6] Although Oracle RDBMS supports `CONNECT BY` clause which can be also ported to IBM DB2 database.

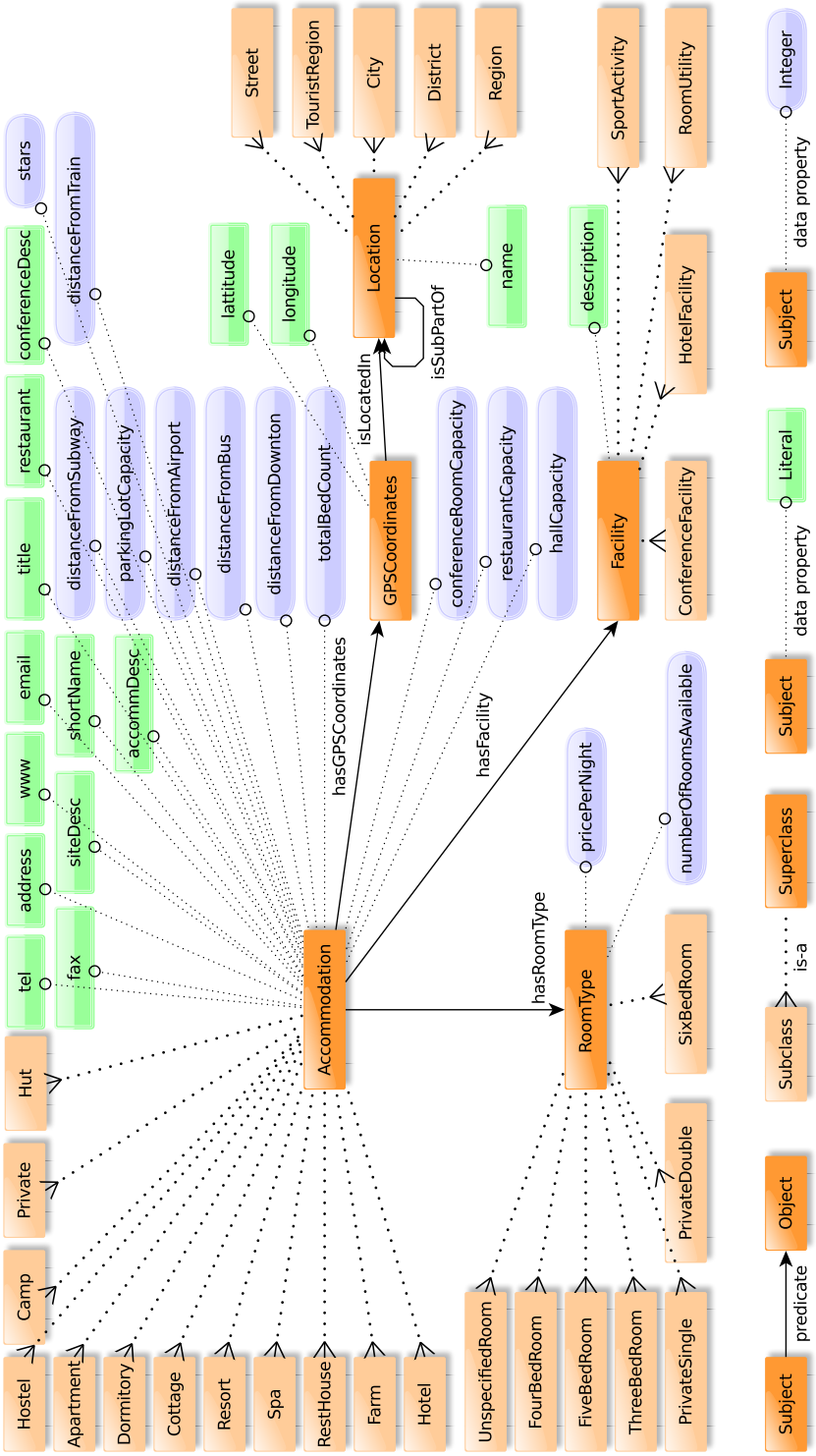

Figure 4.1: Structure of the *accommodation* ontology.

Some domain structure details are explained below:

- The `Accommodation` ontology class has several sub-types (e.g. `Hotel`, `Private`, `Apartment`, etc.) according to the type determined from the website section. The sub-classes are *disjoint* which means that a single accommodation instance cannot be i.e. both `Hotel` and `Spa`.
- The `hasRoomType` predicate has an unlimited cardinality. Each accommodation instance can contain multiple room types with appropriate prices and availability. Needless to say, `RoomType` instances are unique for each accommodation instance (they are not shared among all accommodations). This is very important for modeling e.g. number of rooms or room prices of a particular accommodation instance.[7]
- Similar to the previous point, the `Facility` instances have a certain type (e.g. `SportActivity` or `HotelFacility`). The `Facility` instances are shared among all accommodations, i.e. there is only one instance of `HotelFacility` called `facInternetConnection` which is linked to all accommodation instances with an internet connection. In the other words, there is only one type of internet connection which is the same for all accommodation instances.
- There is an important functional[8] object property `hasGPSCoordinates`. This links an accommodation instance to its unique `GPSCoordinates` instance. This will be discussed in the next paragraph.

Location hierarchy

During creating the KB from the website data, an essential requirement was stated—each accommodation must have its exact location. Without such information, it would be impossible to find any result when asking for an accommodation in a certain area or a city.

Figure 4.1 depicts a transitional relation `isSubpartOf`. This relation allows to capture a hierarchy of `Location` instances, e.g. `Street` is a sub-part of `City`, etc. However, the property is not functional thus it is possible that one location is sub-part of more than one other location. Typically, this is the case of `<c isSubpartOf a> & <c isSubpartOf t>` where `c` is an instance of `City`, `a` is an instance of `District` and `t` is an instance of

[7] For example, an accommodation instance *someHotelInst* is connected to an instance *someHotelPrivateDoubleInst* via the `hasRoomType` predicate and the *someHotelPrivateDoubleInst* contains a property `price`. This means that the room price is valid only for that particular accommodation instance.

[8] A functional property is an property in OWL which can have only one (unique) value y for each instance x, (Dean & Schreiber, 2004)

`TouristRegion`. An interesting task was to create such hierarchy only from the given address and the GPS coordinates.

Exact location extraction. Each accommodation page from the source website contained both an address and a GPS location. We discovered that most of the GPS locations were invalid (they did not correspond to the written addresses). We considered it as a big obstacle for a real-world application so we decided to discard the original GPS locations and tried to recreate them from the given textual addresses. We used the mobile user interface of Mapy.cz at `http://m.mapy.cz` and parsed the GPS coordinates from the retrieved text content.

4.3.3 Qualitative Analysis of the Knowledge Base

This section is dedicated to a deep analysis of the quality of the created KB. This analysis is necessary for further determination of possible system bottlenecks.

The total number of all instances (individuals) in the KB is 32.990 and the total number of all triplets is 281.686. The total number of instances of the `Accommodation` class[9] is 8641.

Statistics of Properties of Accommodation Instances

An accommodation basically contains two types of properties. We will distinguish them as *text properties* and *structured properties*[10].

Structured properties. These properties are both datatype and object properties (in OWL terminology) with exactly specified semantics. This kind of property basically contains a structured information, such as number values (e.g. various capacities or prices) or facilities (e.g. room facilities, sport activities).

Text properties. Text properties are four datatype properties (in OWL terminology), namely *accommDesc*, *siteDesc*, *conferenceDesc* and *restaurant*. These properties are mostly long text fields extracted from the accommodation website. They contain an unstructured description of the accommodation, e.g. restaurant information, description of the site, etc. Furthermore, these properties do not have an exactly specified semantic meaning, in opposite to structured properties.

[9] For the rest of this chapter, we will use simply *accommodation*.

[10] These properties are not the same types of properties as datatype and object properties in OWL; see section 2.2.

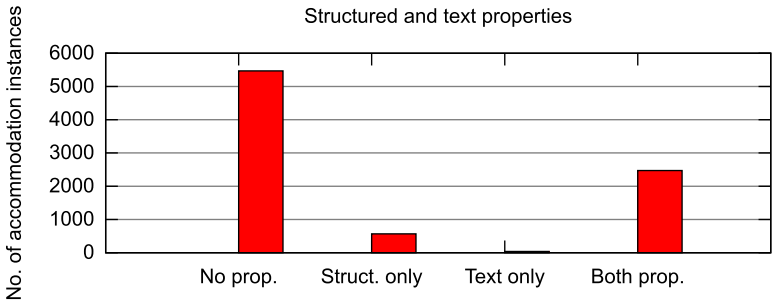

Figure 4.2: Distribution of text and structured properties.

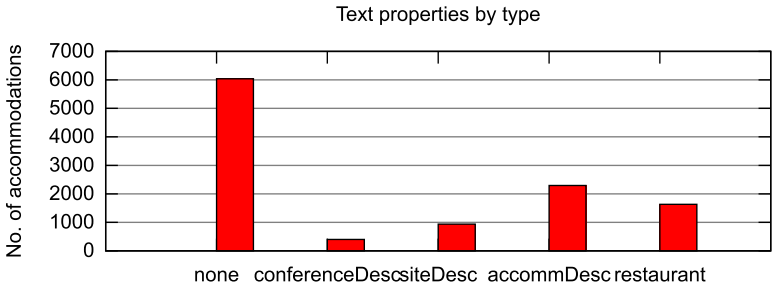

Figure 4.3: Statistics of the four text property types. *None* represents accommodations without text properties.

The structured properties are crucial for searching using a query language. On the contrary, if a piece of information is available only in text properties, it can be hardly searched using a query language. Figure 4.2 shows the distribution of the text properties and the structured properties over the accommodations in the KB. This figure captures one of the *most important* statistics. It says that about 60% of all accommodations has no additional property, except for the obligatory address. This means that we cannot say anything specific about these accommodations. The system only knows that a certain accommodation exists on a certain place but no additional information is offered. However, almost all the user questions presented in section 4.2.2 relate to a more specific type of accommodation, with e.g. certain properties, prices, room types, etc. This causes that almost 40% of accommodations cannot be found when users ask for additional accommodation requirements.

However, the text properties can contain a lot of unstructured information and the system can benefit from a combined search over the text and structured properties. Figure 4.3 depicts statistics of the four text property types.

Another important statistics outlines how many accommodation instances have a certain type of property. Again, this can later give an explanation of some search results. The statistics is shown in Table 4.4.

Property name	***i***	**(in %)**			
distanceFromSubway	182	2.1	restaurant	1632	18.9
distanceFromAirport	369	4.3	totalBedCount	1645	19.0
conferenceDesc	398	4.6	hasRoomType	2169	25.1
hallCapacity	430	5.0	accommDesc	2295	26.6
conferenceRoomCapacity	430	5.0	hasFacility	2614	30.3
parkingLotCapacity	765	8.9	fax*	3490	40.4
restaurantCapacity	768	8.9	www*	6401	74.1
distanceFromDowntown	790	9.1	tel*	6798	78.7
siteDesc	938	10.9	email*	7071	81.8
distanceFromTrain	941	10.9	title	8641	100.0
distanceFromBus	1124	13.0	address	8641	100.0
stars	1284	14.9	shortName	8641	100.0

Table 4.4: Number of accommodations (i) that contain a particular property (data property or object property). Cardinality of the properties is not taken into account, e.g. an accommodation instance can have multiple *hasFacility* properties. Thus, this table shows how many accommodation instances have *at least* one property of each type. *Note*: Star (*) denotes obligatory properties—each accommodation must have *at least one* of those properties.

Facilities. Important structured properties are instances of the *Facility* class connected to accommodation via the *hasFacility* predicate. This kind of property is one of the most important to satisfy user needs since it represents services, facilities and activities offered by a particular accommodation. Figure 4.4 lists how many accommodation instances have a certain number of facilities. This statistics does not say anything about the types of the facilities, however, it shows that most of the accommodation has about three facilities.

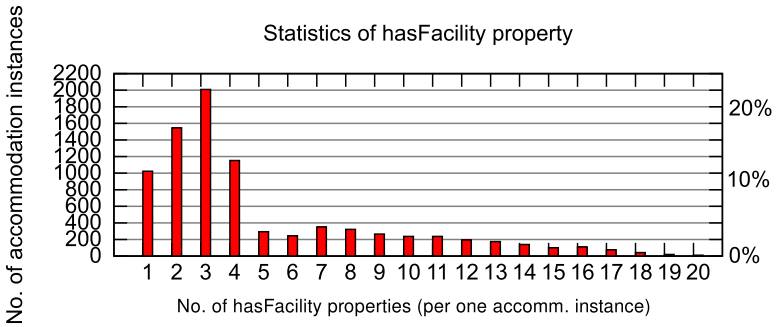

Figure 4.4: Number of accommodations with a certain number of facilities.

4.3.4 Semantic Inconsistencies Causing Practical Issues

The main advantage of a well-designed ontology is its precise semantics. It allows to use complex semantic constructs in order to infer new facts and query the KB using a query language.

In reality, systems must often deal with existing data since the amount of work to create and populate an ontology from scratch would be tremendous. Also our KB is almost completely created from the existing data. As the original data was not designed to respect a precise domain semantics, a sort of semantic inconsistency was transferred into the KB. This can later result in semantic ambiguity.

Our KB contains a few examples of semantic inconsistencies. We reveal some of them in the following list. Note that we had to leave all the data unchanged because it was not feasible to check and correct all the inconsistencies manually.

- The accommodation types are quite ambiguous. There is no clear difference between e.g. *Hut* and *Cottage*, among others.
- Lot of facilities are duplicated in some sense. For example, the KB contains four similar instances of a parking lot (namely *parkingLot*, *parkingLogWithLights*, *parkingGarage*, and *securedParkingGarage*). Apparently, the meaning of these different instances overlaps a lot.
- Some accommodations have total number of beds but the information about room types is missing (and vice versa).
- Moreover, prices of some accommodations are set to 1 CZK which is definitely a fake price.

