MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Automatic Syntactic Analysis for Real-World Applications

PHD THESIS

Vojtěch Kovář

Brno, Spring 2014

Declaration

Hereby I declare, that this thesis is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Acknowledgement

I would like to give thanks to my supervisor Aleš Horák, my former supervisor Karel Pala and all my colleagues from the NLP Centre for cooperation and great working atmosphere. Many thanks also belong to my wife Jana, two daughters and the whole family for support and tolerance, without which this work would never be written. Last but not least, thanks to my colleagues and superiors from Lexical Computing for their patience and for all the tasks that they did instead of me during writing this work.

Abstract

Syntactic analysis (parsing) of natural languages is a subfield of natural language processing (NLP) that is often claimed to be a "corner stone" of the area, a necessary base for any advanced language processing and real understanding. Syntactic analysis deals with revealing the sentence structure, language units bearing meaning and relationships among them; it is hard to imagine real language understanding without this information. On the other hand, in current practical "intelligent" applications, syntactic processing is often substituted by purely stochastic methods. There are even visible opinions in the NLP community claiming that syntactic analysis is not really needed in practical applications.

In this work, we analyse the current status of the field and identify the particular problems that it is suffering from. Based on this analysis, we propose next directions that the research of parsing should accommodate. We discuss methodology and evaluation, manual annotation procedures and approaches to design and implementation of the natural language parsing tools.

Then, we describe results of our research within these directions. They include a new format for manual syntactic annotation and two application oriented tools for automatic syntactic analysis. Many application usages of these tools are reported which supports our methodology considerations. Evaluation of the practical outputs of the work is provided, and evaluation methodology problems are discussed.

Keywords

syntactic analysis, Czech, parser, SET, application, bushbank, EFA, punctuation detection, word sketch, terminology, authorship, authorship verification, grammar checking, collocation extraction, treebank, sketch grammar, information extraction

Contents

Chapter 1

Introduction: Natural language processing challenges

As the use of digital technologies becomes more and more integral part of our lives, there is a strong need for people who can understand machines, as well as machines that can understand people (at least a bit). The development of such machines is the object of research in many scientific and technical fields, ranging from human-computer interaction and virtual reality, through design of graphical interfaces, to artificial intelligence and natural language processing. The aim of natural language processing (NLP) is to learn how natural languages work and exploit this knowledge in practical applications that will serve for easier everyday communication with our machine partners.

Although we might not notice, we work with NLP applications every day: We write our text messages with T9, we listen to synthetic speech at railway stations, we google for instant information, we let Word check our language by spelling checker, we use Google Translate for languages that we do not understand… All of these applications exploit a certain level of understanding of how language works. On the other hand, there are desirable applications for which we do not have enough knowledge yet: You cannot chat with a bus ticket automaton about the best way to get to your old friend whose address you do not know exactly (although sometimes you can use a clever interactive map), the Google machine translation still makes severe stupid mistakes, the spelling checker is not able to check your style nor reasonability of your text… Although the current applications may seem relatively intelligent, there is a lot of potentially useful things that are still not possible with the current level of knowledge.

Syntactic analysis (parsing) is a subfield of NLP that is often claimed to be a "corner stone" of the area, a necessary base for any advanced language processing and real understanding. Syntactic analysis deals with revealing the sentence structure, language units bearing meaning and relationships among them, and it is hard to imagine real language understanding without this information. On the other hand, it is rarely used within current practical

applications and there are visible opinions in the NLP community claiming that it is not really needed.

This looks like an interesting discrepancy between theory and practice, amplified by the amount of effort that has already been invested into the research of automatic syntactic analysis. This research has its own dedicated conferences; hundreds of different parsing algorithms and approaches to syntactic analysis were described; millions of dollars have been invested worldwide into manually annotated syntactic data for training and testing... without a visible success in form of a widely used application. One can say it is a "l'art pour l'art" in the world of natural language processing, at least up to now.

1.1 Syntactic analysis of formal and natural languages

Syntactic analysis of natural languages is different from syntactic analysis of formal languages (e.g. programming languages or formal descriptions). The main difference is massive ambiguity of natural languages on all levels of analysis – words, as well as sentences can have multiple meanings. In sentence "I saw a man with a telescope", the syntactic interpretation decides the meaning of the sentence – if it is me who holds the telescope, or the other man. This is an illustration of the famous "PP-attachment" problem – if the prepositional phrase (PP) "with a telescope" should be attached to "saw" or to "man".

Such ambiguities are present in most natural language sentences. Most of them can be resolved correctly and even unconsciously by a human, but for automatic processing they present a complex problem. Besides other issues, such as dealing with large vocabulary of natural languages, syntactic analysis aims at resolving such ambiguities and providing correct structural information to following processing levels, namely analysis of meaning.

1.2 Goals of the work

In this work, we analyse current status of the field of natural language syntactic analysis and spell out the particular problems that it is suffering from. Based on this analysis, we propose new directions that the research should accommodate, in terms of methodology and evaluation, manual annotation procedures and preferable approaches to parser design.

We describe results of our research within these new directions, namely

- new approach to manual annotation of natural language syntax

- new parsing systems tailored to be used directly in various types of applications
- prototypes of the particular applications that have already made advantage of the syntactic information provided by the automatic tools

As we will see, the successful usage in applications strongly supports our initial methodology considerations, different from the usual state-of-the art approach.

1.3 Structure of the thesis

The text is structured as follows: In the introduction, we list some of the natural language applications that may benefit from syntactic information from natural language parsing. The list does not aim to be complete, its purpose is rather illustrative. The next chapter summarizes the main trends in the state of the art syntactic analysis and discusses its main problems. Based on this criticism, the following four chapters describe

- a new approach to manual syntactic annotation
- a specialized parsing machinery already used in a commercial system, and its further development
- a newly proposed parsing system
- usage of the parsing information within the practical applications listed in the introduction

These four chapters present the main contribution of this work. As some of the topics range across different sub-fields in the natural language processing area, the presented results are often collaborative; we always highlight this fact at the respective sections and provide the credits of the collaborators.

1.4 Author's background

The author of this thesis is based at the NLP Centre of the Faculty of Informatics at Masaryk University; as a student from 2004, as a research employee from 2006. Within the scope of that work, the author took part in several research projects exploiting and developing automatic syntactic analysis, funded by Czech government and European union. From the beginning, he was also involved in university collaboration with Lexical Computing Ltd., English commercial company that makes research related busi-

ness in the field of corpus and computational linguistics, with a main product called Sketch Engine.[1] From 2012, the author is also an employee of the Czech branch of the Lexical Computing company.

As firstly collaborator and then employee of this company, a significant part of author's work has been development of friendly user interfaces and communication with the customers – so within these 8 years, the author got very familiar with needs, ideas and attitudes of many language technology users. He has integrated that experience into this work as well, so the point of view presented in this text may be often a (hopefully contributive) mix of author's academic and commercial stances.

Based in the Czech Republic, the first natural language for doing research in natural language processing is (naturally) Czech. So it is in this work: the described systems and annotation principles were initially developed for Czech, then for English, then for other languages. Czech is a free word order, synthetic language from the Slavic language family, with very rich morphology (which may have influenced some of the initial considerations within the research). However, we are confident that the presented results are language independent to a significant extent, and we will discuss the aspect of universality of the achieved results in the respective sections.

1.5 Natural language processing challenges: A sample

This part provides a non-exhaustive overview of challenges in natural language processing – tasks that are not sufficiently solved (although partial solutions exist in most cases), and that can take advantage from automatic syntactic analysis, if it was in a suitable shape. The particular selection is motivated by the fact that we have been working on improving the tasks using automatic syntactic analysis.

1.5.1 Information retrieval

This is the "Google task": look up the documents most relevant to a given query. Google (followed by other search engines) addresses this problem successfully by combining metrics measuring the relevance of a document to a particular query, based on occurrences of query words (and their stems and synonyms) within a particular document [2], with the PageRank algorithm [135] sorting the best quality pages to the top.

1. www.sketchengine.co.uk

In time, people have accommodated to the way the search works and especially in the young generation, they typically do not have problems to find anything. Also, there are billions of searches per day[2] and huge commercial interests, so the system is set up really well within the scope of the current possibilities. However, there are still situations where more sophisticated processing would help to obtain more precise results, such as in case of queries like "market bulls". From the query, especially from the relationship between the two words, it is clear that we are not interested in information about bull markets and bear markets which are practically the only results we are offered, but rather information about the company named Market Bulls or business with real bulls, that we can get by a related query "market bulls cows".

Another room for making the search more intelligent lies in processing rather naturally formed questions, such as "Who defeated Lenin?". We would expect documents where Lenin is the object of defeating and the information about subject is present as well, e.g. something similar to what we can get for query "marshal who defeated Lenin", in quotes. Instead, we are shown a number of documents where either the subject is missing, or Lenin is not at all the object of defeating.

1.5.2 Information extraction

The goal of information extraction (not to be confused with information retrieval) is to extract formal knowledge, e.g. in form of RDF [76] or other computer database format, from free text in natural language. The information can be general ("extract all we can"), or very specialized, like spotting possible interactions among proteins from biochemical papers [118].

Although it may seem that syntactic relations such as subject, object, modifier etc. are necessary for gaining semantic relations that could be imported into a database, information from syntactic analysis is rarely used in this task. For illustration, the Wikipedia page for "Information extraction"[3] does not mention syntactic analysis at all; rather than that, simple regular expression approaches or machine learning methods are used extensively. There are only a few papers that report using automatic syntactic analysis within this task [118, 120]. Both of these papers reported improvements when using parsing, which supports our confidence that using syntactic

2. `www.statisticbrain.com/google-searches`
3. `en.wikipedia.org/wiki/Informationi_extraction` – 1700 words, visited in January 2014

analysis shall be beneficial in the field and should be investigated more thoroughly within this type of applications.

1.5.3 Question answering

Question answering whose aim is to generate proper answers to natural language questions from user, according to a knowledge base (possibly also in natural language, e.g. the web), relates to the information retrieval and information extraction applications. In some cases, it is understood in the same way as information retrieval with the only difference being the fact that the input is a question in natural language, rather than a query containing keywords. Also, most of the current question answering systems are based on this principle – they extract keywords from the input question, then use Google or another search engine, and simply return to the user what the search engine returns (e.g. [11]).

Our interpretation of the question answering task is more complex. We consider extracting the answer from the document as an integral part of the task; also the answer should be as short as possible, without losing meaning. More precisely, the correct answer is a string forming a phrase, clause, sentence or group of sentences, any substring of which is not judged as correct answer.

This part of the task is sometimes addressed by using the two-line snippets as the answer that Google (or other search engines) returns as part of the result list. These parts of the found documents always contain the keywords from the query, however, this does not imply that they contain the answer to the question, as illustrated on the Lenin example above, or similar simple queries in form of questions, such as "who has Koh-i-Noor?" – the Google results contain parts of the documents that contain the words "have" and "Koh-i-Noor" but they fail to find an answer, at least not in the first results. Obviously, some advanced processing is needed here, such as recognizing particular meaningful chunks in the document and labelling them with types of questions they may be able to answer. Complete solution of the question answering task would of course have to integrate many more advanced features than this, including inter-sentential inference, anaphora resolution. However, using labelled syntactic chunks instead of keywords may be the first step to improve the current question answering systems.

1.5.4 Automatic reasoning

The goal of automatic reasoning in context of computer science and logic usually means generating new formulas in a particular logical formalism and a particular theory, given certain assumptions. Alternatively, it can be viewed as verifying if a particular formula is correct – this is equivalent to checking if that formula is in the set of valid formulas generated from the assumptions. The best known example of such reasoning system is probably Prolog [25] that provides reasoning within the scope of predicate logic.

In context of natural language, automatic reasoning means roughly the same, but within natural language instead of a logical formalism; it is also called *textual entailment*. This means, generating new valid natural language sentences, according to a set of another sentences in natural language, sometimes referred to as knowledge base. A simple example follows:

- knowledge base: "Antonín Dvořák (September 8, 1841 – May 1, 1904) was a Czech composer."
- generated sentences:
  - "Antonín Dvořák is dead."
  - "Antonín Dvořák was born in 1841."
  - "Antonín Dvořák wrote some pieces of music."
  - "Antonín Dvořák was Czech."
  - ...

The induced information may be further used in all of the previously introduced challenges, making the information in the knowledge base (e.g. web) more dense and increasing probability of giving the right answer.

So far, existing systems for natural language reasoning are rather experimental. Among existing approaches, there is e.g.

- automatic translation of the natural language sentences into predicate logic, doing inferences in predicate logic and translating the results back to natural language [45, 117]
- the same process with more powerful Transparent intensional logic [50, 107]
- using manual or stochastic transformation rules directly within natural language, without the formal logic as connection [168, 126]

All of these approaches need flexible automatic syntactic analysis, in various shapes, as described in the respective papers. However, the task is far from being solved so there is plenty of room for improvement.

1.5.5 Authorship recognition

Authorship recognition and verification aims at reliable automatic assignment of author, to an anonymous piece of text (towards automatization of forensic linguistics). The verification subtask is to decide if two anonymous pieces of text were written by the same author or not. It is clear that the tasks are equivalent to certain extent: if we had examples of many author's writing and a reliable authorship verification application, we would be able to perform general authorship recognition; so we further specialize mainly on the authorship verification task in context of this work.

A more general concept is *stylometry* – rather than deciding a particular author, we may aim at different categories, e.g. age, education level or sex of the author. In most cases, same methods can be used for all tasks of this type.