There are two possible solutions how to deal with these inconsistencies. First, after the KB is created from the crawled data, it can be manually checked and altered in order to avoid such semantic inconsistencies. We can call it a *KB post-processing*. Second, the KB is left as-is and an attention must be paid when querying the KB. We decided to use the latter approach. For example, in our case if the system constructs a SPARQL query to retrieve all accommodations with a parking lot, all the four instances of parking lot must be taken into account when creating the SPARQL query.

Transitive Object Properties

As mentioned in section 4.3.2, the ontology contains one important transitive object property *isSubpartOf* and the location hierarchy is constructed

	Instance of TouristRegion	No. of relations
1.	hlavni-mesto-praha	841
2.	krkonose	678
3.	zapadoceske-lazne	466
4.	oblast-sumava	339
5.	jizerske-hory	317
⋮		
42.	kralicky-sneznik	10
43.	kokorinsko	7
44.	hana	3
45.	orlik	2
46.	blansky-les	2
47.	ceska-kanada	1

Table 4.5: Number of direct relations <*Street, isLocatedIn, TouristRegion*>

automatically. In most of the cases, the accommodation is connected to its GPSCoordinates (i.e. <*accommInst, hasGPSCoordinates, gpsCoordInst*>) and further to the location hierarchy (i.e. <*gpsCoordInst, isLocatedIn, streetInst*>, <*streetInst, isSubpartOf, cityInst*>, <*cityInst, isSubpartOf, districtInst*>, <*districtInst, isSubpartOf, RegionInst*>).

Nevertheless, the source website is also categorised by *tourist regions*. These regions are not explicitly defined and some of them overlap with existing cities (this is the case of the Czech capital Prague—on the website, Prague is both a tourist region and a city). This brings a kind of inconsistency to the data and makes the search according to a certain place or region much harder. Table 4.5 depicts such accommodations whose address (Street) is associated directly with a tourist region and does not belong to the above mentioned hierarchy (City-District-Region).

4.4 Test Data Preparation

For an evaluation of a search system, a test set of questions with answers is required. In other words, each question from the testing corpus must be associated with correct answers from the KB. In our case, the correct answers are instances of accommodations that satisfy user requirements, expressed by natural language questions. This will be the *test data* for our evaluation[11].

[11] We will also use the terms *gold results* or *gold data*.

Expression	Interpretation
in the vicinity of a city	all places in the district in which the city is located
close to (subway/downtown/train)	the distance is smaller than 1 km
cheap	any room of the accommodation has price lower than 1.000 CZK
cheapest	returns only one cheapest accommodation

Table 4.6: Semantic interpretation of some NL expressions

4.4.1 Search Results

The most important problem is to decide what is a correct answer for a particular NL question. As mentioned above, users look for an accommodation and thus a *set of accommodation instances* is considered as the search result. Each item from the search result must be a valid answer. In other words, the valid answer is when a human can treat the returned accommodation as *"This is exactly what the user wants to find."*

4.4.2 Semantic Interpretation

Many NL questions contain a sort of vague expressions, e.g. "cheap", "nice", "close to something", etc. It is necessary to decide in advance how to interpret these expressions consistently. The same interpretation must be used also later by the SWSNL system.

Table 4.6 presents some examples of the semantic interpretation. The values (prices and distances) were set after a consensus in our research team, however, this interpretation definitely depends on each user's point of view. We deal only with such expressions that can be quantified or expressed using the KB semantics.

4.4.3 Assigning the Correct Results

The only proper way how to create a perfect test set is to take each NL question, go through all accommodations and decide whether a particular accommodation is a correct answer to the question. Having the corpus of 69 questions and 8641 accommodations it is impossible to do this task manually. For this task we combined a *structured* search and a *fulltext* search.

Structured search. For each NL question a corresponding SPARQL query was manually constructed. We put as much constraints from the question as possible into the query. For example, if the user asks for par-

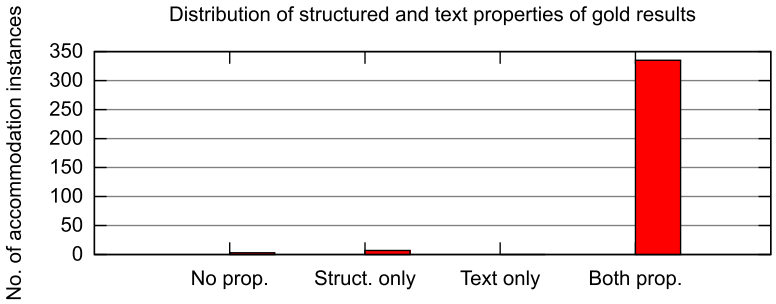

Figure 4.5: Distribution of properties in the gold data set.

ticular facilities (e.g. parking, internet), we add all of them into the SPARQL query. This ensures that only the accommodations that satisfy the constraints are selected from the KB (see the first paragraph from section 4.4.1). However, adding such strict criteria has two effects. First, the results that *partially* satisfy the requirements are simply ignored. At the moment, no ranking of results is involved. Second, if the accommodation has no structured properties (see section 4.3.3), it is also ignored by the structured search in principle.

Fulltext search. As presented in section 4.3.3, each accommodation can have a few text descriptions (labels). Such a description usually contains information which might be available in the structured properties as well (e.g. description of hotel facilities in the text and then again as structured properties). These text fields can be indexed and searched through using a fulltext search.

Finally, the whole process of assigning the correct results to each NL question was performed as follows:

- If the question can be completely expressed using SPARQL, create such a query and add the results to the correct results.
- If the question contains other requirements that cannot be expressed using SPARQL, filter the structured search results with a fulltext search and check the returned results manually.
- Create the least restrictive SPARQL query (only with location and accommodation type) and filter the results using a fulltext search. The fulltext keywords are manually selected according to the question requirements. The results must be checked manually.

In author's opinion, this whole process gives the best approximation of the correct test data. Needless to say, it took few weeks of tremendous manual work to create such a data set.

Figure 4.5 illustrates some statistics of properties of the gold data. Given this statistics, we can draw the following conslusion. The users want to find accommodations with particular properties such as facilities, prices, etc. These properties are stored in the *structured properties* and the *text properties*. Furthermore, the statistics shows that the gold data contains just a few results that are accommodations without any properties (the first column in Figure 4.5). However, as outlined in Figure 4.2 the KB contains almost 60% of accommodations that have no additional properties. This illustrates an imbalance between what the KB contains and what the users ask for.

Chapter 5

Semantic Annotation of NL Questions

5.1 Describing NL Question Semantics

Our formalism for describing the semantics of natural language questions exploits the idea of Semantic Web and ontologies. Since an ontology allows to define semantics very precisely (see section 2.2.), it also seems to be a suitable vehicle for capturing semantics of NL questions.

The first part of this thesis explored various approaches how the NL semantics can be expressed. Some of them are e.g. frames, FOPL (first-order predicate logic) or semantic trees, among others. Our previous research (Habernal & Konopík, 2009) and (Habernal & Konopík, 2010), based upon semantic trees, showed that describing semantics of NL questions should follow a particular structure, e.g. using a schema for adding constraints to semantic trees. Ontologies offer such a mechanism by allowing to define precise semantic relations between entities. Therefore, ontologies were chosen as a main vehicle for capturing NL question semantics.

We will now discuss the term *semantic annotation*. In the Semantic Web, this term has multiple meanings, e.g. webpages are annotated according to a particular ontology (taxonomy) or named entities can be annotated with related instances from an ontology. Our usage comes from NLU research where sentences (or questions) can be annotated with some additional semantic information. Henceforth, the term *semantic annotation* will be used in the sense of a *semantic description of a NL question*[1].

[1]Recall that a question can be expressed using a single sentence or a whole paragraph in our corpus, see section 4.2.

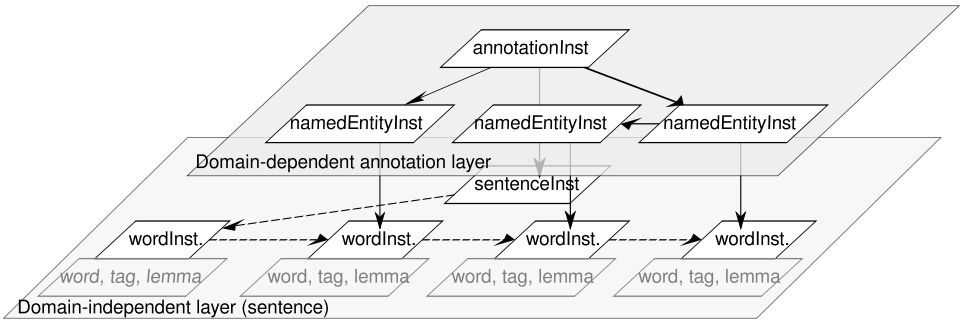

Figure 5.1: A schema of the two layer annotation of NL questions.

5.1.1 Domain Ontology versus NL Question Ontology

Ontologies for storing a domain information (Knowledge Base – KB) were already thoroughly discussed in section 2.2 and 4.3. However, in our proposal *the semantic annotation of NL questions based upon ontologies has no direct relation to the ontologies for storing a domain information (KB)*. This means that our semantic annotation is completely independent of a particular KB. Our semantic annotation uses ontologies *exclusively* for describing question semantics.

This kind of independence allows to separate the KB and the semantic description of NL questions in our SWSNL system. Thus, it is possible to e.g. switch the KB from an ontology to a relational database without affecting the semantic annotation of NL questions.

5.1.2 Two-Layer Annotation Approach

Figure 5.1 summarises our annotation approach in a schematic overview. The semantic annotation consists of two layers. The bottom layer is a domain independent layer with a domain independent sentence ontology. The upper layer is then constructed according to a domain-dependent ontology of NL questions. Both layers will be explained in more details in the following sections.

5.1.3 Ontology of NL Question

The domain-independent question ontology is shown in Figure 5.2. Each word is represented as an instance of the *Word* class[2] with some additional

[2]Note that *classes* and *instances* are seen from the Semantic Web/OWL perspective in this context. There is no direct relation to any particular implementation using an object-oriented programming language.

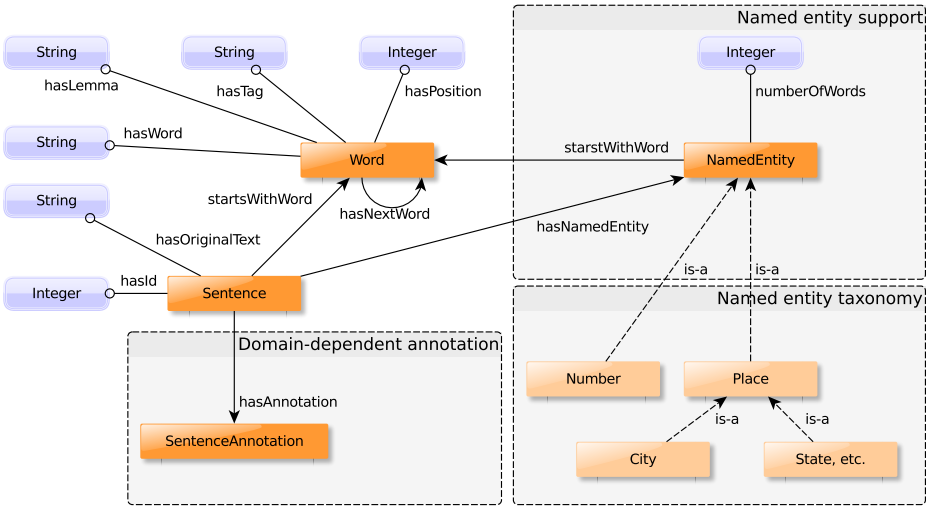

Figure 5.2: Sentence ontology.

morphological properties and its position within the sentence. An instance of the *Sentence* class covers a particular question.

The ontology has also a built-in support for marking named entities. An instance of *NamedEntity* is directly connected to the words. Furthermore, named entities form a taxonomy of classes. The basic sentence ontology contains few general types of named entities, such as *numbers*, *places*, etc. This taxonomy can be extended in the ontology for a particular domain (i.e. more precise places in our accommodation domain, etc.). In most cases, named entities are used to label expressions with a special meaning (e.g. cities, persons, countries, etc.). We extended the range of named entities to *all words/word groups that carry an important semantic information*. For example, we added the following domain named entities: *relative location* (covering e.g. "near", "in the center", etc.), *relative price* (e.g. "cheap"), or *additional requirements* (e.g. "fitness", "internet", etc.), among others.

The semantic annotation of a question is an instance of the *SentenceAnnotation* class and it depends on the domain in which the questions are annotated. An example of a question described by the question ontology is depicted in Figure 5.3. This simplified example shows only the triplet structure of the sentence annotation. The datatype properties are shown in the boxes connected to the ontology instances.

All ontologies are stored in OWL format and they were developed using Protégé[3].

[3]Protégé is a free, open source ontology editor and knowledge-base framework. `http://protege.stanford.edu/`

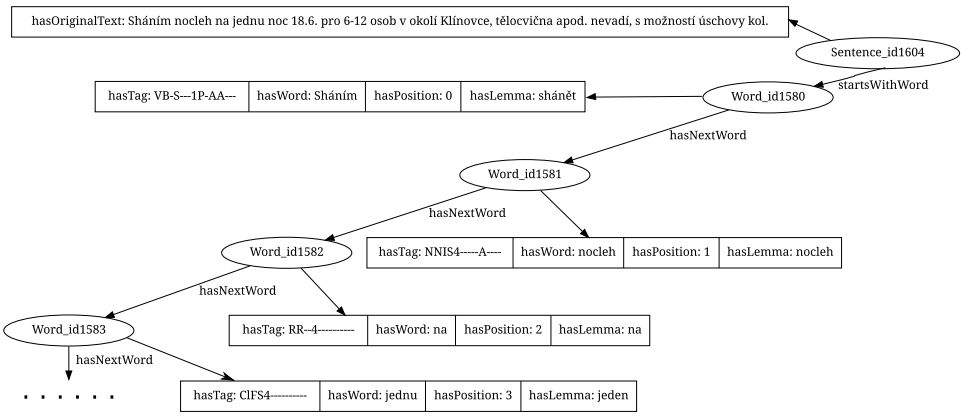

Figure 5.3: The triplet representation of a NL question. Only the first few words are shown.

Ontology for Semantic Annotation of Accommodation Questions

For each question, instances of the *domain-dependent question ontology* classes (triplets) are created. Those triples are then associated with the sentence. Due to its complexity the diagram of the complete triplet semantic annotation is not shown. Figure 5.5 depicts the semantic annotation of the previous example from Figure 5.3. Some words are omitted as well as the relation *hasNextWord* between the words and the datatype properties of the words.

Advantages of our Semantic Annotation Approach

We will conclude this section by a brief list of the main advantages we see in the ontology-based semantic annotation.

- Our semantic formalism does not depend on the structure of the KB.
- It uses Semantic Web standards: The sentence and its annotation is completely expressed in triplets. It can be processed using any OWL tool (e.g. Protégé).
- The semantic annotation has a strong semantic formalism. For example, inheritance, semantic relation between slots, or long dependencies can be captured. This is possible in frames too but no standardised formalism exists.

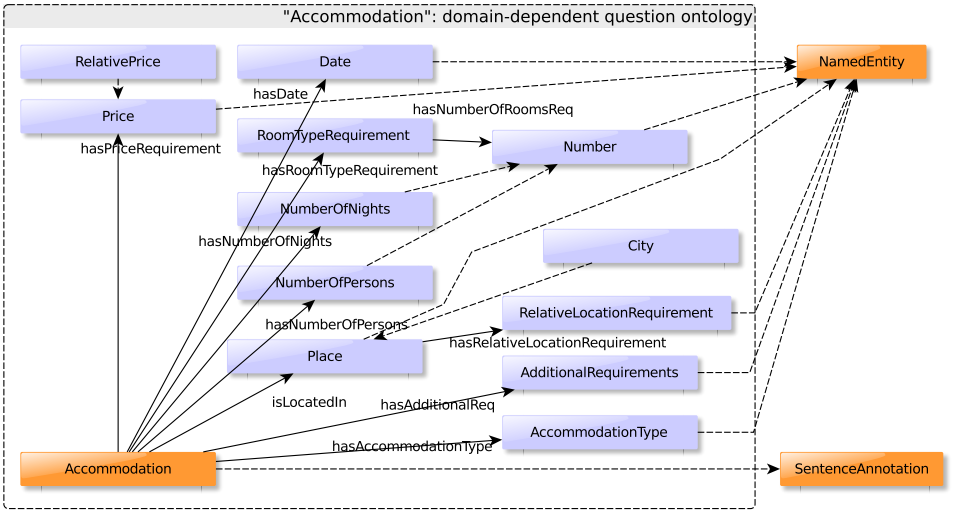

Figure 5.4: Annotation ontology for questions in the accommodation domain. The dashed lines represent the *is-a* relation.

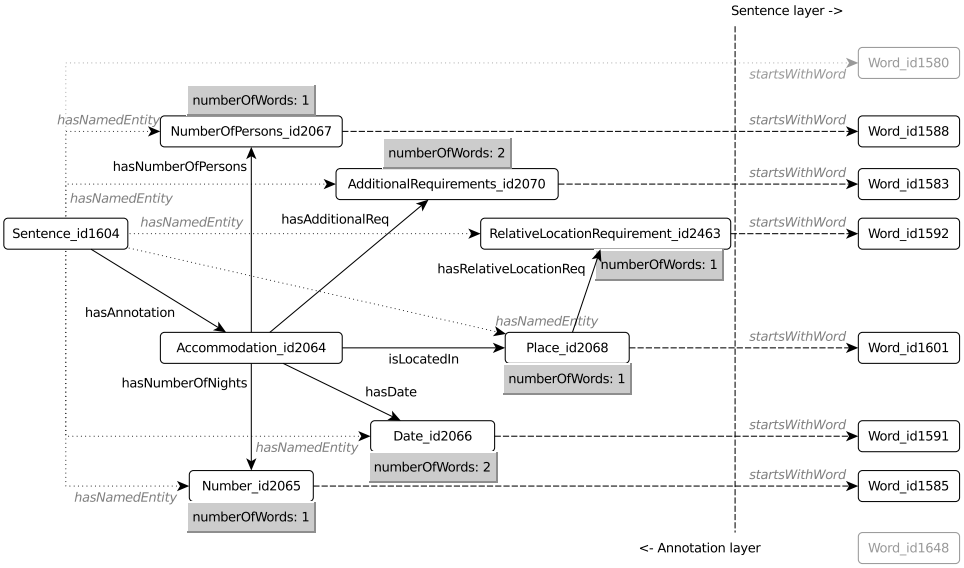

Figure 5.5: An example of semantic annotation. Simplified by omitting unrelated word instances and relations.

5.2 NL Question Corpus Annotation

To provide a complete training/test set for our SWSNL system evaluation, the NL question corpus from section 4.2 was annotated. As explained previously in this chapter, the semantic annotation based upon the accommodation domain ontology was created for each question from the corpus.

Since the complete corpus and its annotations are stored in the OWL format, the annotation can be created using any tool for ontology authoring. However, none of the existing tools (i.e. Protégé) was suitable for such a task. The semantic annotation formalism is fairly complicated (as shown in Figure 5.3 and 5.5) and the existing tools are not intended to create such complicated graph structures with lots of instances. Thus, we had to develop a new annotation editor.

Our *Ontology-based Semantic Annotation Editor* provides a balanced trade-off between usability and robustness. The editor hides the complexity of the underlying triplets and the sentence ontology by adding another layer between the user interface and the inner semantic annotation interpretation. Some important features are:

- Creating new instances and relations in a semantic annotation is constrained by the domain question ontology. It means that e.g. an instance of *Place* can have only one outgoing edge *hasRelativeLocation-Requirement*, according to the accommodation ontology (see Figure 5.4).
- The content words of named entities can be easily selected from a pop-up list.

The whole corpus was annotated by one human annotator and then manually checked by another one. The inter-annotation agreement was not measured because of the small corpus size. An example screenshot of the editor is shown in Figure 5.6.

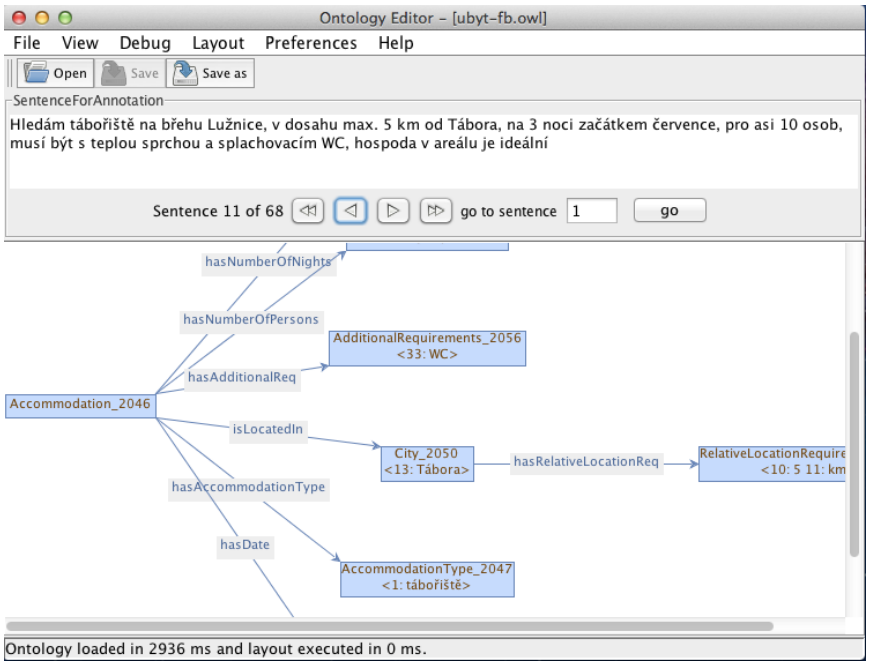

Figure 5.6: A screenshot of the ontology-based semantic annotation editor.

Chapter 6

Semantic Analysis of Natural Language Questions

This chapter describes our statistical model for a semantic analysis of NL questions. The main task of the semantic analysis component is to create a semantic description of previously unseen NL questions. More specifically, given a NL question from the covered domain (i.e. the accommodation, see section 4.2), the goal is to create an ontology-based semantic annotation (see section 5.1).

Various state-of-the-art NLU approaches were presented in section 2.3.2 and chapter 3. Although these systems provide acceptable results in many NLU applications, the algorithms cannot be adapted for our system because a different semantic formalism is used.

6.1 Semantic Analysis Model Overview

The semantic analysis component is based upon a statistical model and supervised learning. An annotated corpus is required for the model training. Furthermore, the component uses both off-the-shelf and newly developed preprocessing tools, such as tokenizer, morphological tagger and named entity recognition (NER).

Training. The model parameters are statistically estimated from a fully annotated corpus (see section 5.2).

Testing. Given a previously unseen NL question, the following steps are performed:

1. *Preprocessing*—The question is tokenized and tagged with lemmas and morphological categories. Currently, we use Prague Dependency Treebank tokenizer and morphological tagger (Hajič et al., 2006).

2. *NER*—Named entities are recognized (this will be discussed in more detail later in section 6.4).

3. *Semantic analysis*—Using the trained model, the most likely semantic annotation of the sentence is created.

The formal properties of the semantic model are proposed in the following section.

6.2 Formal Definition of Semantic Annotation

Let $S = w_1 \dots w_W$ be a sentence S (a question) consisting of W words $w_{1\dots W}$ and let $T(Subj, Pred, Obj)$ be a triplet of a subject, a predicate and an object. Then the semantic annotation Sem of a sentence S is an unordered set of M triplets

$$Sem(S) = \{T_1, T_2, \dots T_M\}. \tag{6.1}$$

Within $Sem(S)$, the triplets can be connected. Formally, two triplets

$$T_x(Subj_1, Pred_1, Obj_1), T_y(Subj_2, Pred_2, Obj_2) \tag{6.2}$$

can share their subjects or objects, so that $Obj_1 = Subj_2$ or $Subj_1 = Subj_2$, forming de facto a directed graph.

Named Entities. Let N^τ be a Named Entity (NE) instance with type τ. Then

$$N^\tau_{j,k} = N^\tau_{span}(w_j \dots w_k) \tag{6.3}$$

is NE of type τ which spans the words w_j to w_k, where $1 \leq j < k \leq W$. It means that NE is associated with the words from the sentence. Let C be a *semantic concept*. C is not associated with any words.

Both N^τ and C can be parts of a triplet so that each triplet has one of the following forms:

$$T(Subj, Pred, Obj) = \begin{cases} T(C, Pred, N^{\tau}_{j,k}) & \text{if subj is } C \text{ and obj is NE} \\ T(N^{\tau_1}_{j,k}, Pred, N^{\tau_2}_{o,p}) & \text{if subj and obj are NE} \\ T(N^{\tau}_{j,k}, Pred, C) & \text{if subj is NE and obj is } C \\ T(C_a, Pred, C_b) & \text{if subj and obj are } C \end{cases} \tag{6.4}$$

where j, k, o, p are the NE spans, a, b are used to determine possibly different concepts C and τ_1, τ_2 can be different NE types.

6.3 Statistical model

Using the above mentioned formalism, the task of a semantic analysis is to find an appropriate semantic annotation $Sem(S)$ given a sentence S. Using probability, we can formulate the problem as follows:

$$P(Sem(S)|S) = P(T_1, T_2, \ldots T_M|S). \tag{6.5}$$

Let's assume that the triplets are independent, thus

$$P(T_1, T_2, \ldots T_M|S) \approx \prod^{M} P(T_m|S) \tag{6.6}$$

where

$$P(T_m|S) = P(T_m(Subj, Pred, Obj)|w_1 \ldots w_W). \tag{6.7}$$

Furthermore, we limit our model to the first two triplet types from Equation 6.4. It means that *all* triplet objects are NE and *some* triplet subjects are NE, formally $T = T(C, Pred, N^{\tau})$ and $T = T(N^{\tau_1}, Pred, N^{\tau_2})$. This limitation is based upon our observation that the other two types of triplets are not currently present in the corpus.