Some research was done in this area, in most cases exploiting machine learning methods [79] to extract a *stylome* – a set of features that characterize the author (or the selected category) – from the given text. The features are usually linguistically motivated, such as ratio of short words, usage of specific words or parts of speech etc. Previous research also showed attempts to include syntactic information into the feature set [49], however, the results were not too convincing. In this work, we will present our results of incorporating syntactic information into an authorship verification project.

1.5.6 Grammar checking

Reliable checking people's writing for correctness is one of the important goals in natural language processing. Spelling checkers became a common part of our lives, but checking more complex language phenomena still presents a challenge. Although there are "grammar checkers" available in software packages like Microsoft Office or as stand-alone programs, they can address only a restricted range of grammar error types, and they are far from being able to find all the errors, wisely following the "minimum number of false alerts" philosophy.

In the Czech language, punctuation errors and mistakes in subject-predicate agreement belong to the most severe and most frequent errors people make, as there are complex and non-intuitive rules for both writing punctuation and correct usage of subject-predicate agreement. At the same time, these rules include numerous syntactic, semantic and pragmatic aspects which makes them very difficult to be formalized for automatic checking.

Punctuation detection and fixing errors in the Czech grammar is often used as a textbook example of how automatic syntactic analysis can be exploited for a prominent practical application. However, in real life, the full parsing is rarely used (and if, the results are not convincing [46]), and the current methods use rather various types of common error patterns or lightweight modifications of the full syntactic formalisms.

1.5.7 Collocation extraction

Collocation is a group of words that co-occur more often than would be expected by chance. Sometimes, they may indicate some degree of non-compositionality and may be used to find idioms [31], in other cases they just identify a common preference between the words. The most famous example of this phenomenon is probably "strong tea" vs. "powerful tea" – the latter is not correct English, although there is no obvious reason; the word "tea", unlike e.g. "computer", just prefers "strong" to "powerful".

It is very important for language learners to know the common collocations in the language, otherwise their communication may sound funny and incorrect. Collocations dictionaries were created for language learners [29, 146], the latter with help of automatic tools. For the same reason, collocations are important for natural language processing as well – e.g. in machine translation, collocations may disambiguate word sense [174] and help with the fluency of the translation (usually, statistical language models are used in this phase which can be considered a type of collocation information).

Automatic tools for collocation extraction are needed for dictionary creators as well as language learners, as there are many cases when a collocation dictionary is not available or its coverage is not sufficient. We will present state of the art methods for extracting collocations from natural language texts which make use of syntactic analysis, and our contribution to the field.

1.5.8 Terminology extraction

Terminology extraction is related to extraction of collocations; it can be considered a specialized version of collocation extraction. The task consists in extracting phrases from domain specific, often expert texts, that form the terminology of that domain or expert field. The automatic tools for terminology extraction are useful for creating specialized dictionaries, but mainly for domain specific translation work (both manual and automatic):

terms should be translated consistently and rarely there are sufficient terminology dictionaries.

We will show our approach to terminology extraction, with exploitation of automatic syntactic analysis, that has been evaluated and used within the scope of a commercial project.

1.5.9 Hidden applications

Apart from real world applications, some of which were introduced above, there are natural language processing tasks that are usually hidden from users, however, they are necessary for subsequent processing.

One of them is **morphology disambiguation** – selecting the proper morphological tag from the set of all possible tags for a particular word (e.g. to decide if "cut" is a verb or a noun). Many of the current applications depend on this processing, and although syntactic analysis is often considered to be one of them, it can also contribute to accuracy of the disambiguation [59].

Anaphora resolution, finding meanings (so called *antecedents*) of the referential expressions within texts [116], such as personal pronouns or unvoiced subjects in Czech, is one of the tasks that could raise accuracy of all natural language processing applications. Referential expressions are used in all natural languages for economical reasons, however, being able to interpret them correctly is necessary for advanced computer processing of language. From sentences and sentence groups like

"I love my grandfather and I won't see him being taken advantage of."

it would be desirable to extract e.g. a collocation candidate (see, grandfather), rather than (see, he); or we would like to infer from this sentence that "grandfather can be taken advantage of". Syntactic preprocessing is necessary for most of non-trivial anaphora resolution algorithms.

Another such application is **automatic extraction of semantic frames** [106] which aims at automatic production of frame semantic resources, such as FrameNet [5] or Verbalex [44]. These resources are not directly exploitable by common users, however, they are very important in complex natural language processing needed for advanced natural language reasoning, question answering or machine translation.

A related task is **extraction of lexical semantic information** such as contained in WordNet [115]. Again, the resource itself is not of intensive interest among crowds, however, it contains very useful information about

relationships among words, and its automatic extension or even creation would be very beneficial for advanced natural language processing.

Natural language synthesis is another important field of natural language processing. It seems straightforward – e.g. it works on railway stations for tens of years already. However, not all types of synthesis are that straightforward, and sometimes a deep analysis of input is needed, including syntactic analysis, to put the words in the right forms correctly together.

We have been working with the applications outlined above as well, within the scope of using syntactic analysis to gain better results. We will explain the procedures implemented and the initial results.

Chapter 2

Automatic syntactic analysis of natural languages

In this chapter, we describe the state of the art in the field of automatic natural language syntactic analysis. The field and its results are really extensive, so the description does not aim to be complete; we cover the main trends and results relevant to our work.

2.1 Preprocessing

When dealing with syntactic analysis, its notation and algorithms, it is not possible to work with plain text, we need additional annotation on lower levels.

Namely, the purpose of syntactic analysis is to reveal the structure of a sentence, and practically all of the parsing algorithms expect a single sentence on their input. So the first necessary step prior to any syntactic analysis is the sentence boundary detection.[1]

Next, basic unit for revealing the syntactic structure is usually a word, or a token (word, number or punctuation), sometimes a morpheme.[2] To be able to work with words, we need a tokenizer, a tool that splits plain text into words.

These two tasks may seem almost trivial (at least for languages that use spaces to delimit words) but even on this level there are severe problems that complicate further analysis. E.g. how should "A. Einstein" (no sentence delimiter) be distinguished from "... chat with person A." (and a sentence end)? (The previous sentence itself is a good example of sentence boundary detection complexity.) With regard to the tokenization problem, how should we tokenize "don't"? Should be e.g. "in spite of" 3 tokens or one token? Each of such decisions has non-trivial implications. And it is very

1. Stream syntactic processing is also possible but it introduces complications for most of the current approaches and it is not frequently used.
2. In further text, we will refer to these basic units simply as words, so "word" can mean also a number or punctuation. Differentiation between words and non-words is not necessary for this text.

important (and often problematic in practice) that all phases of the processing use the same assumptions, i.e. if the tokenizer splits "don't" to "do" and "n't", the morphology and syntactic analysis need to be able to process this tokenization correctly.

The next layer of processing is morphological analysis, i.e. assigning a base form (a lemma) to each word, and a morphological tag which codes the information about part of speech, gender, case etc. The morphological tag can be ambiguous or disambiguated – in the latter case, the task is referred to as tagging. The information about morphology is crucial for almost all of the state of the art parsers.

There are many available morphological analysers and taggers for English. We were mostly working with the TreeTagger [158], because of a positive previous experience. For Czech, the two notable taggers are MORČE [39] based on the positional tagset and Desamb [171] working over the attributive tagset of the Ajka tool [159].

There is a lot of other information potentially useful for syntactic analysis accuracy: namely named entity detection and non-word markup (phone numbers, time expressions, number-unit pairs) [122], multiword expression recognition [153] (these tasks partly overlap) or lexical semantic tagging of different types [173]. Such preprocessing can be used prior to syntactic analysis, but it is not really usual.

Natural language corpora can be exploited, too. A corpus is a large collection of texts in a particular language, in a format suitable for computer processing. Usually, it is indexed in a specialized database to enable fast searching and it contains morphological annotation. For parsing, corpora may be used for generation of various statistics, and if the corpus contains syntactic annotation, it is suitable for training statistical parsing models.

In syntactic representations discussed below, we suppose that morphological data is part of each word of the sentence, as well as its position in the sentence and other available information. In other words, each word is not just a plain string, but a complex node containing all the low level information about that word.

2.2 Encoding syntactic information

To reveal the structure of a sentence means to detect the way how words are organized – either we describe this organization as relationships among individual words, or we identify bigger and bigger units that the words organize into. The two most widespread approaches to notation of the syn-

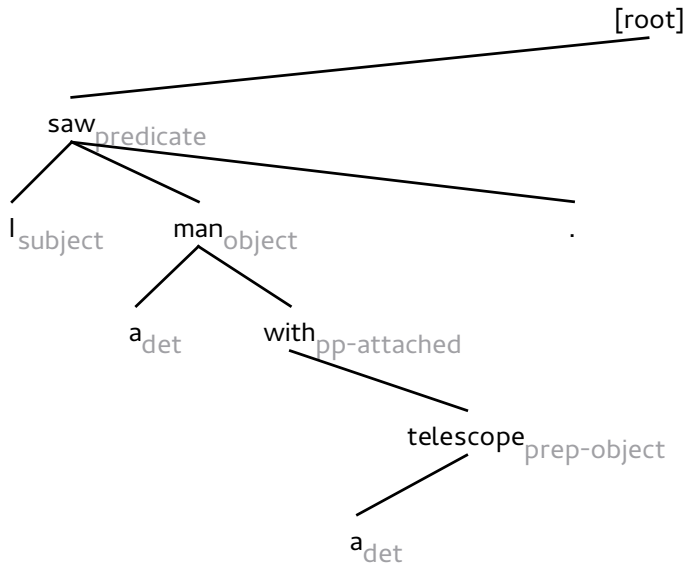


Figure 2.1: Dependency tree for sentence *I saw a man with a telescope.*

tactic information are derived from these two points of view – dependency formalism and phrase structure formalism.

2.2.1 Dependency formalism

Let $S = (w_1, \ldots, w_n)$ be a sequence of words that forms a sentence and L be a set of arbitrary strings. Dependency syntactic tree is defined as a tuple $(S, L, d, l, root)$ where

$$d : \{w_1, \ldots, w_n\} \rightarrow \{w_1, \ldots, w_n, root\}$$

is the dependency function[3] and

$$l : \{w_1, \ldots, w_n\} \rightarrow L$$

is the labelling function assigning a string from L, usually expressing the syntactic role of the word in the sentence, to each word (or equivalently, to each edge in the tree, i.e. each pair $(w_i, w_j) \in d$). Often, $(S, d, root)$ is referred to as unlabelled dependency tree, $(S, L, d, l, root)$ is called labelled

3. the notion of dependency tree can be generalized to dependency graph, then d is unrestricted binary relation

dependency tree. The root is a special technical node in the tree used to cover words that do not depend on any other word, so that the resulting graph is guaranteed to be connected.

The linguistic interpretation of the structure is that the dependent word, and the whole respective subtree, adds information to the governing word. In some languages, the governing word may determine morphological features of the dependent word, e.g. in case of gender, number or case agreement. The labelling function provides a classification of this relationship.

An example of a dependency tree is shown in Figure 2.1. Dependency trees are usually drawn as "ordered" which means that it is possible to recover the sentence word order by reading the tree nodes from left to right.

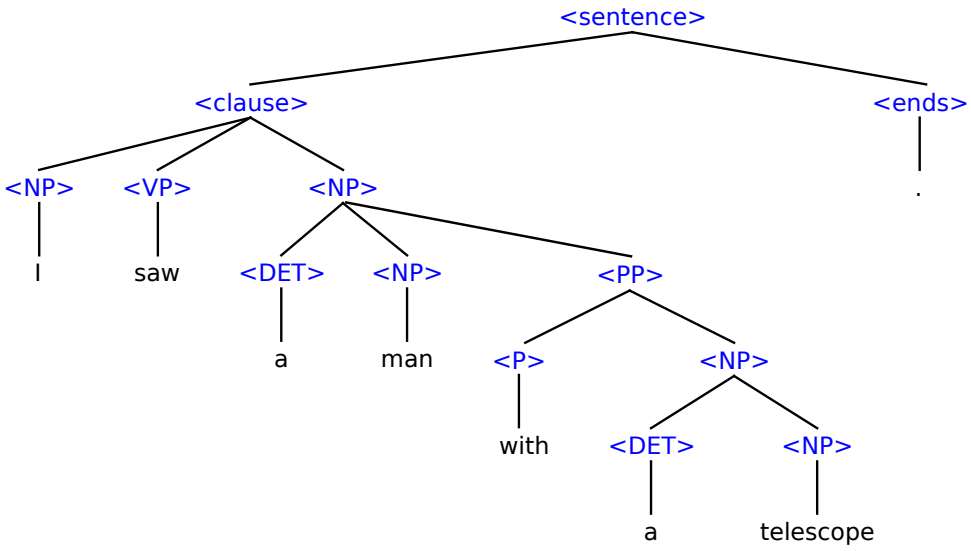


Figure 2.2: Phrase structure tree for the sentence *I saw a man with a telescope.*

2.2.2 Phrase structure formalism

Phrase structure formalism, or immediate constituent analysis, organizes sentence words into phrases, these phrases join to bigger phrases etc., and the whole sentence is marked as the very top phrase. It can be also viewed as a labelled bracketing of the sentence. A phrase structure tree for sentence $S = (w_1, \ldots, w_n)$ is defined as tuple (S, N, p, d) where N is a set of

(arbitrary) nonterminals,

$$p : w_1, \ldots, w_n \rightarrow N$$

is a *preterminal* function that assigns a nonterminal to each word (as each word is considered an atomic phrase), and

$$d : N \rightarrow N$$

is a function that assigns an immediate parent phrase to each phrase. We say that word w *belongs* to phrase phr if and only if

$$phr = p(w) \ \vee \ \exists n \in N : phr = d^n(p(w))$$

Here we assume that each nonterminal refers to exactly one phrase in the sentence. If more phrases need to be marked by the same nonterminal, our definition requires that these two nonterminals are differentiated, e.g. using integer indexes.