In our model the triplet probabilities can be approximated as

$$P(T(C, Pred, N^{\tau})|S) \approx P(T(C, Pred, N^{\tau})|N^{\tau}_{j,k}) \cdot P(N^{\tau}_{j,k}|S) \tag{6.8}$$

and

$$P(T(N^{\tau 1}, Pred, N^{\tau 2})|S) \approx P(T(N^{\tau 1}, Pred, N^{\tau 2})|N^{\tau 1}_{j,k}, N^{\tau 2}_{o,p}) \cdot P(N^{\tau 1}_{j,k}|S) \cdot P(N^{\tau 2}_{o,p}|S) \tag{6.9}$$

Model parameter estimation. Given a training corpus $\mathbb{S}$ of sentences and their annotations $\mathbb{S} = \{S_n, Sem(S_n)\}$, the probabilities are estimated using MLE[1]. The triplet probability from Equation 6.8 can be estimated from the corpus as

$$P(T(C, Pred, N^\tau)|N^\tau) = \frac{\text{Cnt}(T(C, Pred, N^\tau))}{\text{Cnt}(N^\tau)} \tag{6.10}$$

and

$$P(T(N^{\tau 1}_{j,k}, Pred, N^{\tau 2}_{o,p})|N^{\tau 1}_{j,k}, N^{\tau 2}_{o,p}) = \frac{\text{Cnt}(T(N^{\tau 1}_{j,k}, Pred, N^{\tau 2}_{o,p}), N^{\tau 1}_{j,k}, N^{\tau 2}_{o,p})}{\text{Cnt}(N^{\tau 1}_{j,k}, N^{\tau 2}_{o,p})} \tag{6.11}$$

The NE probabilities $P(N^\tau_{j,k}|S)$ are calculated by a Named Entity Recognizer (NER).

Improving the model using a context. The presented model does not take into account any context of the recognized NE. To improve the estimated triplet probabilities, we incorporated a context of NE. The Equations 6.8 and 6.9 will then change into the following:

$$P(T(C, Pred, N^\tau)|S) \approx P(T(C, Pred, N^\tau)|N^\tau_{j,k}, w_{j-1}) \cdot P(N^\tau_{j,k}|S) \tag{6.12}$$

and

$$\begin{aligned} P(T(N^{\tau 1}, Pred, N^{\tau 2})|S) \approx \\ P(T(N^{\tau 1}_{j,k}, Pred, N^{\tau 2}_{o,p})|N^{\tau 1}_{j,k}, N^{\tau 2}_{o,p}, w_{j-1}, w_{o-1}) \\ \cdot P(N^{\tau 1}_{j,k}|S) \cdot P(N^{\tau 2}_{o,p}|S) \end{aligned} \tag{6.13}$$

The model training is then analogous to Equations 6.10 and 6.11, respectively. The model uses the *lemma* as the context of the previous word w_{j-1}.

6.4 Named Entity Recognition

The named entity recognition is a crucial part of the question preprocessing. To achieve reasonable performance, we incorporate three various NER approaches.

[1]Maximum Likelihood Estimation

6.4.1 Maximum Entropy NER

The maximum entropy-based NER (MaxEntNER) introduced in (Konkol & Konopík, 2011) was later extended and trained on a corpus from the Czech News Agency. The parameters of the corpus is very different from our NL question corpus since the Czech News Agency corpus consists of newspaper articles. However, the MaxEntNER is used to identify few general named entities, such as *Place* and *City*.

6.4.2 LINGVOParser

A shallow semantic parser based upon handwritten grammars LINGVOParser (Habernal & Konopík, 2008) is used to identify named entities with complex structure, such as *Date*, *Currency* or *Number*. The tool has proven to be efficient in such a task, as demonstrated in e.g. (Konopík & Habernal, 2009).

6.4.3 String Similarity Matching

This NER approach (called *WordSimilarityTrainableNER*) was used as an *ad-hoc* tool to deal with the remaining named entities that are not recognized by the two previous NERs. Recall that in our semantic annotation, the set of named entity types is much broader than in general NER because we use the named entities to cover all semantically important word groups (e.g. named entities like *AdditionalRequirements* or *AccommodationType*).

6.4.4 OntologyNER

For evaluation purposes we developed another NER called *OntologyNER*. Whereas all the previous NER types do not depend on the KB, this NER uses the KB instances for string similarity matching. Our assumption was that this type of NER would be suitable for recognizing e.g. places or various facilities. However, by incorporating this NER the Semantic Analysis system is no longer independent of the particular back-end KB.

Chapter 7

Semantic Interpretation

In the previous chapters, the domain KB, the NL question corpus, and the semantic analysis model were depicted. To make the SWSNL system working end-to-end, a module responsible for the *semantic interpretation* and the *semantic search* is required. This chapter focuses on various approaches dealing with such task. First, it presents a transformation from a domain-independent semantic annotation of a NL question to the target KB query language. It also involves e.g. mapping from named entities to ontology instances or semantic interpretation of named entities covering numbers or dates. Second, it deals with the semantic reasoning over the KB.

7.1 Transformation of Semantic Annotation into Query Language

After the NL question is automatically annotated with its semantic description, it must be interpreted in order to find the desired answer. Our semantic representation of a NL question, as introduced in section 5.1, is independent of a particular KB. Thus, in order to perform a search in the KB represented by an OWL ontology, the semantic representation must be transformed into an ontology query language. In our system, SPARQL is used.

Therefore, the transformation algorithm depends on the particular "back-end" KB. In our SWSNL the transformation consists of a set of heuristic rules. As an input, a semantic annotation of a NL question is given. As an output, a SPARQL query for the back-end KB is produced.

In our accommodation scenario (having the KB from section 4.3), the output SPARQL skeleton is shown in Listing 7.1. Each triplet from the semantic annotation is transformed into one or more SPARQL query statements that are inserted into the `WHERE` clause.

```
1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX p: <http://www.semanticweb.org/ontologies/2011accomm-cz.owl#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 SELECT DISTINCT ?Accommodation WHERE {
5   ...
6 }
```

Listing 7.1: SPARQL skeleton produced by the transformer.

Example. If the semantic annotation contains a triplet <*Accommodation, hasAdditionalReq, AdditionalRequirements*> where *AdditionalRequirements* is a sub-type of *NamedEntity* and it contains word *'internet'*, it is interpreted as `'?Accommodation p:hasFacility p:facInternet-connection .'`.

Apparently, this is an easy and straightforward example. Nevertheless, it points out practical issues regarding proper semantic interpretation of the named entities.

7.1.1 Semantic Interpretation of Named Entities

The named entities, as a part of the semantic annotation of NL questions, only span over word(s). The named entity has only its type but no semantic meaning is specified. In order to incorporate the named entities into the search process properly, it is necessary to interpret their content.

Matching Proper Names to Ontology Instances

As shown in the previous example, most of the named entities should have some relation to the underlying KB. In our case, entities such as *City* or *Place* somehow relate to the instances of the ontology classes expressing locations, etc.

The mapping between the KB instances and the named entities is based upon string similarity. Only a selected subset of the named entity types is matched using this technique.

In order to select the best string similarity (or string distance) we created a small subset of the ontology instances paired with the named entities and measured the accuracy of various types of string metrics. The best accuracy was achieved with the *Jaro-Winkler* distance (Winkler, 1990). Detailed results will be shown later in the evaluation part since as present the accuracy of matching the named entities to the ontology instances.

Interpretation of Named Entities with Numbers and Dates

Other types of the named entities that must be interpreted as well, are the named entities *Number* and *Date*. Apparently, these entities are not related to any ontological instances (they do not directly represent any instance from the KB), however, they appear mostly as a constraint in the NL questions.

The LINGVOParser tool used to recognize this type of named entities (see section 6.4.2) has also a built-in support for the semantic interpretation of the content of the named entities. Having numbers, dates and currency expressed in NL, the tool is able to translate them into a computer representation (e.g. integers or dates). This allows to integrate interpretations of these entities into the resulting SPARQL query quite easily.

7.2 Practical Issues of Semantic Interpretation

The issue of using vague expressions to express the user needs in NL questions was already pointed out in section 4.4.2 where the correct results from the KB were assigned to the NL questions. In the phase of deciding whether the particular result really "answers" the particular question, the human "annotator" can treat the task from his or her subjective perspective. However, this decision must be precisely formalised for a computer system.

Thus we applied the same decision rules as in the process of creating the testing data. These rules are automatically expressed as SPARQL statements.

Transformation example. Let us have a question asking for a "cheapest" accommodation. Then the semantic annotation of that NL question will contain the following triplet *<Accommodation, hasPriceRequirement, RelativePrice>*. The *RelativePrice* subject is the named entity and its semantic content is *"cheapest"*. Using the heuristic transformation rules, the output SPARQL query will be extended to the snippet from Listing 7.2. We can see the transformation as a *mapping* from the semantic annotation to the SPARQL query. This example also demonstrates that the NL question ontology is different from the KB ontology.

Non-trivial transformation example. A much more complicated situation arises when the semantic annotation contains *relative location requirements*. This typically means that the NL question contains a phrase like *"places near City"* etc.[1] The listing 7.3 depicts a construct which is used

[1] As mentioned earlier in Table 4.6 on page 54, we decided to interpret these constraints using the location hierarchy. Another option would be to use the GPS coordinates and

```
[...]
SELECT DISTINCT ?Accommodation WHERE {
  ...
  ?Accommodation p:hasRoomType ?roomType .
  ?roomType p:pricePerNight ?price
} ORDER BY ASC(?price) LIMIT 1
```

Listing 7.2: A SPARQL snippet with the interpretation of the 'cheapest' named entity from the semantic annotation.

```
[...]
SELECT DISTINCT ?Accommodation WHERE {
  ?Accommodation p:hasGPSCoordinates ?gps .
  p:cityInstance p:isSubPartOf ?district .
  ?district rdf:type p:District
  {
    ?loc p:isSubPartOf ?District .
    ?gps p:isLocatedIn ?loc
  } UNION {
    ?gps p:isLocatedIn p:cityInstance
  }
  [...]
}
```

Listing 7.3: An example of dealing with relative location requirements.

for transforming the semantic annotation containing the following triplets: <*Accommodation, isLocatedIn, City*>, <*City, hasRelativeLocationReq, RelativeLocationRequirements*> where the *RelativeLocationRequirements* is the named entity containing "near".

This (rather complicated) example demonstrates that a lot of attention must be paid to many details if the SWSNL system is intended to be deployed into a real-world use.

7.3 Semantic Reasoning and Result Representation

Given all the mechanisms introduced in the previous sections, it is possible to formulate the SPARQL query very precisely. Once the query is created, it can be passed to an ontology reasoner.

compute distances between the particular city and the possible places around but this brings even more complicated tasks, such as how to get a GPS coordinates of a city, what is the distance for "near", etc.

Reasoner class	Found instances	Total time
PelletReasoner	**206**	**15 min, 14 sec**
OWL_DL_MEM	12	11 sec
RDFSRuleReasoner	12	16 sec
OWL_LITE_MEM	12	12 sec
OWLFBRuleReasoner	**206**	**5 h, 15 min, 26 sec**
TransitiveReasoner	12	11 sec
OWLMicroReasoner	**206**	**55 min, 5 sec**

Table 7.1: Comparison of abilities and performance of various OWL reasoners.

Reasoner Performance Comparison. Currently, various OWL reasoner implementations are available. They also vary in their OWL expressivity which means what subset of the OWL semantics they can handle (see section 2.2). To select the suitable reasoner for our system, a simple test was carried out. The most important OWL feature we use is the transitivity among locations. Thus, a simple SPARQL query was executed to obtain all instances in a particular location and its sub-locations. Table 7.1 outlines the results of this test. We tested reasoners from Jena package as well as the Pellet reasoner. Some reasoners do not support transitive properties (they returned less than 206 instances in our test) so they were disqualified. Given the measured performance among reasoners, the Pellet reasoner was selected for our system. The performance of the reasoners will be discussed later in chapter 9.

Presenting Search Results. As already mentioned, in our domain the result of the semantic search is a set of accommodation instances (in fact, a set of instance URLs). To present the complete information to the user, a listing of all properties of the returned accommodation instances is shown in our system prototype.

Chapter 8

Evaluation

In this chapter, an evaluation of the complete SWSNL system is presented. We focus on the evaluation of the individual parts of the system as well as the complete end-to-end performance. Since there exists no widely accepted standards for evaluation of SWSNL/NLISW systems (see section 2.4), we use standard evaluation metrics for IR, namely the *accuracy*:

$$Acc = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}} \tag{8.1}$$

and the *precision (p)*, the *recall (r)* and the *F-measure (Fm)*:

$$p = \frac{tp}{tp + fp}, \quad r = \frac{tp}{tp + fn}, \quad Fm = \frac{2pr}{p + r}, \tag{8.2}$$

where *tp* are true positives, *fp* are false positives, and *fn* are false negatives (Manning et al., 2008). Unless otherwise stated, for the corpus-based tasks we use the *leave-one-out cross-validation* (Liu, 2011) due to small size of our corpus.

8.1 Semantic Analysis Evaluation

The ontology-based statistical model for the semantic analysis was outlined in chapter 6. Since it consists of two steps, the Named Entity Recognition and the Semantic Analysis, its evaluation will be split into two sections too.

8.1.1 Evaluation of NER

The system uses four types of NER, namely the *MaxEntNER*, the *WordSimilarityTrainableNER*, the *LINGVOParser*, and the *OntologyNER* (see section 6.4).

NER type	p	r	Fm
WordSimilarityTrainableNER	0.7056	0.4279	0.5327
MaxEntCZNER	0.2614	0.1143	0.1590
OntologyNER	0.8137	0.0244	0.0475
LingvoParserNER	0.4118	0.0125	0.0243

Table 8.1: Overall results of various NER on the accommodation corpus.

When measuring the NER performance, the *good result* (true positive) is when the named entity from the annotated corpus exactly matches the entity returned by NER, taking into account both the NE type and the word span.

Explanation of the results. As can be seen in Table 8.1, the best Fm is obtained from the *WordSimilarityTrainableNER*. The results of the *MaxEntNER* are poor in this overall evaluation. The very high precision of the *OntologyNER* has probably the following reason. If the KB contains a place or a city, it is very likely to be a place or a city in the testing sentence, too. On the other hand, only a small amount of named entities from the question corpus are stored in the KB which causes very low recall.