Usually it is required that every phrase (or nonterminal) covers a connected part of the sentence, i.e. if $w_i, w_j, i < j$, belong to a phrase phr, then $w_{i+1}, w_{i+2}, \ldots, w_{j-1}$ also belong to phr. This restriction enables the phrase structure analysis to be treated as labelled bracketing of the sentence, and allows straightforward use of formal grammar machinery proposed by Noam Chomsky [23], for the analysis.

An example of a phrase structure tree is given in Figure 2.2. If the connectedness assumption mentioned above holds, and the words in the sentence are written in their original order, the edges in the tree defined by the functions p and d do not cross.

2.2.3 Dependency vs. phrase-structure

The phrase structure formalism formed a base for work of Noam Chomsky [23], concept of formal grammars and the formal languages theory. Because of this fact, it is more commonly known in the world and often used as the first model when working with syntactic information. On the other hand, the dependency formalism (introduced by Tesnière [169]) has its tradition in Czech linguistics [163, 41]; the majority of syntactic research on Czech is exploiting it, and it is often claimed to suit better to the Czech language and its relatives, mainly due to its capability to record non-projective constructions [175].

In fact, both of the approaches have their strengths and weaknesses, even with regard to the Czech language. Both of them can record phrases, or constituents – in the phrase structure formalism, this information is explicitly given (by words covered by particular nonterminals), whereas in the dependency formalism constituents correspond to subtrees in the dependency tree.

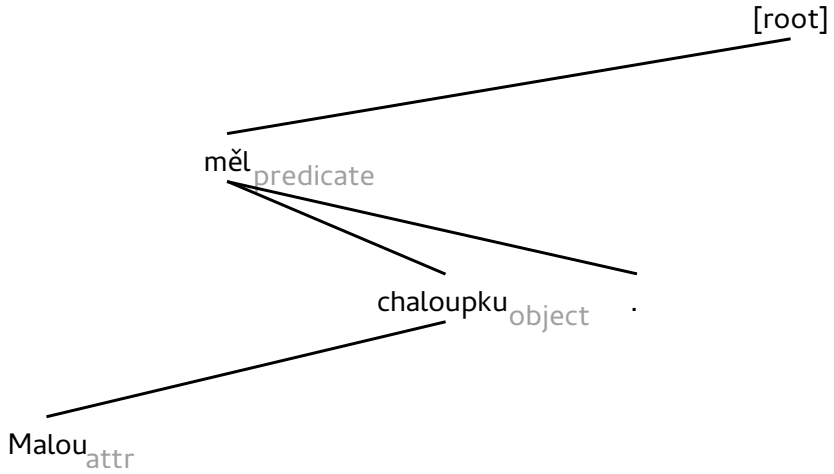


Figure 2.3: Illustration of non-projective dependency tree on Czech sentence *"Malou měl chaloupku."* (*"Little (he) had cottage."*). The phrase *"malou chaloupku"* (*"little cottage"*) is disconnected and the corresponding dependency edge is non-projective.

As we already mentioned, the dependency formalism can be used for straightforward annotation of non-projective constituents – where non-projective means disconnected, with regard to the original sentence word order. An example of non-projective dependency construction is given in Figure 2.3. If we want to record non-projective constructions using the phrase structure formalism, we need to define an additional relation over nonterminals that would join smaller connected parts of a particular phrase, or allow crossing edges, or change the sentence word order, as illustrated in Figures 2.4, 2.5 and 2.6.

The importance of non-projectivity for analysis of Czech is debatable. Zeman [175] reports that about 2% of all dependency edges in a Czech dependency corpus are non-projective, and about 20% of Czech sentences contain a non-projective construction, which seems to be significant. On the other hand, a closer look reveals that many of these non-projective construc-

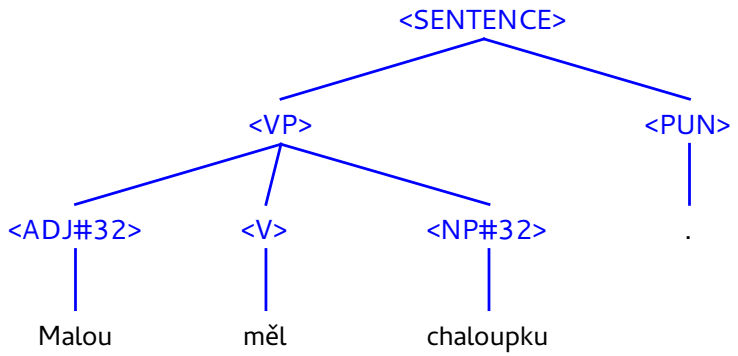


Figure 2.4: Marking non-projective constructions using the phrase structure notation with an additional relation over non-terminals. The same sentence as in previous example is used – *"Malou měl chaloupku."* (*"Little (he) had cottage."*)

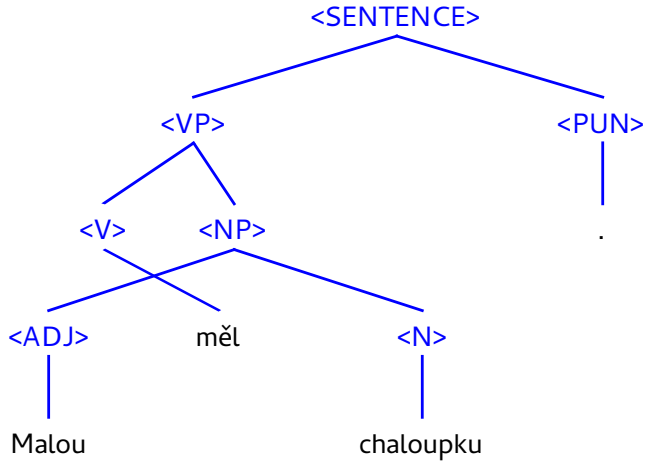


Figure 2.5: Marking non-projective constructions using the phrase structure notation with crossing edges. The same sentence as in previous examples is used – *"Malou měl chaloupku."* (*"Little (he) had cottage."*)

tions are rather technical and they would not be present if the annotation instructions were formulated in a slightly different way, with no impact on linguistic expressivity. A detailed discussion of this topic would be interesting but it is not directly related to the main focus of this work, so we will not go into more detail.

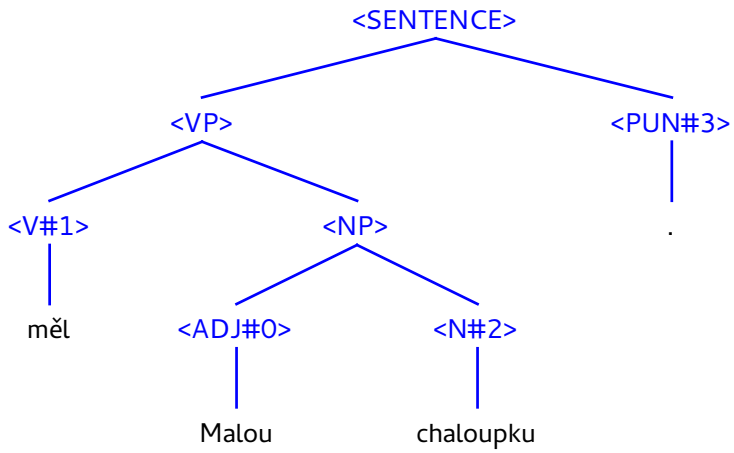


Figure 2.6: Marking non-projective constructions using the phrase structure notation with changing the word order. The same sentence as in previous example is used – *"Malou měl chaloupku."* (*"Little (he) had cottage."*)

Unlike the phrase structure approach, the dependency formalism is not able to record the difference between "new queen of beauty" and "new generation of fighters" straightforwardly (see Figures 2.7 and 2.8 for dependency analyses and desirable phrase structure analyses), with possible implications on the semantic level of analysis: new queen of beauty is definitely not a new queen, new generation of fighters is not really a generation of fighters.

The dependency formalism has also problems with phenomena that do not have binary nature, and cases where parts of a phrase are on the same level and do not have a clear governing word. Examples are coordinations like "small and ugly", multiwords like "instead of" or information such as phone numbers or addresses. These phenomena can be annotated by selecting one of the words (more or less arbitrarily) as the governing word and making the other parts depend on it, with a special label saying that this dependency is "different". However, it is not natural and it causes real problems in readability of the trees (see Figure 2.9) and in the parsing procedure – coordination is often reported to be hard for dependency parsing (e.g. [142]).

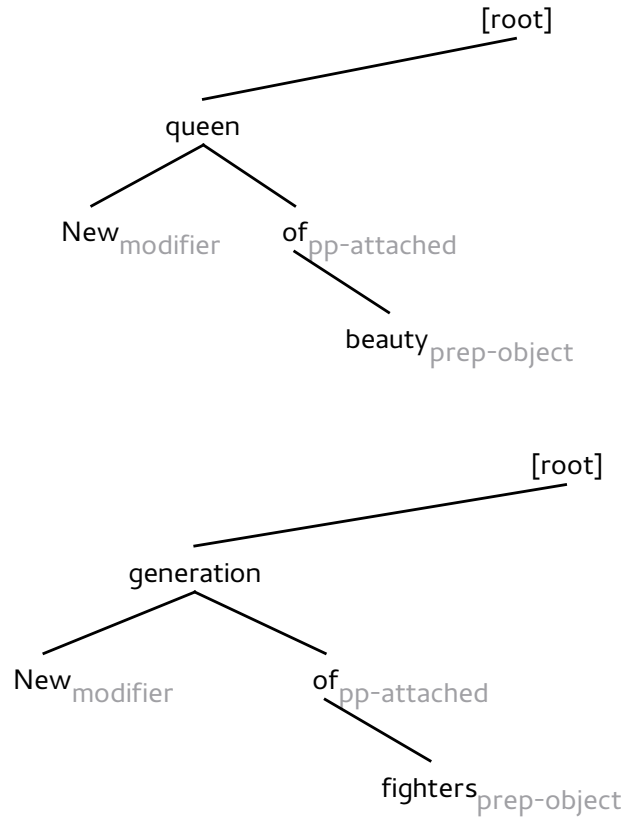


Figure 2.7: Dependency analyses of phrases *"new queen of beauty"* and *"new generation of fighters"*. The dependency analysis does not differentiate them.

2.2.4 Partial syntactic analysis

The formalisms introduced so far are sometimes referred to as "full" syntactic analysis. A concept opposite to the full analysis is partial, or shallow syntactic analysis [1]. Partial analysis does not aim to obtain a whole syntactic tree for a given sentence; rather than that, its goal is to annotate only selected syntactic phenomena, e.g. noun phrases, some (not all) types of dependencies among the words, or ambiguous syntactic relations.

Within the formal description, this can be achieved by reducing restrictions on functions d, l, p introduced above: they can be partial, they can be relations rather than functions… In practice, there are hundreds of ap-

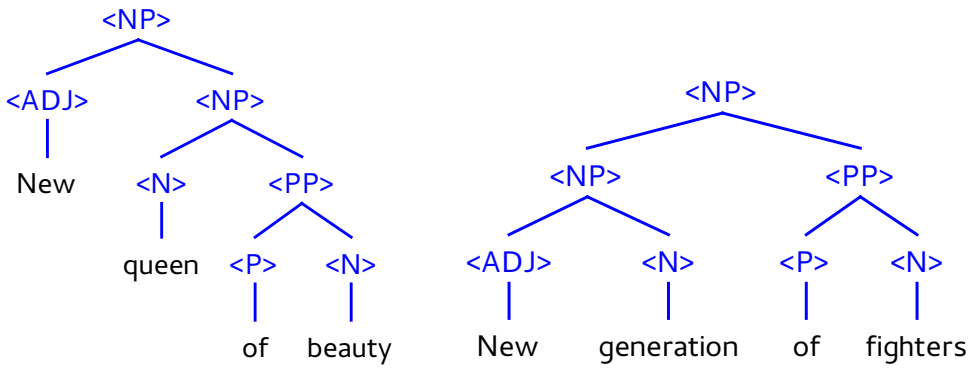


Figure 2.8: Phrase structure analyses of phrases *"new queen of beauty"* and *"new generation of fighters"*. The phrase structure analysis is able to capture the subtle difference between them.

proaches to partial syntactic analysis and the exact format is usually driven by the projected application.

Simple XML or even plain-text markup, rather than syntactic trees, are typically used to record the partial syntactic information which makes this approach very easy to understand and the annotated data much more readable than in case of full syntactic trees. As we will discuss later, this is a great advantage; although the expressivity of the approach is not as high as in case of full parsing, its transparency makes it much more usable in practical applications.

2.2.5 Advanced formalisms

Several advanced approaches were introduced that enrich the concepts of usual dependency and phrase structure syntactic annotation, aimed at even more precise description of linguistic aspects of syntax. Important representants of this group include Combinatory Categorial Grammars [166], Head-Driven Phrase Structure Grammars [140], Tree Adjoining Grammars [67] or Lexical Functional Syntax [10].