NER Performance by Entity Type

In order to explain the performance of NERs in more detail, we measured the performance determined by the named entity type. Figure 8.1 outlines Fm of the above mentioned NERs on different named entity types. Given these results, it is apparent that the *MaxEntNER* outperforms the other NERs in NE *City* almost by 100% relatively. This validates our assumption that the *MaxEntNER*, even though it is trained on a completely different corpus, yields acceptable results for general named entities, such as cities or places. Furthermore, the *LINGVOParser* performs well for numbers.

CombinedNER

Inspired by these results, we developed the *CombinedNER* which is a combination of the existing NERs. For each entity type, the NER giving the best result is chosen. The overall results of the *CombinedNER* are $p = 0.5218, r = 0.4891, Fm = 0.5049$. This NER gives slighly smaller Fm than the *WordSimilarityTrainableNER* ($Fm = 0.5327$, see table 8.1). However, we expected that the *CombinedNER* would perform reasonably on different corpora. We assume that the *WordSimilarityTrainableNER* works well for identifying domain-specific named entities that are lexically close, whereas the *MaxEntNER* is suitable for recognising more general domain-independent named entities. This will be shown later in this chapter.

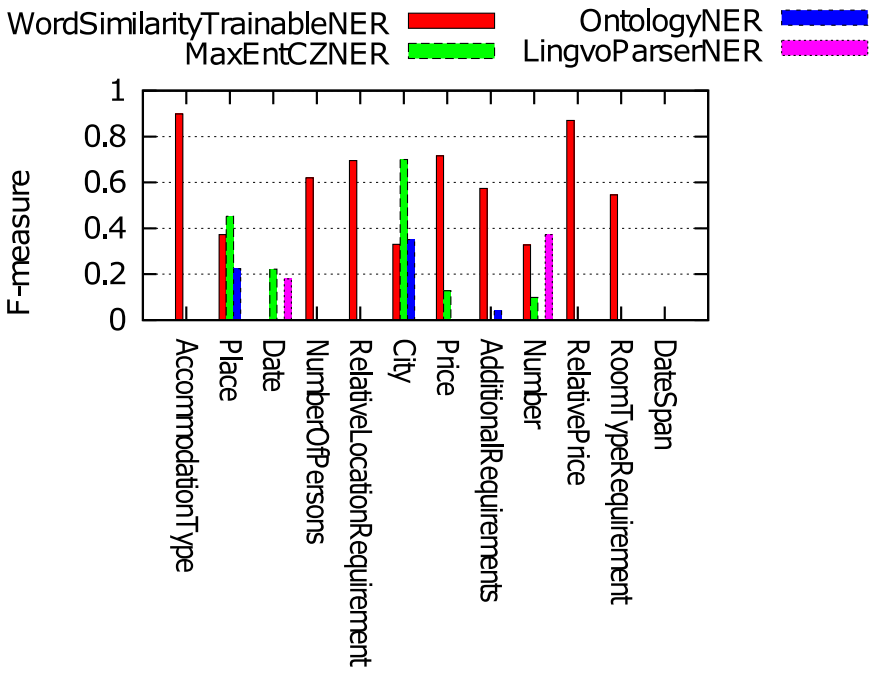

Figure 8.1: Performance of the NERs on different NE types.

8.1.2 Evaluation of the Semantic Analysis Model

Evaluation metrics. Although the result of the semantic analysis is a rather complex semantic annotation, there are few possible techniques how to measure the performance. Measuring the *accuracy* based upon the exact match of the returned result and the gold annotation[1] is, in our opinion, unreasonably strict. The *tree edit distance* (see section 3.5.3) is also not applicable because the semantic annotation can form a directed graph instead of a bare tree.

Thus, a good approximation of the quality of the analysed annotation seems to be the *number of matching triplets*. Recall that our semantic annotation can be also seen as a set of connected triplets. This allows us to measure the precision and the recall. A *true positive* result triplet is such a triplet that it matches the gold triplet as follows:

- The predicates are the same.
- The objects/subjects have the same ontology class, i.e. <*City, pred, obj*> and <*City, pred, obj*> match while <*City, pred, obj*> and <*Place, pred, obj*> do not.

[1] We use the term *gold annotation/triplet* for the correct manual annotations from the corpus.

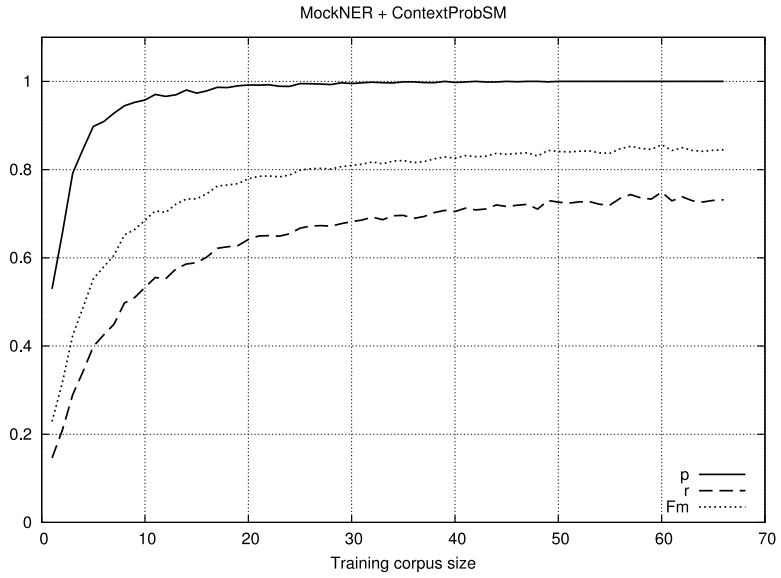

Figure 8.2: The semantic model performance depending on the training set size.

- Furthermore, if the objects/subjects are named entities, their positions in the sentence and their contents are the same, i.e. <*subj, pred, City(Velké*$_1$*, Popovice*$_2$*)*> and <*subj, pred, City(Popovice*$_2$*)*> do not match.

Simulation of a perfect NER – MockNER.

As mentioned before, the statistical semantic analysis model heavily depends on two inputs. First, a preprocessing and a NER are crucial for its functionality. Second, it requires an annotated corpus to train the model. To evaluate the model without influencing it by errors from NER, we performed tests with the so-called *MockNER*. This NER recognizes all entities in the sentence correctly because it uses the annotated data.

First experiment in Figure 8.2 shows how much the model performance depends on the training corpus size. For this experiment we used the *MockNER* and our statistical context-dependent semantic model (see section 6.3). Recall that *leave-one-out cross-validation* was used.

Discussion of the results achieved with MockNER. Given these results, we can draw the following conclusion. Even with a small training data, all returned triplets are correct ($p \to 1$). However, to find all triplets, a larger training set is probably required (r is growing slower). To validate

this conclusion, results on other corpora will be shown later in this chapter.

Results with CombinedNER

The *real* results (achieved by incorporating the *CombinedNER* into the semantic analyser) are shown in Table 8.2. Since the results might seem poor at the first glance, we will try to uncover the reasons thoroughly. First, the criteria for comparing the triplets, as shown earlier in section 8.1.2, are very strict from our point of view. This is mostly the case of incorrectly recognized named entities with mismatching types (e.g. *City* vs. *Place*) or NE content words. Second, the corpus of questions is very small to train the model properly. The very important part of the *CombinedNER* for recognizing non-general named entities (*WordSimilarityTrainableNER*) is completely trained from the data. This negatively affects the semantic model performance. However, to prove the qualities of our semantic analysis model, we will show results from a different corpus later in this chapter.

	p	**r**	**Fm**
CombinedNER+ContextSM	0.6627	0.3398	0.4492

Table 8.2: The results of the semantic analysis with the *CombinedNER*

8.2 Matching Named Entities to KB Instances

As described in section 7.1.1, some NEs are mapped onto their corresponding instances from the KB. This step is essential for creating the a SPARQL query. For the mapping a string similarity metrics is used. We tested various types of metrics on a set of 50 named entities and their manually assigned KB instances. Figure 8.3 shows accuracy of the mapping for various types of metrics. The best results were obtained by using the Jaro-Winkler metrics. Also we experimentally found the best similarity threshold for deciding whether a NE matches to an instance or not.

These results also give a slight insight into the accuracy of the NE matching which later affects the overall performance. Even when the semantic analysis produces correct annotation with correct named entities, the chance it will be mapped onto the KB instance is not 100%. In the overall evaluation, we examined several SPARQL queries produced by the system. Basically, there are three types of errors. First, the named entity is expressed in such a way that the string similarity between the words and the instance is very low. Second, the desired named entity has no corresponding instance in the KB. These two types of errors are quite obvious. The third type of error is caused by the inconsistence of the KB where two very similar instances

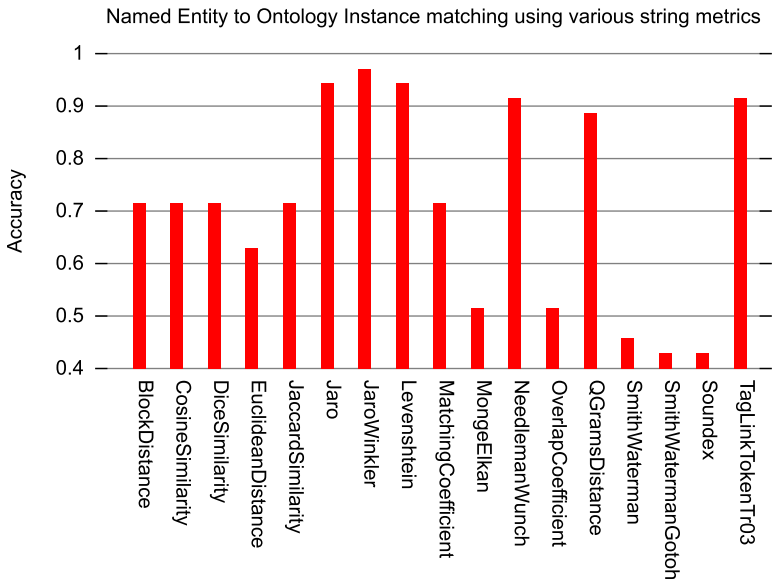

Figure 8.3: Various string metrics testing.

have a very similar description. For example, the KB contains more than 30 instances with *Praha* in their labels (referring to e.g. *District*, *City* or a city part). The reason of this inconsistence is given by the fact, that the KB was constructed automatically from existing data (see sections 4.3.2 and 4.3.3).

8.3 End-to-end Performance Evaluation

Each question from our NL question corpus has its corresponding set of correct search results (gold results), as described in section 4.4.3. It allows us to evaluate the performance of the complete SWSNL system.

Evaluation metrics. For each question the system returns a set of accommodations. There is no ranking of the results because the KB querying is purely boolean. Thus, we compute the precision and the recall of the returned set comparing to the gold set.

8.3.1 Fulltext Search

As a baseline for the end-to-end evaluation we used the fulltext search. First, the content of the KB was transformed into "text documents". More precisely, each accommodation instance with all its properties was treated as a single document. The documents were indexed using Apache Lucene

with the best Czech stemmer available (Dolamic & Savoy, 2009). In this experiment, the question was treated as a bag of words with stop-words removed. The overall results are shown in table 8.3.

	p	**r**	**Fm**
Fulltext	0.0159	0.4229	0.0306

Table 8.3: The end-to-end results of the fulltext search.

The results show very clearly that the keyword-based search performs poorly for such complicated tasks. However, this might serve as an approximation of results that would be returned by the current state-of-the-art search engines.

8.3.2 Simulation of End-to-end Performance With Correct Semantic Annotation

Another interesting and important experiment was performed by measuring the results returned when using the correct (gold) semantic annotation for each NL question. These results also set the limits for our semantic model approach because the gold semantic annotation is the best semantic annotation possible. In the other words, it simulates a scenario in which the semantic model reaches 100% accuracy.

	p	**r**	**Fm**
GoldSemAnnot	0.7310	0.8432	0.7831

Table 8.4: The end-to-end results of the search using the gold semantic annotations.

The results (table 8.4) not only illustrate the influence of the semantic interpretation and the named-entity to ontology matching (see Chapter 7) but also confirm that some answers cannot be found by purely structured search. This topic was already discussed in section 4.4 where the process of creating the gold results was presented.

8.3.3 The end-to-end results of SWSNL system

Table 8.5 shows the overall results of the the SWSNL system. The table also contains the best baseline keyword search as well as the simulation with the gold semantic annotation.

Discussion of the results. The most important conclusion, given these results, is that our semantic web search *significantly outperforms* the fulltext search. Whereas the fulltext-based search simply fails in this complex search task, our SWSNL system is able to yield acceptable results.

	p	r	Fm
SWSNL	**0.2493**	**0.6955**	**0.3671**
Fulltext	0.0159	0.4229	0.0306
GoldSemAnnot	0.7310	0.8432	0.7831

Table 8.5: The end-to-end results of our SWSNL system.

The results are very promising. In order to prove the qualities of our approach using the semantic analysis model, we conducted more tests. They are presented in the following section.

8.4 Evaluation on Other Domains and Languages

In the following section, only the semantic analysis module is evaluated as there exist no databases/KBs for these datasets. Thus, it is impossible to evaluate the end-to-end performance.

8.4.1 ConnectionsCZ Corpus

The *ConnectionsCZ* corpus is another NL question corpus in the Czech language. It contains 238 questions in the public transportation domain. Most of the utterances ask for a certain connection from one place to another, for train types, travel times, etc. The corpus is a subset of a larger corpus used in (Habernal & Konopík, 2009).

This type of corpus follows the fashion of corpora for SLU or human-computer spoken dialogue, such as ATIS, etc. Since our current work deals with the search and the Semantic Web technologies, we are not convinced that our SWSNL system would be the best approach for this domain and task. Nevertheless, this evaluation serves as a proof that our semantic model is domain independent and it can perform well if a larger training corpus was available.