We will not discuss these formalisms in detail here, mainly because their added value is not immediately relevant to the topic of this work, and only a few tools exist that work within the scope of these formalisms and have significant coverage on common language, necessary for practical applications. Also, as we will discuss in following sections, we consider adding

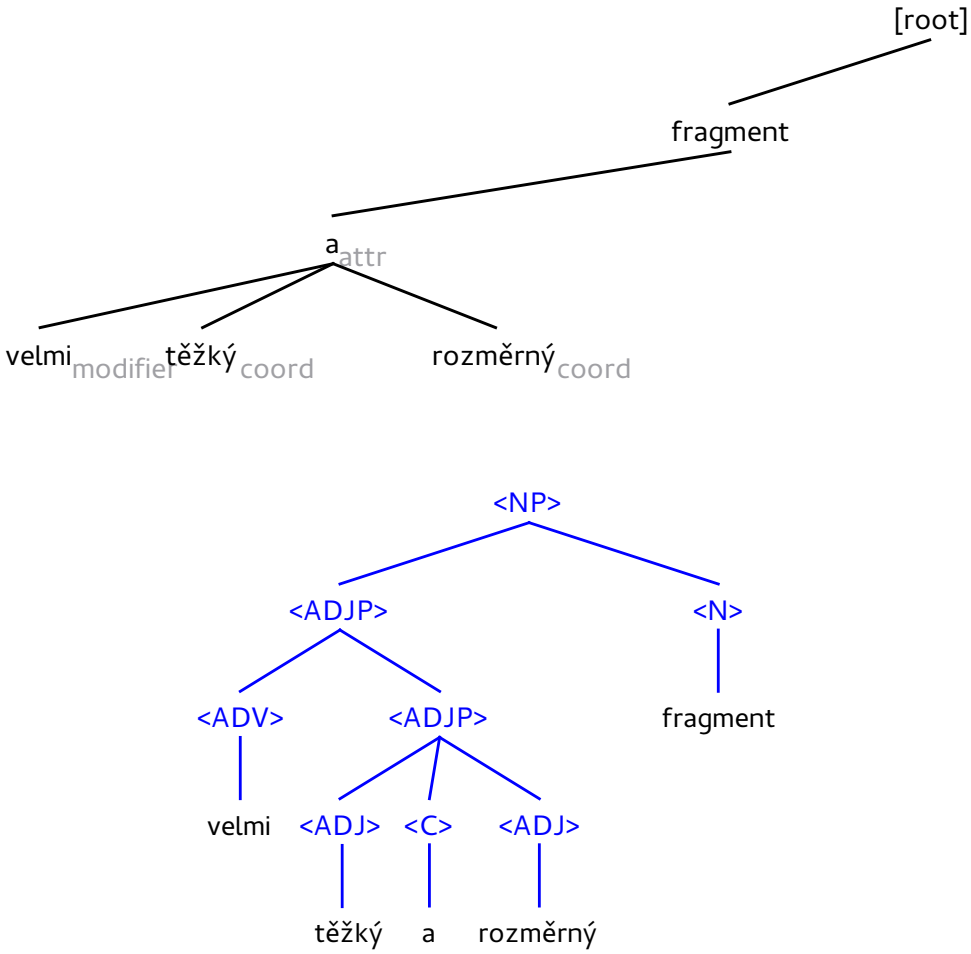


Figure 2.9: Dependency and phrase structure analysis of a phrase with coordination – *"velmi těžký a rozměrný fragment"* (*"very heavy and (very) bulky fragment"*). In this case, the phrase structure notation is much more readable.

more complexity a backward step on the way to broadly usable automatic syntactic analysis, at least from today's perspective.

2.3 State of the art parsing methods

The task for a natural language parser is to produce a linguistically correct syntactic tree, given the input sentence annotated on lower levels, namely for morphology. We can distinguish two basic approaches:

- **Rule-based** where the syntactic tree is constructed by a set of manually collected rules.
- **Statistical** where statistical information gained from a syntactically annotated language corpus, is used for building the syntactic tree. Such information can be in form of automatically extracted rules, a language model or weights assigned to various possible constructs.

Of course, combinations of these two basic approaches exist, namely the rule-based tools often use additional statistical information.

The main advantage of the statistical approach is definitely its language universality – provided there are annotated data, the same algorithm can be used for learning rules or models for any language. Also, the statistical algorithms are more precise when evaluating by tree similarity metrics, which is current standard in the field. Statistical algorithms can imitate the annotated data better than human experts.

The main advantage of the rule-based approach is its independence on annotated data. Building a manually annotated syntactic resource is a long-term expensive task and such a resource may not be available for some languages, or domains. Also, statistical parsers are always limited to the exact shape of the source data annotation – if such annotation is not suitable for a particular application, the statistical approach cannot be used straightforwardly.

Similarly to many other NLP sub-fields, the statistical approach is significantly leading in the last years' research on parsing, unlike the early stages of syntactic processing, in terms of accuracy against manually annotated corpora, as well as in overall number of statistical parsers compared to number of rule-based parsers. However, rule-based solutions are still used [98, 97, 15] and we are confident that rule-based approach should not be abandoned, mainly because of its flexibility and independence on the fixed shape of the annotated data.

There is a lot of natural language parsers in the world and it is not the aim of this work to explain their principles in detail, so we present only an outline here. The most important representatives of the statistical parsers include:

- Charniak's maximum-entropy-inspired parser [20], based on a probabilistic generative model
- Charniak & Johnson's parser [108] using lexicalized n-best probabilistic context-free grammar (PCFG [22]) and discriminative reranking
- Stanford University parser [75] which is able to provide dependency structure as well as phrase structure (expressed by PCFG) and each of them is scored with separate model
- Clark and Curran's CCG parser [24, 30] combining the Combinatory Categorial Grammar formalism with log-linear models
- Collins' statistical parser [27], based on a lexicalized PCFG model
- the MaltParser [128], a data-driven parser-generator for dependency parsing that supports several parsing algorithms and learning algorithms and allows user-defined feature models, consisting of arbitrary combinations of lexical features, part-of-speech features and dependency features
- McDonald's Maximum Spanning Tree dependency parser (further referred to as MST Parser) [109] that implements a discriminative learning method exploiting online large-margin training combined with spanning tree inference algorithms
- Nakagawa's multilingual dependency parser [123] which combines Gibbs sampling with Support Vector Machines

The best known parser with a rule base backbone is the RASP (Robust Accurate Statistical Parsing) system that combines rule-based grammar with a probabilistic parse selection model [12, 13].

For Czech, the notable statistical parsers are

- Holan's pushdown automata dependency parsers [48] that work on a base of premise-action rules learned from the annotated corpus
- Holan's ANALOG dependency parser [48, 47] that searches for the local tree configuration most similar to the training data
- Zeman's statistical dependency parser [176] based on dependency bigram modelling
- Collins' parser adapted to Czech [28]
- McDonald's MST parser adapted to Czech [110] and its improved version obtained by feature engineering on the model [130],
- Nakagawa's multilingual dependency parser [123] trained on Czech data for the CoNLL4 Shared Task 2007
- Nivre's MaltParser adapted for Czech [129, 127]

The rule based parsers for Czech include:

- Žabokrtský's rule based dependency parser [48] that implements a set of reduction rules in the form of Perl subroutines
- Dis/VaDis partial parser [172, 121] using manually created definite clause grammar in Prolog
- Synt phrase-structure parser [51, 83, 63] that combines manually developed context-free meta-grammar with contextual actions, an efficient variant of the chart parsing algorithm and a tree-ranking function partly learned from a corpus of sentences annotated on constituent structure [84]
- SET parser [90] based on a set of pattern-matching linking rules that is able to produce both phrase-structure and dependency trees; this parser is described in detail in Chapter 5, as one of the important contributions of this work

2.4 Treebanks and parsing evaluation

For the purpose of training the statistical parsers and measuring accuracy of parsing, manually annotated corpora are built, usually containing syntactic trees for natural language sentences; therefore they are called treebanks.

The best known and most used treebank for English is the Penn Treebank [103] based on the phrase-structure formalism. It contains about 3 millions of syntactically annotated words and it plays the role of a standard for training and testing of English parsers. For Czech, the biggest and most used such resource is the Prague Dependency Treebank (PDT) [38] which uses the dependency formalism and contains about 1.5 million tokens annotated on the syntactic layer (called "analytical" within the scope of the functional generative description theory [160] that is the annotation based on). Treebanks of similar sizes are available for most of world's major languages, using either dependency or phrase-structure notation of syntactic information.

Treebanks are crucial components in the state of the art parsing evaluation methodology. The evaluation usually takes the raw (morphologically annotated) sentences from a particular treebank, uses them as input for a parser and compares output of the parser with manual syntactic annotation within the treebank. In case of statistical parsers, different parts of the treebank are used for training and evaluation – either the treebank has a dedicated evaluation part, or n-fold cross-validation is used.

To compare trees on the output of a parser with the so called "gold standard" trees present in the treebank, various tree similarity metrics are used. In case of dependency formalism, the standard precision and recall over the dependency edges is measured. Precision is defined as number of correctly identified edges divided by the number of all edges identified by the parser, and recall is number of correctly identified edges divided by the number of all edges in the gold standard data. Because we are comparing two trees and the number of edges is always the same in the parser output and in the treebank data, precision is always the same as recall:

$$P = R = \frac{\#correct\_edges}{\#words\_in\_sentence}$$

In further text, we will refer to this metric simply as dependency precision (in most papers it is called accuracy but we consider this term too ambiguous for our purposes). Labelled and unlabelled dependency precision can be distinguished, according to what edges count as correct: in case of unlabelled precision, only the governor of each word is taken into account, the labelled precision counts with pairs (governing word, dependency label).

In case of phrase-structure trees, there is the PARSEVAL measure [36] which is used at most. It is comprised of precision and recall over all the identified phrases, rather than edges (in this case, precision and recall are usually not the same). Again, labelled and unlabelled version of both measures can be distinguished where the the nonterminal covering the phrase is considered as label. A more complicated metric called leaf-ancestor assessment was proposed [154, 155] which is reported to provide results that better correlate with human judgement of the parse quality.

Best results against the testing part of the Penn Treebank were achieved by self-trained Charniak & Johnson's parser [108] – slightly over 92 percent in terms of the PARSEVAL numbers.

Best parser of Czech, according to evaluation against the testing part of the Prague Dependency Treebank, seems to be the Nakagawa's multilingual parser [123] trained on PDT for the CoNLL4 Shared Task 2007, with 86.3 percent unlabelled and 80.2 percent labelled dependency precision. MaltParser [127] and MST parser [110] also reported high unlabelled precision – 85.8 and 84.7 percent respectively. In case of CoNLL4 task, only a part of the PDT data was used, so it is not clear how the numbers relate to each other precisely. A more detailed and up-to-date information regard-

ing evaluations against the PDT can be found on the web of the Institute of Formal and Applied Linguistics.[4]

2.5 Criticism of current methods

As we have illustrated, the spectrum of parsers is very rich, even for a language with less speakers like Czech there are tens of them. Their precision is relatively high – 85 to 90 percent means about 2 errors per sentence but this includes attachment of all words including e.g. punctuation or particles that most of possible applications would not use at all.

Also, it is a question where a theoretical maximum lies, i.e. the percentage of agreement of human annotators familiar with the task. For morphological tagging of English, 97 percent is reported, although this number probably comes from a semi-official note [101], and such precision has already been achieved. For parsing, the inter-annotator agreement (percentage of edges/phrases where human annotators agreed with each other) is almost never reported in case of the prominent treebanks, and it may well be around 90 percent. One of the few resources where the agreement *is* reported is [113] and although it is a slightly different task, the figures for the most important parts of the annotation ("structure" and "functor", i.e. the governing word and the syntactic label) indicate that it is even *below* 90 percent. Sampson and Babarczy [156] provided strong evidence that the discussed agreement cannot go beyond 95 percent, in terms of the leaf ancestor assessment metric, and more precise annotation instructions do not help to achieve higher figures of agreement. This further supports our hypothesis of about 10 percent people disagreement on the syntactic annotation tasks.

Quite a few questions arise here. Clearly, the field is very well developed; why does it have so little impact on real-life applications? Why major commercial IT players like Google or Facebook (or Seznam, kind of Czech Google) that work with a lot of natural language data and exploit other NLP tools such as morphology taggers heavily, do not use syntactic tools on a daily basis? Are the results from parsing useless? If the field is close to the theoretical maximum that parsing can achieve, is the methodology correct? Is the task well defined, if the upper bound of inter-annotator agreement lies below 90 percent? And why is the inter-annotator agreement not consistently published? Do we know what we actually need from natural language parsing, for practical applications, or does the field exist just for

4. `ufal.mff.cuni.cz/czech-parsing`

itself, happy with increasing percentages of accuracy against 20 years old treebanks?

2.5.1 Low usage

Let us bring more evidence for the lack of parsing usage. We have already mentioned that Google and other big players rely rather on word-level statistical methods of analysis.

Even in other advanced NLP tasks such as question answering, and even within the NLP community, the tendency to use statistical word-level methods rather than parsing, is clearly visible. A recent attempt for a question answering system for Czech does not even mention parsing technologies [78]. As we have already mentioned, Wikipedia page for information extraction does not mention parsing or syntax at all.

As Jakubíček noticed, the word "parser" is mentioned in 7,232 of 21,066 papers available within the ACL Anthology (2012), but looking for phrases matching the regular expression

$$(used|using|employ|employing|exploit|exploiting)\ a?\ parser$$

gets only 133 results [60, pp. 12–13]. The important and awarded papers on parsing technologies deal usually with improving the accuracy figures on available annotated data rather than using the parsers in real-world applications.

All these facts indicate that there is something wrong with the research on parsing and that the field needs a restart, or a change of direction, for the results to be usable in practical applications. In the rest of this chapter, we analyse the particular problems and outline our methodology aimed at changing this bad state. In the next chapters, we introduce our contribution within this changed direction.

2.5.2 Application-sparse output

The first and crucial problem is that the information contained in syntactic trees is too complex and often tangled. The trees are built according to a complex linguistic theory, and the annotation follows complicated and extensive guidelines, built on top of the theory. The annotation manual for the syntactic layer of the Prague Dependency Treebank has more than 300 pages [42], as well as the Penn Treebank annotation guidelines [9].

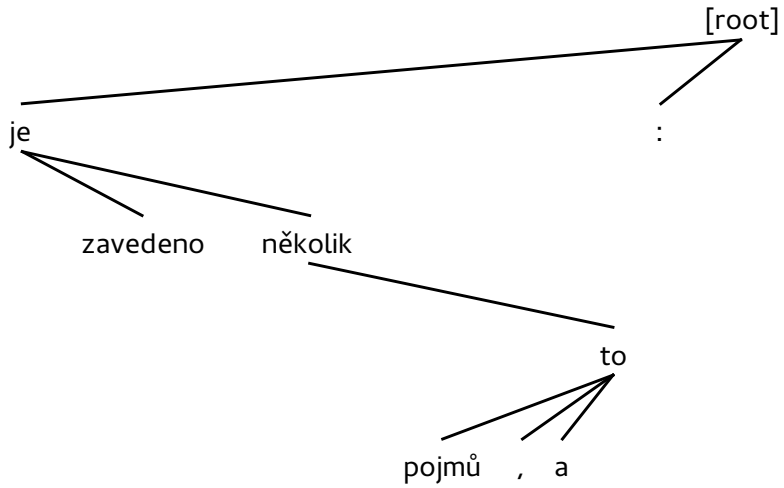


Figure 2.10: A PDT tree for a phrase *"je zavedeno několik pojmů, a to:"* (*"are introduced several concepts, as follows:"*). Because of trying to capture the subtle relationship between the word *"pojmů"* (*"concepts"*) and the next sentence, expressed by the *"as follows"* hook, a clear constituent *"několik pojmů"* (*"several concepts"*) does not form a subtree, so it will not be recognized as a phrase.

For people not educated in linguistic theory, it is thus almost impossible to get familiar with the format of the trees and exploit the information within. Even for people dealing with natural language processing on a daily basis, it is often very hard to decode some constructions – see also discussion in Section 3.1.5. We have already mentioned the fact that there is lack of NLP papers on using parsers, compared to papers on parser development. However, even among papers that report using the information from parsing, the majority exploits the syntactic information simply as additional input to some machine learning, a black box that can make use of any additional information [118, 49]. Especially, there is usually no need to understand the meaning of the inputs for the machine learning, so this is a strong evidence that even NLP researchers rather do not bother with understanding parser outputs more in detail.

Another related problem is, the trees contain far bigger amount of information than a particular application can possibly exploit. Attachment of punctuation nodes and particles, relationships among various parts of analytical verb forms, complex relationships of rather big units such as clauses

or appositions... Any current practical application can hardly get any benefit from these. However, the fact itself that this information is present, is not only disturbing for readers and users but adds real noise to the data. See Figure 2.10 for an example of how advanced annotation can skew e.g. noun phrase information. The complexity of the annotation is trying to address advanced issues but this fact complicates straightforward use of the structures.

Quite often, the formalism itself forces adding an unintuitive link where it has no clear interpretation, e.g. in case of ellipsis, or coordinations in the dependency formalism – words that should depend on all members of a coordination are usually marked as dependencies of the conjunction (see Figure 2.9 for an example) [142].

On the other hand, some important information is clearly missing in the syntactic trees. It is not easy to identify what exactly is missing but there are some hints. From the Prague Dependency Treebank, it is impossible to extract boundaries of clauses in a reliable way, which is documented by the fact that a next round of annotation is needed for adding this information [100]. Miyao et al. [118] report for parsers of English:

> *The results show that the task accuracy improves when using a double parser/representation ensemble. Interestingly, the accuracy improvements are observed even for ensembles of different representations from the same parser. This indicates that a single parse representation is insufficient for expressing the true potential of a parser.*

It seems that the common tree representation may not be enough for purposes of real world applications and that the information contained within the usual syntactic trees and the information needed by various applications somehow miss each other. Parsing should offer results directly exploitable in applications, according to the needs of these applications, rather than trees based on theory, containing a lot of noise (from the application point of view) and missing important information.

2.5.3 Application-free evaluation methodology

The second problem is methodology of parser evaluation – the tree similarity metrics against a treebank "gold standard" – which probably causes that the supposedly best parsers are not the best for the applications. The problem consists of multiple aspects.

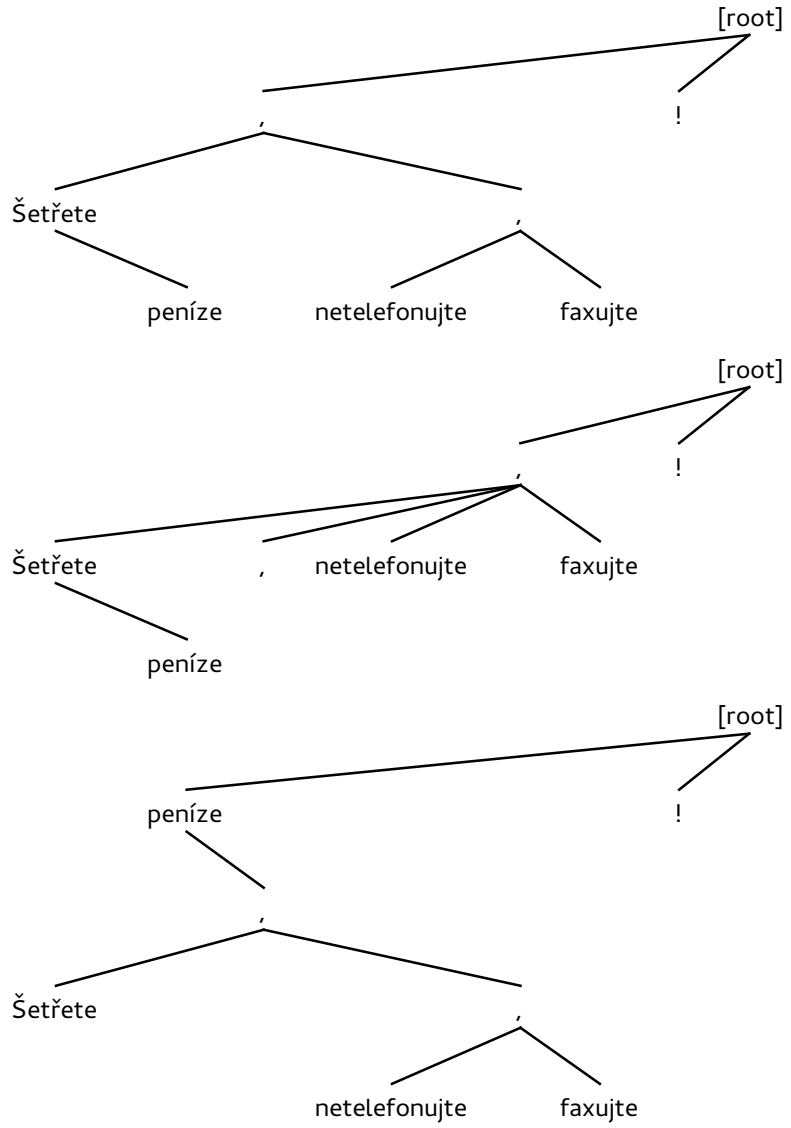


Figure 2.11: Three possible analyses of sentence *"Šetřete peníze, netelefonujte, faxujte!"* (*"Save money, do not phone, fax!"*). PDT gold standard is first. The second analysis does not reveal the (debatable) coordination nesting but captures all the important relationships. The third analysis is really wrong (does not record even the relationship between *"save"* and *"money"*), however, in terms of dependency precision it is better than the second one, with respect to the gold standard.

The similarity metrics do not take into account that attachment of different words has different importance, e.g. attachment of a particle would be probably less serious error than attachment of a subject. The figures do not tell us if the parser makes rather trivial errors, or if the errors are serious – for illustration, see Figure 2.11. To fix this, we would need to introduce complex weightings which is probably not achievable without the metric becoming complicated in an unacceptable way, and without endless arguments about the weighting among parser developers. In addition, these metrics would probably need to be different for different applications, and language dependent.

This problem, in combination with inappropriateness of the tree format for the applications, can cause that the statistical tools may be misjudged as winners of the best parser competition. The statistical algorithms are very powerful in mimicking the training data, and they optimize for the best score within the tree similarity metrics. As we have illustrated, both the training data in form of trees and the tree similarity metrics are not what we want, as they do not tell us anything about usage in any particular application. So, the state of the art evaluation is based on a task that has no real usage – mimicking the trees in the training data – according to not very expressive optimisation function, and this approach is supposed to be universal. This naturally helps statistical algorithms (that are, however, great in case they are used with data that represent a practically oriented task) to gain bigger score, even if it may be not significant for particular applications (or even linguistically significant).

Also, when moving around the possible 90 percent agreement among human annotators, how much of the data are actually random, with random, rather than linguistically relevant agreements, some of which statistical algorithms would spot? Unfortunately, it is impossible to answer this question, as the appropriate numbers were not published (and we doubt they will ever be). But on pseudo-random data, the statistical algorithms would be also expected to win, simply by principle. We are convinced that the victory of the statistical tools in parsing is not that definite as it seems to be.

Last and probably most important, trees in treebanks describe the sentence structure from the language theory point of view, which is different from the application point of view. These are two very different things, and it has been empirically proven that they not correlate well. Already mentioned paper by Miyao et al. [118] about extraction of protein-to-protein interactions (PPI) concludes that there is a "strong correlation between parse accuracy and PPI accuracy", however, after a closer look, the correlation

is only about 0.75 which cannot be really marked as strong. More importantly, regardless the correlation, the paper states that "a 1% absolute improvement in parser accuracy corresponds roughly to a 0.25% improvement in PPI extraction accuracy". If parsing accuracy moves around 85 percent (with theoretical maximum possibly near 90) and the PPI accuracy around 57, it does not give much hope that a better parser, in terms of accuracy against a treebank, can help with achieving usable results in the PPI task.

Our research on collocation extraction, further described in Section 6.5, showed no correlation between accuracy against a treebank and accuracy of collocation extraction.

Another paper dealing with another task [68] reports even negative correlation: the higher the accuracy of the application, the lower the accuracy against a treebank. This can be considered an empirical proof that the tree similarity metrics against gold standard are useless, and rather hostile, for application-oriented parsing. The state of the art evaluation methodology has led the parsing research along a wrong way.

2.5.4 Technical aspects

Last problem complicating the usage of parsers in real world applications does not relate to methodology or theory, but to the very mundane technical aspect of their usage: They are hard to be put into operation and ran. This is related to the fact that they are created primarily as research applications and the majority of their usage is to run and evaluate them against treebank data.

In a bachelor student project where part of the task consisted in re-training and replicating the published results of MaltParser and MST parser on the Prague Dependency Treebank, the student (of computer science) spent about three full person-weeks with putting into operation, re-training and optimization of the parsers. In case of MaltParser, she has not been able to replicate the published results (see the bachelor thesis [77] in Slovak).

Another lovely illustration of technical difficulties in using a parsing tool is a desperate email we got as the Sketch Engine support members, asking if we could run an Arabic parser and tagger within the Sketch Engine service; citation of part of that e-mail follows:

I downloaded AMIRA into my computer and as I have no knowledge of perl, I could not figure out how to run it. I asked many linguists in my lab and no one knew how to run it and after numerous emails back and

forth to the computer science department, no one is available to do it and the only computational linguist I knew was away till after Christmas.

We believe that the inability of the whole university department (!) to run the tool does not relate to being or not being computational linguist but to the fact that the tools are complicated and it is hard to run them. We have had lots of more such experience in context of Lexical Computing work, as the Sketch Engine uses many external tagging and shallow parsing tools. Even that we are very skilled in running this type of tools, it often takes person-days to person-weeks to run them. There is also problem with robustness of the tools: They often fail on long or imperfect sentences, consume all the memory and make the machine fail and so on.

The developers of parsing applications should think more about their users and make more accessible applications.

2.5.5 You aren't gonna need it

To sum up the content of the last pages:

- The output of natural language parsers does not contain directly the information that the applications need.
- The parsing evaluation methodology leads to bad development direction. Instead of optimizing output for applications, it is optimized for treebank annotation.
- The parsers are technically hard to use.

In the following paragraphs, we introduce another methodology based on principles used in the field of software development, that will help us to deal with these problems.