Originally, the questions were annotated with semantic trees. In order to port the corpus into our system, we created an ontology for a NL question in this particular domain (see chapter 5). The domain-dependent question ontology is shown in Figure 8.4. The questions were then annotated according to the ontology.

NER performance. We used the same set of the NERs as in the accommodation domain. The results for each entity type are depicted in Figure 8.5. It shows that the LINGVOParser performs well on this task (compared to the results on the accommodation). Finally, we incorporated the

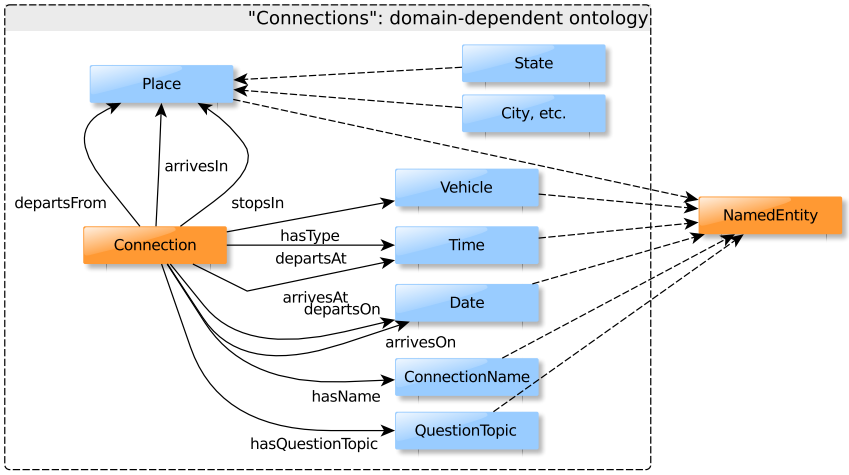

Figure 8.4: The ontology of the questions in the connection domain.

CombinedNER into the semantic analysis module.

Semantic Analysis performance. In our two experiments, we used again the *MockNER* to simulate 100% accuracy of the NER. Figure 8.6 (the left-hand side) illustrates the improvement of the semantic analysis as the training set size grows. The same figure (the right-hand side) contains also the results when the *CombinedNER* was used. The improvement in the performance is caused by larger training size for both the NER and the semantic model. Note that in the first scenario (with the *MockNER*) we run the test for every training data size $(1, 2, \ldots N)$, whereas in the second scenario (with the *CombinedNER*) the training data size was increased by adding ten sentences each run $(1, 11, 21, \ldots N)$.

8.4.2 ATIS Corpus Subset

The English ATIS corpus (already introduced in section 3.6) has been widely used in NLU research. We extracted only a small subset of the corpus (348 sentences) and annotated them using the same question ontology as for the czech ConnectionsCZ corpus, see Figure 8.4. Since our NER components are focused on the Czech language, we performed only one test on the ATIS corpus to evaluate the performance of our semantic model on a different language. The results with the *MockNER* are shown in Figure 8.7.

Discussion of results. We decided to test our system on the ATIS corpus just to get an insight into the possibilities of porting the semantic model to another language. This should not serve as a comparison with other state-

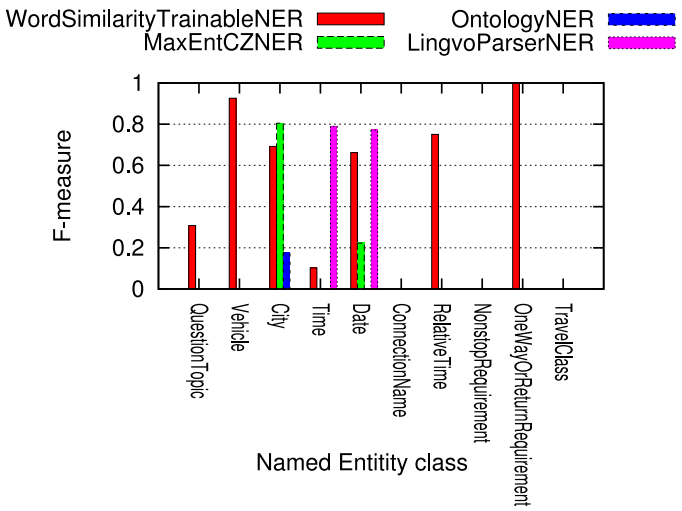

Figure 8.5: The NER results by the entity type on the ConnectionsCZ corpus.

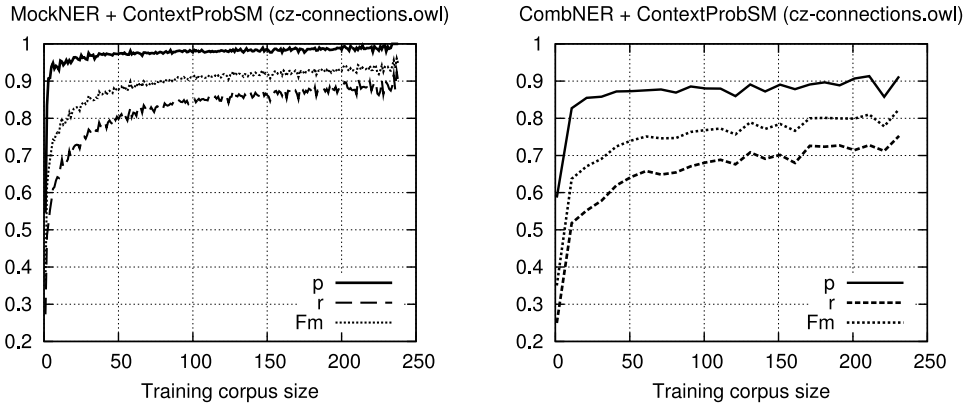

Figure 8.6: The evaluation of the semantic analysis on the ConnectionsCZ corpus using different training set sizes.

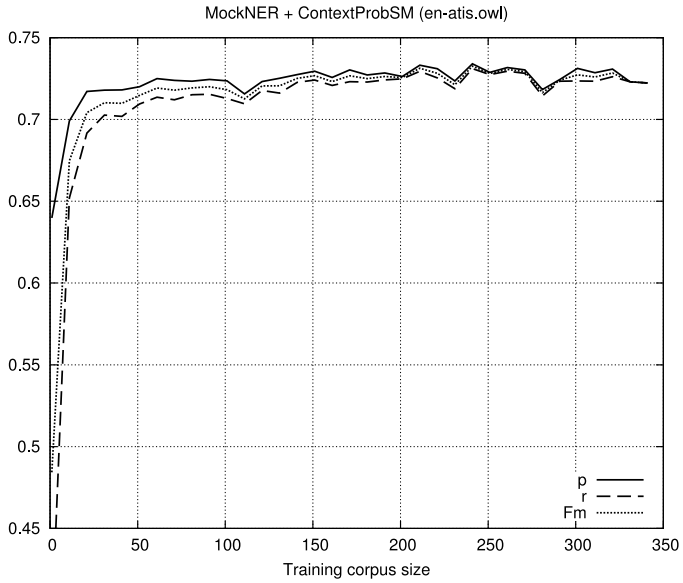

Figure 8.7: The evaluation of the semantic analysis on the ATIS corpus using different training set sizes.

of-the-art semantic parsers, basically for two reasons. First, our semantic annotation is different from the original frame-based ATIS annotation. Second, we use a small subset of the whole corpus which also negatively affects the performance. The complete corpus was not used because a manual annotation according to our semantic represenation was required and it was not feasible to annotate the complete corpus.

The obtained results illustrate that our semantic analysis model can be used across different languages. No tweaking is required to adapt the system to English. The achieved performance (about 0.73 F-measure) is satisfactory, given the fact that the system was primarily developed on a different domain and language.

Chapter 9

Conclusion

This final chapter outlines the major contributions of this thesis as well as open issues and a future work. The Semantic Web Search using Natural Language is, beyond all doubt, a challenging task and many theoretical and practical problems must be solved in order to develop a real-world system. We will examine these issues from many perspectives and also propose solutions to some of them.

9.1 Open Issues

9.1.1 Performance Issues

One of the most critical issues which actually prevents our system from being tested in the real Web environment is the *performance of the ontology reasoning.* As already shown in Table 7.1 on page 73, we tested several OWL reasoners and their ability to deal with transitive properties.

The *Pellet* reasoner requires approximately 15 minutes on average to answer a single SPARQL query with transitive relations. We suspect that the reason is the size of our KB (281.686 triplets, see section 4.3.3). However, to our best knowledge, there is no room for improvement without changing the back-end Semantic Web tools completely.

Possible solutions. First, another ontology storage engine can be used, e.g. Sesame[1]. Second, the back-end can be completely switched to a relational database which would, however, disable reasoning capabilities of the system.

[1] `http://www.openrdf.org/`

9.1.2 Deployment and Development

Whereas relational databases provide stable, scalable, robust, and flexible solution for storing large-scale data, many Semantic Web technologies are still far from being suitable for deployment in the commercial sector. Although the Semantic Web vision is now more than ten years old, many of the software tools are intended only for basic research and are impractical for any real work. This is the case of e.g. Protégé which is one of the most popular ontology designer, yet, it still has very poor usability.

Moreover, there is a lack of good guidelines for developers who are new to the Semantic Web field. After reading e.g. (Hebeler, Fisher, Blace, Perez-Lopez, & Dean, 2009) or (Segaran, Taylor, & Evans, 2009), developers get an idea how the Semantic Web *should* work in an ideal case but the programming experience is then very different. Developers must often rely on a small user base, obsolete or non-existing documentation, or reading the source code of libraries.

9.1.3 Problems Caused by Real Web Data

Many practical issues related to the structure and the content of the KB was pointed out in section 4.3.3. The most important problem is the missing structured data. In the other words, the data from the source Web site are incomplete. However, this will be the case of most of the systems that use public Web sources for populating their KBs. It is not feasible either to check or even to correct the data manually.

Another problem is caused by the inconsistent source data. In our scenario, various KB instances have the same label which makes the mapping from the named entities to ontology instances much harder (see section 7.1.1).

Possible solutions. A better data post-processing after the information extraction step. Nevertheless, a huge manual effort would be required.

9.1.4 Research-related Issues

Since the developed SWSNL system is unique in many ways, it is not possible to compare its results with any other system. However, this is not the case of our system only. As pointed out in section 2.4 on page 14, the whole research of NLISW or NLIDB suffers from the lack of standard test sets and performance measures.

Possible solutions. A sort of consensus in the Semantic Web community is necessary to formulate the goals of the Semantic Web search more precisely. Furthermore, a dataset should be provided, together with clear and reasonable evaluation criteria.

9.1.5 Use in the Business Sector

This thesis presents a proof-of-concept of an interesting and promising technology with a good commercial potential. However, before the system is used in the business sector, a lot of questions must be answered.

The main problems are the quality and the source of the domain data. While the data can be carefully prepared for a proof-of-concept development, this is not feasible for a system dealing with various sources and operating on different domains.

In the others words, the ontology, the KB or the database (anything, where the domain data is stored) is the most valuable commodity for the Web portals. If a company runs a business by e.g. providing a Web portal in the accommodation domain, it profits from *presenting the data on its website*, including advertisements, hot-deals, etc. By providing a data access to third parties, this business model fails. In our opinion, no company will provide highly valuable and high-quality domain KB for free. The idea about interchanging information using ontologies in the Semantic Web misses its business model aim.

9.2 Future Work

Basically, there are four directions that are worth further exploring:

- *Larger NL question corpus.* The main limitation of our current semantic model performance is the small size of the corpus. Thus, it would be beneficial to obtain more data in e.g. a bootstrapping manner when the prototype is deployed on a testing server and open to the public access.
- *Better Named Entity Recognition.* The results definitely show that the NER component is crucial in the semantic search. The better the NER is, the better the semantic model performs and the the more precise the search results are.
- *Performance Improvement.* Replacing the back-end with more scalable Semantic Web tools or with a relational database.

- *Integrating the fulltext search.* The combination of the structured search and the fulltext search is a promising future work, based upon our preliminary research.

9.3 Final Conclusion

This thesis describes a complete end-to-end system for the Semantic Web search using a Natural Language. The work is put into the context of the state-of-the-art systems for Natural Language Understanding and Natural Language Interfaces to Semantic Web. It presents a complete work-flow including a data preparation, a natural language corpus, an ontology design, an annotation, a semantic model and a search. It contains very detailed evaluation with promising results.

9.3.1 Major Contributions

- The ontology-based question semantics independent of the ontology for storing the domain knowledge.
- The statistical model for the semantic analysis based upon supervised training.
- The evaluation of the fully functional end-to-end system with a real Web data and real user queries.

9.3.2 Review of Aims of Ph.D. thesis

The following tasks are taken from author's Ph.D. thesis exposé (Habernal, 2009).

- Continue in the work of (Konopík, 2009), study the system, obtain additional semantically annotated data and measure the performance of the system.
- Compare the system to another existing systems. Explore the possibilities to obtain a standard semantic corpus (e.g. ATIS) and evaluate the system using this data.

These two goals were accomplished successfully, see Appendix A. The semantic analysis system from (Konopík, 2009) was evaluated on the ATIS corpus with encouraging results that were published in the proceedings of the international conference Advanced Data Mining and Applications 2010.

- Propose and evaluate novel methods in order to improve the semantic analysis system. Focus on robust processing of a faulty data. Consider specific properties of the Czech language.
- Experiment with the use of the developed system for the semantic web search using a natural language.

These two goals are covered in the main content of this thesis and they were fulfilled completely. First, a new formalism for describing the semantics of natural language questions was introduced. Second, a statistical semantic analysis model based upon supervised learning was proposed. Third, a complete end-to-end Semantic Web search system using a natural language was developed. Finally, the system was thoroughly evaluated on a real Web data with very promising results.