A few principles that share a common core exist in agile and rapid software development [104], not grown up from research work but from everyday routine in software development. Namely,

- **"Worse is better"**[5] extensively discussed by Richard P. Gabriel [33, 34]
- **"Keep it simple stupid"** (KISS) principle[6] emphasizing the importance of the simplicity of system design (in general – originally, the phrase has been associated with aircraft engineer Kelly Johnson and U.S. Navy)

5. `en.wikipedia.org/wiki/Worse_is_better`
6. `en.wikipedia.org/wiki/KISS_principle`

- **"You aren't gonna need it"** (YAGNI) principle[7] which discourages from implementing functionality which is not immediately needed

All these three principles, or approaches to system design, share several values, mainly prioritizing simplicity in design over all other values – completeness, correctness and consistency. Usually, the software design follows priority order that Gabriel calls "The Right Thing":

- **Correctness:** incorrect solutions are simply not allowed.
- **Consistency:** the design must be consistent. Consistency is as important as correctness.
- **Completeness:** all reasonably expected cases must be covered. Simplicity is not allowed to overly reduce completeness.
- **Simplicity:** If the design fulfills the above three criteria, it should be also simple. It is more important for the interface to be simple than the implementation.

In spite of that, the "Worse is Better" approach follows a different order:

- **Simplicity:** is the most important consideration in design.
- **Correctness:** the design must be correct. But it is slightly better to be simple than correct.
- **Consistency:** The design must not be overly inconsistent. However, consistency can be sacrificed for simplicity in some cases.
- **Completeness:** the design must cover as many important situations as is practical. But this is the bottom priority; completeness can be sacrificed in favor of any other quality.

The YAGNI principle adds an important note to the development process, rather than project design, that new features should be implemented at the time when they are really needed, not when it is foreseen that they will be needed. This is given by the (usually underestimated) aspect of the development work, that the reality often differs from initial projections dramatically. Not following this principle can lead to incredible amounts of wasted work.

Examples of well working systems based on this approach are Unix or the C programming language. There are also roots and overlaps within philosophy, namely the Occam's razor stating that from competing hypotheses (or approaches), the simplest should be selected. There are also many quotations supporting the basic idea, by the biggest personalities in history and

7. `en.wikipedia.org/wiki/You_aren%27t_gonna_need_it`

across disciplines, such as Leonardo da Vinci's "Simplicity is the ultimate sophistication", Mies Van Der Rohe's "Less is more", or Antoine de Saint Exupéry's "It seems that perfection is reached not when there is nothing left to add, but when there is nothing left to take away".

How does this philosophy apply to the automatic syntactic analysis research? The field desperately needs to abandon the approach which starts with a language theory and hopes to reach practical applications subsequently. The language theories are currently in position of initial projections in a software development project, according to the YAGNI principle; strictly following them and evaluating results against them, quite an amount of work has been wasted which may be never used practically. We have already presented empirical observations which indicate that this approach is not leading to satisfactory results. Also, it should be noted that syntactic theories are not good empirical theories in the sense of Karl Popper's criteria [143] – namely, there are no clear conditions that would falsify such a theory – so, in our opinion, they do not really pose a good theoretical basis, a fixed point for further practically oriented research, as it might seem on the first sight. They are just ideas of how it could work in the human brain, with cannot be really falsified.

Rather than starting with a syntactic theory, usually represented by treebanks and research groups around them, we propose starting with particular applications, as we outlined them in the introduction, and viewing them as software development tasks. Namely, according to the principles introduced above, we should

- identify the immediate needs of the applications and work on these, not trying to predict future needs (YAGNI)
- build simple, possibly task-specific parsers in terms of both design and user interface, rather than try to create parsers that cover all nuances of natural language and produce all-encompassing trees (preferring simplicity to completeness)
- adapt the methodology of the parser evaluation to the application's need, not the language theory represented by the treebanks, to be able to track real progress

Besides other implications, this means:

- Tree representations should be adjusted to better reflect the needs of the particular applications. This contrasts with the current most frequent pattern of using parsers, where the output is usually taken "as it is".

- Research should rather aim at rule-based parsing methods first, to find out what type of information is actually needed by particular applications (and only after finding it, eventually start annotation of data and employ statistical algorithms, to improve the analysis – statistic algorithms are good only if we know exactly how the result should look like).
- Research should strongly prefer extrinsic evaluation of the parsing tools (i.e. evaluating the applications that exploit the syntactic information), to evaluating accuracy against a treebank.

Although widely accepted (though often not followed) within the software development community, the "worse is better" approach is not very popular in the research area. We regularly meet reactions like "preferring simplicity to X is a great misunderstanding" or "with your approach, you cannot do X". Mostly they are right from their theoretical point of view, but these voices do not understand the core message of the approach. It is unavoidable that there are problematic aspects in a system built according to the "worse is better" approach and rapid application development. However, the enforced design simplicity significantly helps usability, as can be seen in many other examples from information technology (hypertext, JSON,[8] Python, ...). And usability is the key feature that we expect from software tools, including NLP tools and especially parsers.

As we illustrate in the next sections, this approach works, and tools based on its ideas are used in many follow-up applications.

8. `www.json.org`

Chapter 3

Bushbank

This chapter starts with more criticism of treebanks, as the most common representatives of syntactically annotated language resources. Then, we introduce a new format of syntactic annotation, bushbank (in contrast to treebank) that overcomes most of the described problems and is coherent with the "worse is better" principle described in the previous chapter. We also mention pioneer corpora annotated in this format. Content of this chapter describes a joint work with Marek Grác with author's contribution of ca. 50 percent. The bushbank idea was firstly outlined in [35], but it was significantly developed since then.

3.1 More criticism of treebanks

Apart from the above mentioned methodological problems related to parsing evaluation, treebanks as main representatives of syntactically annotated corpora suffer from other, clearly visible disadvantages.

3.1.1 Price

Treebank annotation always has to agree with some, usually complex, language theory and the annotation process needs human annotators deeply educated in this theory. Besides, the annotators need to be able to decide borderline cases (usually, there are many) and real corpus instances not covered by the core theory, consistently. This results in annotation manuals with hundreds of pages [9, 42] and the annotators are asked to follow these instructions precisely (which, in case of several hundreds of pages, may not be even reachable). This makes treebank annotators rather outstanding (not everyone is able and willing to finish linguistic education) and very expensive yet before they start the annotation work.

What is also expensive is the annotation process itself: Building a correct syntactic tree from words in a sentence is an abstract task and the need for

applying extensive manuals and complex language theory causes it takes 3–10 minutes per sentence [113].

We should strive for cheaper methods of syntactic annotation which would allow creating new resources.

3.1.2 Age and domain specificity

Since annotation is expensive, treebanks are very rare; creation of new ones is too costly and therefore only the old existing ones are still used. The texts in the Penn Treebank are already more than 20 years old and the character of English has changed massively in the meantime. 20 years ago, there was no Google (and no verb "to google"), no Facebook, most of the published texts were carefully edited... It is probably correct assumption that a parser trained on Penn Treebank will have problems when analysing up-to-date Facebook posts.

There is a similar problem with domain specificity – texts in treebanks mainly consist of journal papers which is a common domain, however, it has its specific features that are not general. Assumptions induced by training and testing a parser on such a data set may become invalid when the parser is used for analysis of other text types, e.g. specialized texts from domains like chemistry, medical records, the above mentioned Facebook or Twitter posts or SMS messages [19].

These problems relate to the price of syntactically annotated resources – if we found a faster and cheaper annotation schema, we would be able to produce annotated resources for various domains.

3.1.3 One tree per sentence

Syntax is a language level that comes with massive ambiguity. Sometimes, this ambiguity can be resolved by wider context, e.g. in case of the famous sentence *"I saw a man with a telescope"*. However, sometimes the ambiguity resolution is not only hard, even with context and for human beings, but even not important at all in the process of understanding the sentence. In sentences like *"A plane crashed into the field behind the forest"*, it is not at all important if it was the field behind the forest, or if the crashing took place behind the forest. In any format of a syntactic tree, however, this needs to be resolved and no ambiguity is allowed, although both of the syntactic interpretations are valid. In other cases, the difference between the two interpretations may be important but intentional and not resolvable by the context – but only one of them must be selected for the annotation.

As Sampson reports [156]:

... not only is genuine structural ambiguity quite common in English, but (more surprisingly) there is often no reason to resolve ambiguities because in practice either interpretation amounts to the same thing.

This goes strongly against the "one tree per sentence" principle. We should aim at recording genuine ambiguity in syntactically annotated resources.

3.1.4 Senseless annotations

Similar problem, enforced by the formalism, is annotation of each node in the sentence, including rather technical non-words like punctuation, bullets and numbering of ordered lists, structure of addresses, phone numbers or tables. These phenomena should not be marked for natural language syntactic structure, as the connection to natural language is very weak and the syntactic tree does not bring any new information. However, the formalism enforces attachment and recording the inner structure of such nodes which leads to entirely artificial annotations with practically no linguistic interpretation [42, pp. 259–262, 267–270, 274–275], that are defined only by rules in the annotation manual rather than by language understanding. Such annotations are confusing when using the annotated data and misleading by training and evaluation of natural language parsers. Sometimes, the evaluations do not take attachment of punctuation into account [14] which overcomes the most visible part of the problem, however, this is not sufficient.

For example, in the Prague Dependency Treebank, we can find annotations as in Figures 3.1 and 3.2 (both of them come from the beginning of the `train-1` part of the treebank). In our opinion, such constructions should not be annotated for natural language syntactic structure at all.

3.1.5 Inconsistent and counterintuitive annotations

Very large number of borderline cases in the annotation theory and too extensive manuals that, nevertheless, fail to cover all cases, lead to errors and inconsistencies in the annotations. Because the process of annotation is expensive, it is not easy to process each sentence multiple times, which would minimize random errors.

However, the random errors are not the only problem: In many cases, the rules from the annotation manual themselves lead to problematic an-

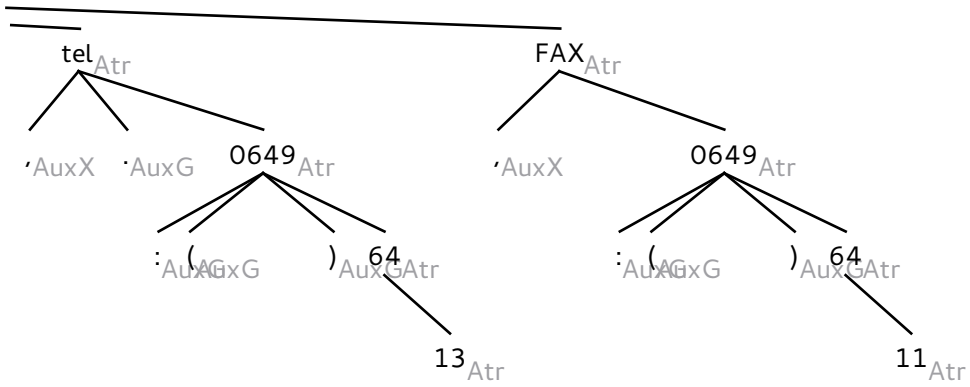


Figure 3.1: PDT annotation of text fragment *", tel.: (0649) 64 13, FAX: (0649) 64 11"*.

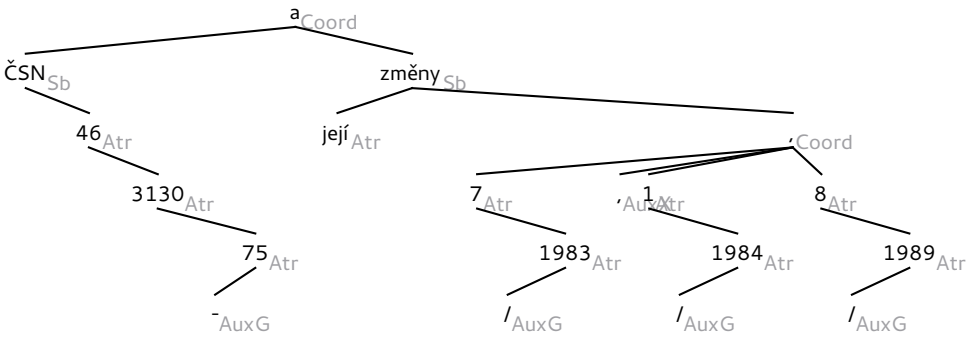


Figure 3.2: PDT annotation of text fragment *"ČSN 46 3130-75 a její změny 7/1983, 1/1984, 8/1989"* (*"ČSN 46 3130-75 and its modifications 7/1983, 1/1984, 8/1989"*)

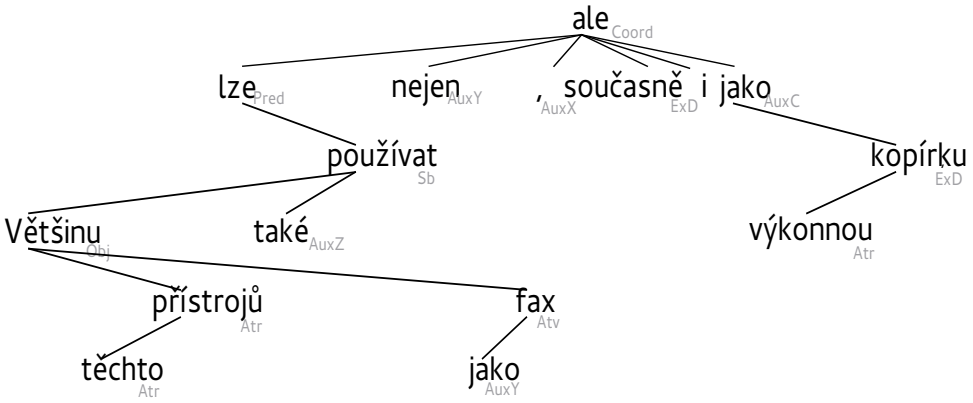


Figure 3.3: PDT annotation of sentence fragment *"Většinu těchto přístrojů lze také používat nejen jako fax, ale současně i jako výkonnou kopírku"* (*"Most of these devices can be used not only as fax but in the same time also as an effective copier"*). Because of complex annotation instructions, the relationship *"používat ← jako fax"* (*"used ← as fax"*) cannot be recognized in the tree, as well as the relationship *"jako fax ↔ jako kopírku"* (*"as fax ↔ as copier"*).

notations. In that case, annotations are formally correct (they do fulfill the annotation manual) but for people who are not precisely informed about the annotation rules, they seem just random and do not make sense at all.

We have analysed a part of Prague Dependency Treebank for such deficiencies and reported the results in [85]. In the paper, we have presented all the problems as errors in the annotation; later we found out that many of them are actually correct annotations according to the manual and the background language theory – they are just absolutely counterintuitive and not usable in any possible language technology application. The same would hold for outputs of statistical parsers trained on such data.

Examples of such annotations are given in Figures 3.3 and 3.4. In the annotation process, such problems should be avoided as much as possible, and the annotation result should be intuitive and transparent.

To sum up: Building of treebanks is expensive; therefore there are at most few of them for each language, and building new ones (e.g. specialized or domain-specific) is almost an impossible task, due to limited financial as well as human resources. Annotations contained in treebanks are

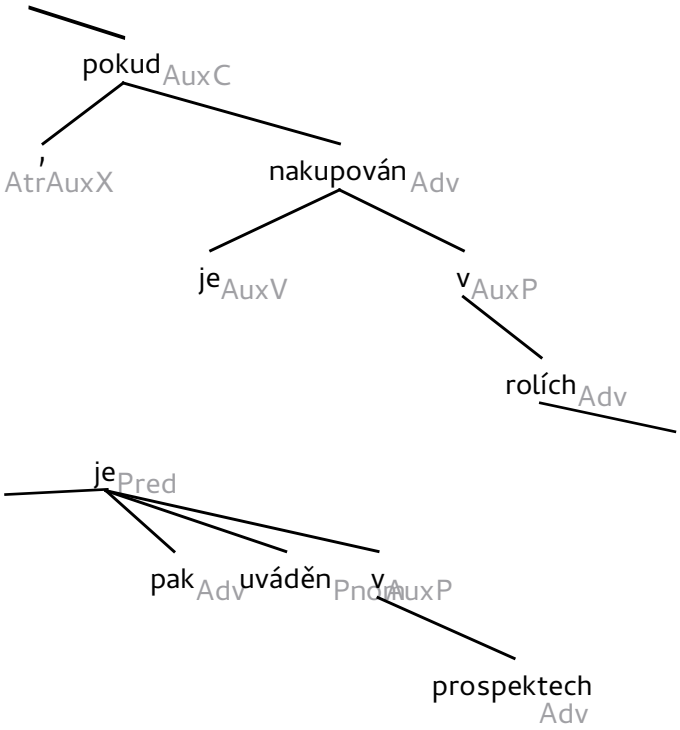


Figure 3.4: Frequent inconsistency among annotation of passive participles in PDT, illustrated on fragments *", pokud je nakupován v rolích"* (*"if (it) is bought in reels"*) and *"je pak uváděn v prospektech"* (*"is then reported in brochures"*). In the first case, the passive participle *"nakupován"* is the governing word, in the second case the governor is the auxiliary verb *"je"* (*"is"*). Sometimes this difference is correct according to the annotation manual, based on subtle semantic features of the particular verbs. However, such distinction is non-intuitive and changes the overall shape of the tree.

problematic as they do not allow ambiguity, and extensive manuals often skew the annotation into a non-readable and counterintuitive noise.

3.2 Syntactic bush

In this section we introduce a new format of the syntactic annotation that minimizes the problems described above and enables cheap, intuitive and ambiguous annotation on the syntactic level. We start with a formal description of a syntactic bush which is an analogy of syntactic tree (compare to description of dependency and phrasal trees in Section 2.2). Then we explain its semantics, introduce our particular annotation schema and discuss various aspects of the annotation.

3.2.1 Formal description

Let $S = (w_1, \ldots, w_n)$ be a sequence of words that forms a sentence. We define a *phrase* as a subsequence of the sentence:

$$(w_{i_1}, w_{i_2}, \ldots, w_{i_m})$$

where $i_j < i_{j+1}$ for all j. (Note that this definition allows disconnected phrases.)

Let *type* be an arbitrary string (e.g. "NP") from a set T. We define labelled phrase as pair (p, t) where p is a phrase and t is a type. We further define a weighting function w that assigns a number in interval $[0, 1]$ to each labelled phrase.

We further define *dependency* as any pair of labelled phrases (lp_1, lp_2) and a dependency weighting function dw that assigns a number in interval $[0, min(w(lp_1), w(lp_2))]$ to each dependency. (Note that this enforces that if $w(lp)$ is 0, both $dw(x, lp)$ and $dw(lp, x)$ are 0, for any other labelled phrase x.)

Further, let P be set of all labelled phrases that are assigned a positive weight by the function w. For each labelled phrase lp, we require that

$$\sum_{x \in P} dw(lp, x) \leq w(lp)$$

Let D be set of all dependencies that are assigned a positive weight by the function dw. Clearly, D is a binary relation over P. We do not impose

any further restrictions on this relation. Then, the syntactic bush for the sentence S is defined as tuple (S, T, P, D, w, dw).

3.2.2 Linguistic interpretation

It is probably clear from the definition above that the syntactic information encoded in bush consists of two layers: The first one is represented by labelled phrases or chunks (set P), the second one describes the relationships among the phrases (relation D). The relationships in our conception bear roughly the same linguistic information as dependencies in dependency trees (therefore, we also call it dependencies), however, unlike the tree dependency relation, D connects phrases, not words, and it does not have to be a function.

Another crucial feature of this approach to syntactic annotation is its fuzziness provided by functions w and dw: Each phrase, and each dependency, is assigned a probability of being correct which enables two attachment options e.g. for the phrase *"behind the forest"* in the sentence *"A plane crashed into the field behind the forest"* mentioned above.

3.2.3 Implementation

Of course, the content and value of the annotation depends not only on the used formalism but also on its practical implementation. For example, if we decided that every word is a phrase, values of the dependency weighting function are always 1 or 0, and each word must have exactly one dependency with weight 1, we will end up with an equivalent of pure dependency trees. (One of the important practical differences between bushbank and dependency formalism is that in bushbank, some tokens in the sentence may not be part of any phrase.)

We have carefully considered the drawbacks of the treebanks described above and proposed an annotation format that overcomes most of them. Above, we identified the key problems in treebank style annotation:

- it is expensive
- it requires extensive annotation manuals
- the underlying formalism is too restrictive (does not allow ambiguity)

We have already defined a formalism with greater descriptive power than dependency or constituent syntax, which solves the third point (and helps with the second). The particular annotation schema, annotation rules

and annotation process, including specialized software, should minimize the other two. We have developed annotation schema and annotation process on the Czech language but we performed a case study in English too, and are confident it generalizes well.

3.2.4 Annotation schema

As phrases (set P), we selected noun, prepositional, adjective and adverbial phrases, verb phrases, coordinations (with certain restrictions, see below) and clauses. Using labels (set T), we divide these phrases into 4 groups, as follows.

The first group consists of noun, prepositional, adjective and adverbial phrases. We define these phrases as maximal, i.e. only the biggest valid phrase should be annotated, except for PP-attachment – we split phrases on prepositions because we do not want to record PP-attachment and the related massive ambiguity at this level. We have labelled all of these as "PHRASE" (but the morphological information of the phrase syntactic head is stored as well, so we can consider the whole morphological tag as the label).[1]

Such definition has one advantage and one disadvantage. The advantage is, the notion of "PHRASE" is very intuitive and it takes 5–10 minutes and 2–3 paragraphs of text to explain what it is, and to cover borderline cases. The disadvantage is, we do not keep the information about the syntactic structure within the phrase. We decided it is a good trade-off – the phrase is usually short and its inner structure is easy to analyze automatically (e.g. [94] reports that parsing of short segments is easy). Also, this decision makes the annotation process faster (and therefore cheaper) and the resulting annotation more understandable.

The next type of phrase is verb phrase (label "VP") – this phrase type does not mark the maximal verb phrase including all noun phrase arguments and adjuncts, but all parts of the analytical verb form, including modal verbs and particles in case of phrasal verbs. Verb phrases are often disconnected in both Czech and English.

For practical reasons, we have introduced two other types of phrase: "COORD" for marking coordinations ("PHRASE" can contain coordinations but coordination of two "PHRASE"s is not a new "PHRASE", as their nesting is prohibited) and "CLAUSE" for marking clauses. (Let us point out that it is a problem to extract clause information from the Prague De-

1. In the linked materials, cited papers and annotation manual, these phrases are marked as "NP", to make the notion more intuitive for linguistically uneducated annotators.

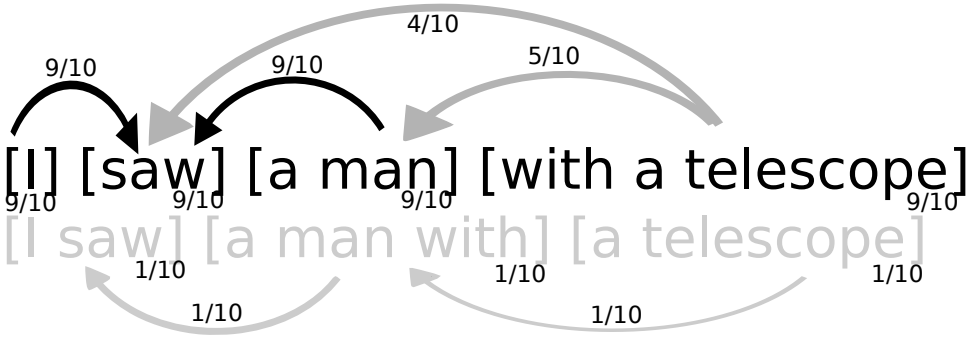


Figure 3.5: A model bush for the sentence *"I saw a man with a telescope"*. The sentence was annotated by 10 annotators, 9 of which agreed on correct segmentation and the unambiguous dependencies. The attachment of the phrase *"with a telescope"* can be recognized as ambiguous, thanks to the lower weights of both of its possible dependency edges.

pendency Treebank annotation, as illustrated by [100], so this is another advantage of our approach compared to the pure dependency formalism).

Dependencies (set D) among these phrases are then defined naturally: If a phrase A modifies another phrase B, then A depends on B. According to our experience with the annotators, there is a minimum number of annotation rules needed, and this level of annotation works very intuitively, mainly thanks to the specification of phrases described above.

The weighting functions w and dw are determined by percentage of inter-annotator agreement. If all the annotators of one sentence agreed on a phrase or a dependency, the weight of such phrase (dependency) is 1. If only half of them agrees, the weight is 0.5. An illustration of the annotation structure is given in Figure 3.5.

The whole annotation manual for Czech consists of five A4 pages (1300 words) including examples that form about 70 percent of the text, and it is attached to this work in Appendix A. In combination with the intuitive specification of the task, this made annotator training very fast and cheap, one hour task.

3.2.5 Annotation process

If the annotators would have to select phrases in text manually and then link them together, which is a common practice in annotation projects, it would be tedious and time-consuming work. Because of budget, need for

Figure 3.6: The interface of the *pHrase annotator* tool.

annotation speed and also because we wanted to limit creativity of the annotators to increase consistency, we decided to process the input text by automatic parsers and only let the annotators decide if the offered phrase is correct, or incorrect. Annotators were not allowed to mark phrases that were correct but were not found by automatic means.

When annotating dependencies, annotators were offered only phrases marked as correct, in a select menu (typically there were up to 10 phrases in a sentence, up to 5 within a clause) and the task was to assign a correct dependency to each phrase. If a covering clause was correctly identified, annotators could choose only from phrases within that clause.

We have developed a specialized GUI tool for annotators, the *pHrase annotator*. Within the tool, annotators can see the full sentence, together with automatically extracted phrases. Each phrase is given three radio buttons for "correct", "incorrect" and "don't know". If the mouse pointer hovers over the phrase candidate, the phrase gets highlighted in the sentence. Using the tool is illustrated in Figure 3.6.

3.2.6 Methodological aspects

Using automatic pre-processing and especially omitting the possibility of identifying correct phrase in case it was not found automatically, has significant implications. Mainly, such a resource will never be complete, i.e. we will never be able to cover all the correct phrases; there will always be some percentage unnoticed by the automatic tools. This poses a problem

with using such resource for evaluation of parsing precision – if we do not find a particular phrase on parser's output in the annotated data, and it does not overlap with an existing correct phrase, we cannot claim such a phrase is either good or bad. This implies that we cannot compute a precise number in case of both precision and recall, but just set the lower and upper bound for them, as suggested in [37, 4.5].

In practice, however, this problem is not so painful because a combination of automatic parsers is able to identify vast majority of correct phrases, over 90 percent according to our tests. The number of undecided cases is thus rather low and the span of precision and recall is small. Also, it is possible to complete the annotation, either by annotation of unknown phrases at the output of a particular parser under evaluation only, or by identifying all nouns (prepositions, adjectives, VP parts, etc.) that are not part of any phrase, and annotating those manually (or alternatively, offering the annotators all possible options). However, we did not perform this step, as it would make the annotation process more expensive (though still many times cheaper than creating a treebank) and we do not consider the problem of incompleteness as crucial for ongoing research.