Appendix A

Hybrid Semantic Analysis System – ATIS Data Evaluation*

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.

*This appendix is the author's conference paper (Habernal & Konopík, 2010). As outlined on page 88, one of the thesis goal was to *"[...] Explore the possibilites to obtain a standard semantic corpus (ie. ATIS) and evaluate the system using this data."*. This task was successfully fulfilled resulting into the following conference paper. However, there are two reasons why this achievement is not presented in the main part of the thesis. First, the semantic analysis system used in this paper is based on a work of (Konopík & Habernal, 2009) and it is not related to the semantic analysis system developed in this thesis. Secondly, the semantic annotation of ATIS corpus uses a tree representation, whereas the main semantic framework in this thesis is based on ontologies and Semantic Web technology. The conference paper is left without editing to demonstrate that one of the thesis goals was accomplished successfully.

A.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 (Pieraccini et al., 1992) and the HVS model (He & Young, 2005).

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 (D. Zhou & He, 2009).

A.2 Related Work

A significant system for stochastic semantic analysis is based on HVS model (hidden vector-state model) (He & Young, 2005). The system was tested on the ATIS and DARPA corpora, recently the system was also used for semantic extraction from bioinformatics corpus Genia. 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 (D. Zhou & He, 2009) 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 (Jurčíček, 2007). 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

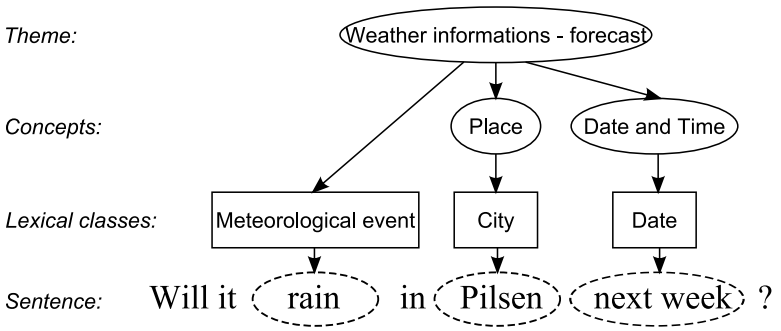

Figure A.1: An example of a semantic annotation tree.

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 (Y.-Y. Wang et al., 2005), 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.

A.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 A.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 (He & Young, 2005). 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.

A.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.

A.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 A.3). More information about the data can be found in (Konopík, 2009).

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

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

A.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. (He & Young, 2005), (Iosif & Potamianos, 2007), (Jeong & Lee, 2008) and (C. Raymond & Riccardi, 2007). The original ATIS corpus is divided into several parts: ATIS2 train, ATIS3 train, two test sets etc. (Dahl et al., 1995).

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 A.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 (He & Young, 2005), 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 A.3). Using the methodology described in (Habernal & Konopík, 2009) 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.

A.5 System Description

The system consists of three main blocks (see Figure A.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 A.5.1 and the *probabilistic parsing* is explained in Section A.5.2.

A.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 A.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

[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 (Habernal & Konopík, 2009).

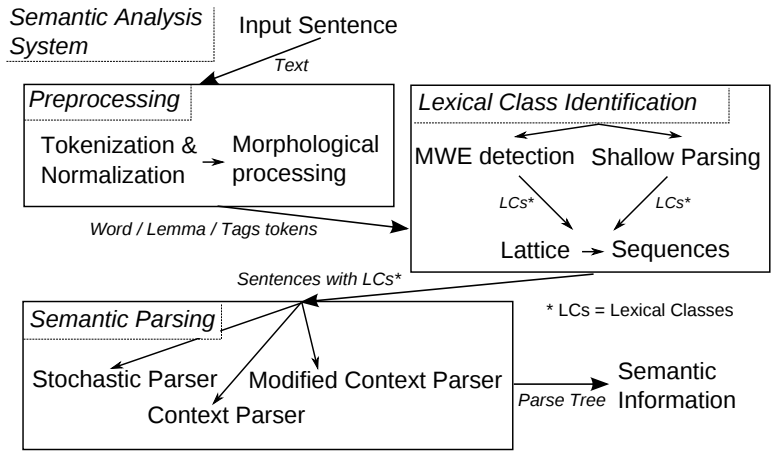

Figure A.2: The structure of our semantic analysis system.

is converted into possible sequences of lexical classes. Only the sequences that contain no overlapping classes are created.

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 (Knuth, 1997) 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 (Konopík & Habernal, 2009) in details.

For the analysis of complicated phrases, such as dates, numbers, time, etc., the LINGVOParser (Habernal & Konopík, 2008) 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 (Habernal & Konopík, 2008)).

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 (Konopík & Habernal, 2009).

A.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 A.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 A.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 A.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:

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

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. (Allen, 1995)) 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:

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

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:

$$\hat{T} = \underset{i}{\operatorname{argmax}} P(S_i) P(T_i) \ , \tag{A.3}$$

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 A.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:

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

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:

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

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 A.2 is reformulated to consider the context:

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

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

$$P(\hat{T}) = \underset{i}{\text{argmax}}\, P(S_i) \prod_j (P(w_j|S)P(T_i) , \tag{A.7}$$

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 (Grepl & Karlík, 1998) 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.

A.6 Performance Tests

A.6.1 Results on the LINGVOSemantics corpus

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

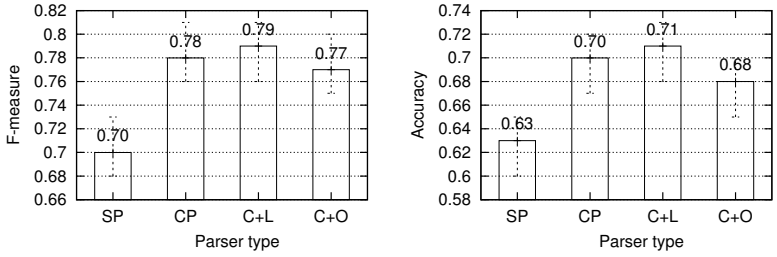

Figure A.3: Results of Semantic Parsers. *SP* = Stochastic Parser, *CP* = Context Parser, *C+L* = Context + Lemma, *C+O* = Context + word Order. Confidence intervals are computed at confidence level: $\alpha = 95\%$.

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 (He & Young, 2005). We use the same metrics as in (He & Young, 2005) for sake of mutual comparison of our results and the results in (He & Young, 2005).

A.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 A.5.1). Aditionally, the multiword expression identificator was re-trained using the data from ATIS `*.tab` files that contain cities, airports, etc. Second, the semantic parser was trained using the training set described in section A.4.2.

Figure A.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 (He & Young, 2005) (89.28%) or (D. Zhou & He, 2009) (91.11%), we must consider that in (He & Young, 2005) and in (D. Zhou & He, 2009) their systems were trained on a larger training set (4978 utterances). The reasons that we used a smaller set are explained in section A.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

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

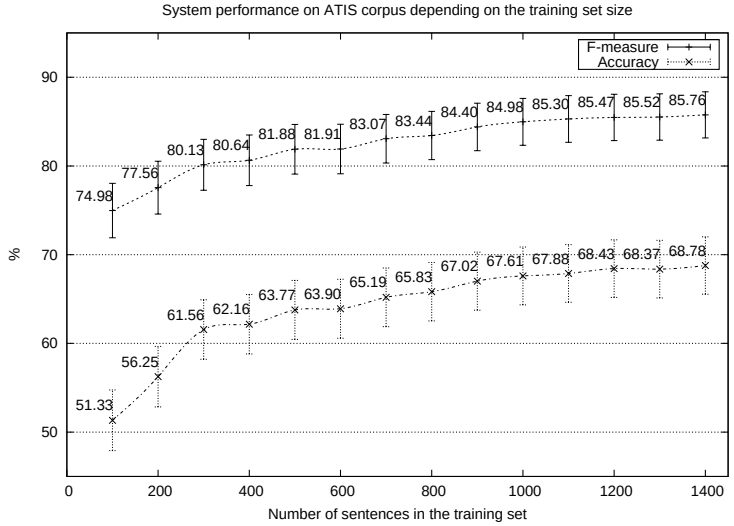

Figure A.4: System performance on the ATIS corpus with various amount of training data. Confidence intervals are computed at confidence level: $\alpha = 95\%$.

that the performance testing on this corpus is affected by the testing set inconsistency and the objectivity of the evaluation is compromised.

A.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 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 (D. Zhou & He, 2009)) 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 (D. Zhou & He, 2009) 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.

Bibliography

Allen, J. (1995). *Natural language understanding (2nd ed.)*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases – an introduction. *Natural Language Engineering*, *1*, 29–81.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284*(5), 34-43.

Beuschel, C., Minker, W., & Bühler, D. (2005). Hidden markov modeling for semantic analysis – on the combination of different decoding strategies. *Internation Journal of Speech Technology*, *8*(3), 295–305.

Bonneau-Maynard, H., Rosset, S., Ayache, C., Kuhn, A., & Mostefa, D. (2005). Semantic annotation of the french media dialog corpus. In *Interspeech 2005 - eurospeech, 9th european conference on speech communication and technology* (p. 3457-3460). ISCA.

Broekstra, J., & Kampman, A. (2003). SeRQL: A second generation RDF query language. In *SWAD-Europe workshop on semantic web storage and retrieval.*

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st north american chapter of the association for computational linguistics conference* (pp. 132–139). Stroudsburg, PA, USA: Association for Computational Linguistics.

Cimiano, P., Haase, P., Heizmann, J., Mantel, M., & Studer, R. (2008). Towards portable natural language interfaces to knowledge bases – the case of the orakel system. *Data and Knowledge Engineering*, *65*(2), 325 - 354.

Cimiano, P., & Minock, M. (2010). Natural language interfaces: What is the problem? – a data-driven quantitative analysis. In H. Horacek, E. Métais, R. Muñoz, & M. Wolska (Eds.), *Natural language processing and information systems* (p. 192-206). Springer Berlin / Heidelberg.

Codd, E. F. (1974). Seven steps to rendezvous with the casual user. In *Ifip working conference data base management* (p. 179-200).

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th annual meeting of the association for*

computational linguistics (pp. 16–23).

Copestake, A., Flickinger, D., Pollard, C., & Sag, I. (2005). Minimal recursion semantics: An introduction. *Research on Language and Computation*, *3*, 281-332.

Cunningham et al. (2011, April). Text processing with gate (version 6) [Computer software manual]. (University of Sheffield Department of Computer Science)

Dahl, D., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., et al. (1995). *ATIS3 Test Data.* (Linguistic Data Consortium, Philadelphia)

Damljanovic, D., Agatonovic, M., & Cunningham, H. (2010). Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In L. Aroyo et al. (Eds.), *The semantic web: Research and applications* (p. 106-120). Springer Berlin / Heidelberg.

Damljanović, D., & Bontcheva, K. (2009). Towards enhanced usability of natural language interfaces to knowledge bases. In V. Devedžić & D. Gašević (Eds.), *Web 2.0 & semantic web* (p. 105-133). Springer US.

Dean, M., & Schreiber, G. (2004). *OWL web ontology language reference* (W3C Recommendation). W3C.

Dolamic, L., & Savoy, J. (2009). Indexing and stemming approaches for the czech language. *Information Processing & Management*, *45*(6), 714 - 720.

Fazzinga, B., & Lukasiewicz, T. (2010). Semantic search on the web. *Semantic Web*, *1*(1-2), 89-96.

Ferrández Óscar, Izquierdo, R., Ferrández, S., & Vicedo, J. L. (2009). Addressing ontology-based question answering with collections of user queries. *Information Processing & Management*, *45*(2), 175 - 188.

Frank, A., Krieger, H.-U., Xu, F., Uszkoreit, H., Crysmann, B., Jörg, B., et al. (2007). Question answering from structured knowledge sources. *Journal of Applied Logic*, *5*(1), 20 - 48.

Gao, M., Liu, J., Zhong, N., Chen, F., & Liu, C. (2011). Semantic mapping from natural language questions to OWL queries. *Computational Intelligence*, *27*(2), 280–314.

Ge, R., & Mooney, R. J. (2005, June). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the ninth conference on computational natural language learning* (pp. 9–16).

Grepl, M., & Karlík, P. (1998). *Skladba češtiny* (1st ed.). Olomouc, Czech Republic: Votobia.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition.*, *5*(2), 199–220.

Habernal, I. (2009, May). *Stochastic semantic analysis* (Technical Report

No. DCSE/TR-2009-04). Department of Computer Science and Engineering, University of West Bohemia. (PhD Study Report)

Habernal, I., & Konopík, M. (2008). Active tags for semantic analysis. In *Proceedings of the 11th international conference on text, speech and dialogue* (pp. 69–76). Berlin, Heidelberg: Springer-Verlag.

Habernal, I., & Konopík, M. (2009). Semantic annotation for the lingvosemantics project. In *Proceedings of the 12th international conference on text, speech and dialogue* (pp. 299–306). Berlin, Heidelberg: Springer-Verlag.

Habernal, I., & Konopík, M. (2010). Hybrid semantic analysis system - ATIS data evaluation. In *Proceedings of the 6th international conference on advanced data mining and applications - volume part ii* (pp. 376–386). Berlin, Heidelberg: Springer-Verlag.

Habernal, I., Konopík, M., & Rohlík, O. (2012, May). Question Answering. In C. Jouis, I. Biskri, J.-G. Ganascia, & M. Roux (Eds.), *Next generation search engines: Advanced models for information retrieval.* IGI Global. (in press)

Habernal, I., & Konopík, M. (2010). On the way towards standardized semantic corpora for development of semantic analysis systems. In *Semapro 2010: The fourth international conference on advances in semantic processing* (pp. 96–99). IARIA.

Hajič, J., Panevová, J., Hajičová, E., Panevová, J., Sgall, P., Pajas, P., et al. (2006). *Prague dependency treebank 2.0.* (Linguistic Data Consortium, Philadelphia)

Hallett, C. (2006). Generic querying of relational databases using natural language generation techniques. In *Proceedings of the fourth international natural language generation conference* (pp. 95–102). Stroudsburg, PA, USA: Association for Computational Linguistics.

He, Y., & Young, S. (2005). Semantic processing using the hidden vector state model. *Computer Speech & Language*, *19*(1), 85-106.

He, Y., & Young, S. (2006a, Sept). A clustering approach to semantic decoding. In *9th international conferennce on spoken language processig (interspeech 2006 — icslp)* (pp. 17–21). Pittsburgh, USA.

He, Y., & Young, S. (2006b). Spoken language understanding using the hidden vector state model. *Speech Communication*, *48*(3-4), 262-275.

Hebeler, J., Fisher, M., Blace, R., Perez-Lopez, A., & Dean, M. (2009). *Semantic web programming.* Indianapolis, IN: Wiley.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., & Slocum, J. (1978, June). Developing a natural language interface to complex data. *ACM Trans. Database Syst.*, *3*, 105–147.

Horrocks, I., Patel-Schneider, P. F., & Harmelen, F. van. (2003). From SHIQ and RDF to OWL: the making of a web ontology language. *Web Semantics*, *1*(1), 7 - 26.

Iosif, E., & Potamianos, A. (2007, August). A soft-clustering algorithm

for automatic induction of semantic classes. In *Interspeech-07* (pp. 1609–1612). Antwerp, Belgium.

Jeong, M., & Lee, G. G. (2008, April). Practical use of non-local features for statistical spoken language understanding. *Computer Speech and Language*, *22*(2), 148–170.

Jurafsky, D., & Martin, J. H. (2008). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall.

Jurčíček, F. (2007). *Statistical approach to the semantic analysis of spoken dialogues.* Unpublished doctoral dissertation, University of West Bohemia, Faculty of Applied Sciences.

Kate, R. J. (2009). *Learning for semantic parsing with kernels under various forms of supervision.* Unpublished doctoral dissertation, Department of Computer Sciences, University of Texas at Austin.

Kate, R. J., & Mooney, R. J. (2006). 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* (pp. 913–920). Morristown, NJ, USA: Association for Computational Linguistics.

Kaufmann, E., & Bernstein, A. (2007). How useful are natural language interfaces to the semantic web for casual end-users? In *Proceedings of the 6th international the semantic web and 2nd asian conference on asian semantic web conference* (pp. 281–294). Berlin, Heidelberg: Springer-Verlag.

Kaufmann, E., Bernstein, A., & Fischer, L. (2007, November). NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies. In *4th european semantic web conference (eswc 2007)* (p. 1-2).

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting on association for computational linguistics - volume 1* (pp. 423–430). Stroudsburg, PA, USA: Association for Computational Linguistics.

Knuth, D. (1997). *The art of computer programming, volume 3: Sorting and searching, second edition.* Addison-Wesley.

Konkol, M., & Konopík, M. (2011). Maximum entropy named entity recognition for czech language. In *Proceedings of the 14th international conference on text, speech and dialogue* (pp. 203–210). Berlin, Heidelberg: Springer-Verlag.

Konopík, M. (2009). *Hybrid semantic analysis.* Unpublished doctoral dissertation, University of West Bohemia, Faculty of Applied Sciences.

Konopík, M., & Habernal, I. (2009). Hybrid semantic analysis. In *Proceedings of the 12th international conference on text, speech and dialogue* (pp. 307–314). Berlin, Heidelberg: Springer-Verlag.

Lassila, O., & McGuinness, D. (2001). The role of frame-based representation on the semantic web. In *Linköping electronic articles in computer and information science* (Vol. 6).

Libkin, L., & Wong, L. (1997). Query languages for bags and aggregate functions. *Journal of Computer and System Sciences*, *55*(2), 241 - 272.

Liu, B. (2011). *Web data mining: Exploring hyperlinks, contents, and usage data* (2nd ed.). Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Lopez, V., Fernández, M., Motta, E., & Stieler, N. (2011). PowerAqua: Supporting users in querying and exploring the Semantic Web. *Semantic Web*, 1–17.

Lopez, V., Uren, V., Motta, E., & Pasin, M. (2007). AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics*, *5*(2), 72 - 105.

Maedche, A., & Staab, S. (2001, March). Ontology learning for the semantic web. *IEEE Intelligent Systems*, *16*(2), 72–79.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press.

Manola, F., & Miller, E. (Eds.). (2004). *RDF Primer*. W3C. (http://www.w3.org/TR/rdf-primer/)

Miller, S., Bobrow, R., Ingria, R., & Schwartz, R. (1994). Hidden understanding models of natural language. In *Proceedings of the 32nd annual meeting on association for computational linguistics* (pp. 25–32). Morristown, NJ, USA: Association for Computational Linguistics.

Mooney, R. J. (2007, February). Learning for semantic parsing. In *Computational linguistics and intelligent text processing: Proceedings of the 8th international conference, cicling 2007* (pp. 311–324). Berlin, Germany: Springer.

Mykowiecka, A., Marciniak, M., & Głowińska, K. (2008). Automatic semantic annotation of polish dialogue corpus. In *Tsd '08: Proceedings of the 11th international conference on text, speech and dialogue* (pp. 625–632). Berlin, Heidelberg: Springer-Verlag.

Nguyen, L.-M., Shimazu, A., & Phan, X.-H. (2006). Semantic parsing with structured svm ensemble classification models. In *Proceedings of the coling/acl on main conference poster sessions* (pp. 619–626). Morristown, NJ, USA: Association for Computational Linguistics.

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, *29*(1), 19–51.

Pieraccini, R., Tzoukermann, E., Gorelov, Z., Levin, E., Lee, C.-H., & Gauvain, J.-L. (1992). Progress report on the chronus system: Atis benchmark results. In *Proceedings of the workshop on speech and natural language* (pp. 67–71). Stroudsburg, PA, USA: Association for Computational Linguistics.

Pla, F., Molina, A., Sanchis, E., Segarra, E., & García, F. (2001). 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* (pp. 403–409). London, UK: Springer-Verlag.

Pollard, C., & Sag, I. A. (1988). *Information-based syntax and semantics: Vol. 1: fundamentals.* Stanford, CA, USA: Center for the Study of Language and Information.

Pollard, C., & Sag, I. A. (1994). *Head-driven phrase structure grammar. studies in contemporary linguistics.* Chicago, IL/London: The University of Chicago Press.

Popescu, A.-M., Etzioni, O., & Kautz, H. (2003). Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on intelligent user interfaces* (pp. 149–157). New York, NY, USA: ACM.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J., & Jurafsky, D. (2004). 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.

Price, P. J. (1990). Evaluation of spoken language systems: the atis domain. In *Proceedings of the workshop on speech and natural language* (pp. 91–95). Stroudsburg, PA, USA: Association for Computational Linguistics.

Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL Query Language for RDF.* W3C Recommendation.

Raymond, C., & Riccardi, G. (2007, August). Generative and discriminative algorithms for spoken language understanding. In *Interspeech-07* (pp. 1605–1608). Antwerp, Belgium.

Raymond, C., Rodriguez, K. J., & Riccardi, G. (2008, May). Active annotation in the luna italian corpus of spontaneous dialogues. In *Proceedings of the sixth international language resources and evaluation (lrec'08).* Marrakech, Morocco: European Language Resources Association (ELRA).

Raymond, R. G., & Mooney, J. (2006). Discriminative reranking for semantic parsing. In *Proceedings of the coling/acl* (pp. 263–270). Morristown, NJ, USA: Association for Computational Linguistics.

Ruiz-Martínez, J., Castellanos-Nieves, D., Valencia-García, R., Fernández-Breis, J., García-Sánchez, F., Vivancos-Vicente, P., et al. (2009). Accessing touristic knowledge bases through a natural language interface. In D. Richards & B.-H. Kang (Eds.), *Knowledge acquisition: Approaches, algorithms and applications* (Vol. 5465, p. 147-160). Springer Berlin / Heidelberg.

Schwartz, R., Miller, S., Stallard, D., & Makhoul, J. (1997). 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 (p. 1479). Washington, DC, USA: IEEE Computer Society.

Schwitter, R. (2010). Creating and querying formal ontologies via controlled natural language. *Applied Artificial Intelligence*, *24*(1-2), 149–174.

Segaran, T., Taylor, J., & Evans, C. (2009). *Programming the semantic web*. Cambridge, MA: O'Reilly.

Sha, F., & Pereira, F. (2003). 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* (pp. 134–141). Morristown, NJ, USA: Association for Computational Linguistics.

Spanos, D.-E., Stavrou, P., & Mitrou, N. (2012). Bringing relational databases into the semantic web: A survey. *Semantic Web Journal*. (to appear)

Stratica, N., Kosseim, L., & Desai, B. C. (2005). Using semantic templates for a natural language interface to the CINDI virtual library. *Data & Knowledge Engineering*, *55*(1), 4 - 19.

Svec, J., Jurčíček, F., & Müller, L. (2007). Input parameterization of the hvs semantic parser. *Lecture Notes in Artificial Intelligence*, 415-422.

Tang, L. R., & Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th european conference on machine learning* (pp. 466–477). London, UK: Springer-Verlag.

Thompson, C. W., Pazandak, P., & Tennant, H. R. (2005, November). Talk to your semantic web. *IEEE Internet Computing*, *9*(6), 75–78.

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In W. Daelemans & M. Osborne (Eds.), *Proceedings of conll-2003* (pp. 142–147). Edmonton, Canada.

Tur, G., Hakkani-Tur, D., & Heck, L. (2010). What is left to be understood in atis? In *Spoken language technology workshop (slt), 2010 ieee* (p. 19 -24).

Wang, C., Xiong, M., Zhou, Q., & Yu, Y. (2007). Panto: A portable natural language interface to ontologies. In *Proceedings of the 4th european conference on the semantic web: Research and applications* (pp. 473–487). Berlin, Heidelberg: Springer-Verlag.

Wang, Y.-Y., Deng, L., & Acero, A. (2005). An introduction to statistical spoken language understanding. *IEEE Signal Processing Magazine*, *22*(5), 16–31.

Ward, W., & Issar, S. (1994). Recent improvements in the cmu spoken language understanding system. In *Hlt '94: Proceedings of the workshop on human language technology* (pp. 213–216). Morristown, NJ, USA: Association for Computational Linguistics.

Warren, D. H. D., & Pereira, F. C. N. (1982, July). An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, *8*(3-4), 110–122.

Weischedel, R. M. (1989). A hybrid approach to representation in the janus natural language processor. In *Proceedings of the 27th annual meeting on association for computational linguistics* (pp. 193–202). Stroudsburg, PA, USA: Association for Computational Linguistics.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the section on survey research (american statistical association)* (pp. 354–359).

Wong, Y. W., & Mooney, R. J. (2006). 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* (pp. 439–446). Morristown, NJ, USA: Association for Computational Linguistics.

Young, S. (2002). *The statistical approach to the design of spoken dialogue systems* (Tech. Rep.). Cambridge, UK: University of Cambridge: Department of Engineering.

Zenz, G., Zhou, X., Minack, E., Siberski, W., & Nejdl, W. (2009). From keywords to semantic queries–incremental query construction on the semantic web. *Web Semantics*, *7*(3), 166 - 176.

Zhou, D., & He, Y. (2009). Discriminative training of the hidden vector state model for semantic parsing. *IEEE Trans. on Knowl. and Data Eng.*, *21*(1), 66–77.

Zhou, G., & Su, J. (2005). Machine learning-based named entity recognition via effective integration of various evidences. *Natural Language Engineering*, *11*(2), 189–206.

List of Published Articles

Book Chapters

- Habernal, I., Konopík, M., Rohlík, O. *Question Answering.* In C. Jouis, B. Ismail, G. Jean-Gabriel, & R. Magali (Eds.), Next generation search engines: Advanced models for information retrieval. March, 2012. IGI Global Selected Conference Papers

Journal Articles

- Habernal, I., Konopík, M. *SWSNL: Semantic Search Using Natural Language.* Manuscript submitted to Journal of Web Semantics in May 2012.

Selected Conference Papers

- Habernal, I., Konopík, M. *Hybrid Semantic Analysis System - ATIS Data Evaluation.* In Advanced Data Mining and Applications - Part II., ADMA2010, Chongqing, China. Berlin: Springer, 2010. s. 376-386. ISBN: 978-3-642-17312-7
- Habernal, I., Konopík, M. *On the Way towards Standardized Semantic Corpora for Development of Semantic Analysis Systems.* In SEMAPRO 2010. Florence, Italy: IARIA, 2010. s. 96-99. ISBN: 978-1-61208-000-0
- Konopík, M., Habernal, I. *Hybrid Semantic Analysis.* In Text, Speech and Dialogue. Berlin: Springer, 2009. s. 307-314. ISBN: 978-3-642-04207-2
- Habernal, I., Konopík, M. *Semantic Annotation for the LingvoSemantics Project.* In Text, Speech and Dialogue. Berlin: Springer, 2009. s. 299-306. ISBN: 978-3-642-04207-2
- Habernal, I., Konopík, M. *Active Tags for Semantic Analysis.* Lecture Notes in Artificial Intelligence, 2008, 5246, s. 69-76. ISSN: 0302-9743
- Habernal, I., Konopík, M. *JAAE: The Java Abstract Annotation Editor.* In Proceedings of Interspeech 2007. Bonn: ISCA, 2007. s. 1973-1976. ISBN: 978-1-60560-316-2