Also, in case of using the data for parser evaluation, such resource may be accused of favouring the parsers that were used for its creation. There is not only the above discussed problem of missing phrases – the fact that the input is pre-annotated automatically may divert annotators to mark phrases as correct that are actually not correct according to manual, or that lie in the grey zone but may have been judged otherwise if there would be another or no preprocessing. Our answers to this objection are as follows:

- Automatic preprocessing is vastly used in creating manually annotated NLP resources; it has been used in Prague Dependency Treebank annotation [40] and it is generally reported as a positive decision (e.g. [162]) that makes the annotation process faster without significant disadvantages.
- Most if not all projects used only one preprocessing tool whereas we decided to use more, to make the resource as complete as possible and to avoid bias from automatic preprocessing as much as possible.
- Even if there was a significant bias, the fact itself that an automatic tool agreed with majority (or all) of human annotators (we expect to have more annotations of each sentence and use only the salient ones) is already enough evidence that the particular phrase should be marked as correct. If the annotation manual says otherwise, it is

rather a question if the manual is created in a right way (of course it might, especially if the particular rule is enforced by needs of an application, but the human-machine agreement is strongly significant anyway).

- Using preprocessing and limiting annotator's freedom increases the consistency of the annotation because the automatic parser is usually right (at least in the simple cases), whereas human annotators often dive into full linguistic complexity of language phenomena, even the trivial ones, and as a result they differ in the annotations where they should not. Our aim was not to perform deep linguistic analysis but rather create a viable resource in a cheap way and therefore consistency gets higher priority than linguistic correctness and completeness.

To sum up this section: We admit the problem with incompleteness of the resource. However, there are solutions to overcome it, and it brings important advantages, mainly speed and price of the annotation process. We do not accept the objection that our procedure causes bias to the resulting annotation.

3.2.7 Technical details

As the annotation schema proposed above is relatively complex, we used the NITE NXT toolkit [18] written in Java (and originally dedicated to multimodal corpora) which allows easy annotation on multiple layers, using an XML format. We have distinguished word layer, morphological layer, phrase layer and dependency layer. A specialized library, *libBushBank* was created by Marek Grác for convenient work with the bushbank data [37, 4.4.2]. Among other functions, the library offers different options of exporting and simplifying the data according to various criteria (e.g., export only the annotations where all annotators agreed) or converting such simplified data into a more readable format, such as the vertical text format.[2]

The pHrase annotator tool introduced above was also written in Java so that all the annotators could use the same binary package on their local machines, independently on platform.

2. `www.sketchengine.co.uk/documentation/wiki/SkE/PrepareText`

3.3 Case studies – Czech and English bushbank

We have created several initial resources according to the methodology described above. We have started with adding syntactic annotation to the DESAM corpus [137] that already contains manually disambiguated morphological tagging. The first 10,000 sentences were annotated 3–10 times each. During this phase of annotation, we tuned the annotation manual; we measured its quality by inter-annotator agreement (IAA) and achieved a shift from "moderate/substantial" (87% average pairwise agreement; 0.65 in terms of Cohen kappa [26, 17]) to "excellent" (92% average pairwise agreement and Cohen kappa 0.80), according to kappa interpretations by Landis and Koch [96] and Fleiss [32]. Although the manual is very brief, these results indicate that our method can achieve highly reliable data.

The speed of the annotation stabilized on 100 sentences per hour, for one annotator. With average sentence length 20–25 words, such speed enables us to annotate a one million word corpus in about 11 person-weeks (with one annotation per sentence) which is about 10 times faster than in case of Prague Dependency Treebank [113], not counting the setup time before the annotation starts.

After setting up the manual, we have used the methodology for annotating a small (2400 sentences) highly specialized corpus of cooking recipes, for the purpose of a proof-of-concept project on text entailment in Czech [126]. The corpus has proven suitable for the application which has been another evidence that our approach to syntactic annotation is meaningful.

As there are copyright issues with texts in the DESAM corpus that do not allow to share it publicly, and also because the texts and the language within this corpus are rather outdated, we decided to create a new reference corpus for bushbank syntactic annotation from internet blogs. At the current time, the annotation is in the middle, with target size 1 million tokens. We have agreed with the authors of the texts that the corpus will be available to public under one of the free licences. Although the annotation is not yet finished, the corpus has already been used for training of a new precise chunker for Czech [144]. The morphological annotation of this corpus is only automatic, obtained by the Desamb tagger [171].

Annotation of Czech corpora used the following parsers for preprocessing:

- the SET parser [90], also described in Chapter 5, with 4 different grammar modifications, including specialized grammar for detec-

tion of verb phrases, a result of bachelor thesis [8], further referred to as *SET – VP*.

- the Collins' parser adapted for Czech and distributed with Prague Dependency Treebank [28], together with an algorithm for converting dependency trees to the bushbank format
- MST parser [110] and MaltParser [127], both trained on Prague Dependency Treebank and combined with an algorithm for converting dependency trees to the bushbank format

We have also started creating an English bushbank. We have adjusted the annotation manual, so that it follows the same principles as the Czech one, however, they differ in particular details, naturally. The preliminary English manual is attached to this work in Appendix B. Up to now, we have annotated an English bushbank of 700 sentences, by 2–3 annotators each. TreeTagger [158] was used for morphological analysis and both TreeTagger and an English grammar of the SET parser ([90], see also Section 5.4) were used for preprocessing the syntactic layer. The annotators were Czech students of English from the Faculty of Arts at Masaryk University, rather than native speakers of English, which we consider the biggest problem with this annotation and we are currently trying to intensify our collaboration with native English annotators.

3.4 Usages

Up to now, apart from the above mentioned entailment project, the Czech bushbanks were used for evaluating parser's ability to detect text chunks (PHRASEs and VPs, as described in Section 3.2.4). The results against the testing section of the blog bushbank are shown in Table 3.1 (they were already partly published in [144]). The parsers involved in this evaluation were the ones listed above, that were used by creation of the Czech bushbanks. Phrases where the inter-annotator agreement was over 50% were used as the gold standard.

Also, the blog bushbank was used for training a new statistical chunker for Czech, based on the Polish IOBBER [145]. Its results, included in Table 3.1, show that in case of PHRASEs, the statistical chunker outperformed the tools that were used for creating the resource. In case of VPs (that are often disconnected in Czech), a specialized grammar for the SET parser was the best.

VPs

Parser	Precision (%)	Recall (%)	F-score (%)
SET	80.2	85.0	82.5
SET – VP	**89.4**	**95.5**	**92.4**
MaltParser	50.7	52.3	51.5
MST Parser	50.9	53.0	51.9
IOBBER	81.6	90.7	85.9

PHRASEs

Parser	Precision (%)	Recall (%)	F-score (%)
SET	74.7	89.5	81.4
Collins	73.2	72.8	73.0
MaltParser	44.3	56.3	49.6
MST Parser	44.1	56.8	49.7
IOBBER	**85.9**	**90.3**	**88.1**

Table 3.1: Results of parser evaluation against Czech bushbank data.

Two remarkable conclusions can be drawn from these results. First, if an appropriate statistical machinery (in our case the IOBBER chunker) is trained on the right data, it will produce very good result, independently on annotation guidelines. If a statistical tool is trained on data with different annotation style (in our case Collins, MaltParser and MST Parser, all trained on PDT), they will perform significantly worse. In this case, all these three were outperformed by the rule-based parser SET, and a negative correlation can be observed (-0.83) with the dependency precision on PDT, as shown in Table 3.2.

Second, if a specialized rule-based tool is tailored to a particular task, it can outperform the statistical approach. In our case, the specialized VP grammar of the SET parser was better than the statistical IOBBER, probably because of the disconnected nature of Czech verb phrases. Both of these conclusions are consistent with our arguments in Section 2.5.

The speed of the annotation together with the accuracy of the IOBBER training algorithm makes it possible to create domain-specific statistical chunkers, using new domain-specific bushbanks, with a reasonable amount of financial resources, which was not possible before. This may be one of the possible future research directions.

Parser	PDT dependency precision (%)	PHRASE F-score (%)
IOBBER	N/A	90.3
SET	56.0	81.4
Collins	80.9	73.0
MST Parser	84.7	49.7
MaltParser	85.8	49.6

Table 3.2: PDT dependency precision vs. bushbank F-score on PHRASEs.

3.5 Conclusions

In this section, we have introduced the bushbank concept of syntactic annotation together with particular case studies, that aimed to overcome some of the problems of treebanks that are currently leading syntactic resources. We have shown that

- creating a bushbank resource is faster and cheaper than creating a treebank
- it can be used for training chunkers and parsers as well as their evaluation, so it is a viable alternative to treebanks
- it allows annotating ambiguous and indefinite phenomena in a very natural way, using multiple annotators and their agreement
- the annotation manual can be very short while gaining high inter-annotator agreement
- thanks to this, the resulting annotation is much more intuitive and understandable for people not specialized in any particular linguistic theory
- the annotation result is much more transparent compared to the treebanks and does not contain senseless annotations forced by the formalism
- thanks to the automatic preprocessing and strong annotator restrictions, the consistency of the resource is well guaranteed

We have also discussed some possible disadvantages of the approach, namely

- the resulting resource is not complete
- the annotation is less fine-grained than in case of treebanks

We have shown that the first issue is not a problem in many cases, and it would be easy to overcome; and we consider the second point rather an advantage that helps readability and usability of the resource.

We have also presented initial parser evaluations based on the Czech bushbank indicating that the evaluation results depend heavily on the gold standard data annotation style; namely, accuracy figures for PDT and Czech bushbank were dramatically different.

Chapter 4

Sketch grammar: A shallow approach to syntax

In this chapter, we introduce the concept of sketch grammars which represents an application driven approach to the syntactic analysis, according to the "worse is better" principle introduced in Section 2.5.5. The idea was originally proposed by Kilgarriff, Rychlý et al. [73] for extraction of two word collocations within named grammatical relations, called a word sketch. The concept has proved to be very popular and commercially successful within the Sketch Engine system and since its introduction, it has been enhanced and adapted for a larger spectrum of tasks – namely finding collocations of more than two words, term extraction and bilingual collocation extraction. The author of this thesis was substantially involved in the design and development of these extensions, all of which were already introduced on international forums and have their commercial users within the Sketch Engine system. Also, an extensive quality evaluation of word sketches has been done, with author's significant contribution. The evaluation and its results are described later in Section 6.5.

We firstly describe the word sketch functionality and the formalism that it is based on; this part has been mostly in place before we started our work on it, so it is rather introductory and does not present original contribution of this work. Then we describe the particular extensions of the functionality that we were substantially involved in – this second part does present an original contribution of this work, and describes author's joint work with Adam Kilgarriff, Pavel Rychlý, Miloš Jakubíček, Vít Baisa and Vít Suchomel, with author's contribution of ca. 50 percent.

4.1 Basic formalism

Word sketch application was designed to provide users with a one-page, automatic corpus-based summary of a word's grammatical and collocational behaviour [73]. An example of a word sketch user interface is pre-

water *(noun)* **British National Corpus freq = 34246** (305.3 per million)

modifier	9591	1.1	object_of	5126	1.6	subject_of	2835	1.7
hot	665	10.17	pump	92	8.82	**flow**	113	9.29
drinking	352	9.97	**pour**	139	8.74	drip	36	8.33
cold	459	9.63	**drink**	133	8.55	seep	30	8.2
boiling	237	9.58	heat	72	8.43	gush	23	7.94
fresh	231	8.81	boil	55	8.12	boil	25	7.62
mineral	173	8.76	tread	43	7.88	cascade	17	7.48
running	145	8.74	fetch	41	7.51	swirl	18	7.45
Thames	140	8.54	splash	29	7.26	lap	17	7.43

Figure 4.1: A word sketch for word *"water"*. The most salient collocations of the word are displayed sorted into grammatical relations.

sented in Figure 4.1 and we can see that it is a model instance of a collocation extraction application, as discussed in the introduction.

Its crucial feature is that the extracted information comes from a natural language corpus. Therefore, it is closely related to the features of a corpus manager program, in this case the Sketch Engine which includes Manatee and Bonito [147] as components, and its implementation of the corpus query language CQL [66].

The basic idea of the approach is to combine statistical measures for finding collocations with rule-based linguistically motivated filtering of collocations and their classification into grammatical relations. There is a number of purely statistical tests for finding collocations such as the T-test, mutual information, Dice coefficient [102] or even raw frequency, all of which have their strengths and weaknesses and can be also used within the Sketch Engine system. The word sketch application itself uses logDice, a modified Dice score proposed by Rychlý for better interpretability [148].

4.1.1 Query language grammar

The novelty of the word sketch approach consists in combining the statistics with a type of manually defined syntactic rules that limit what counts as a co-occurrence of particular two words – a *sketch grammar*. The core

```
=subject

    2:[tag="N.*"]  [tag="RB.?"]{0,3}  [lemma="be"]?
        [tag="RB.?"]{0,2}  1:["V.[^N]?"]

    1:[tag="V.N"]  [tag="RB.?"]{0,2} [word="by"]
        [tag="N.*"]{0,2}  2:[tag="N.*"]  [tag!="N.*"]

=modifier

    2:[tag="(JJ|N).*"]  [tag="JJ.?"|tag="RB.?"]{0,3}
        [tag="N.*"]{0,2}  1:[tag="N.*"]  [tag!="N.*"]
```

Figure 4.2: Example of a sketch grammar.

of a sketch grammar is a set of queries in corpus query language CQL,[1] each with marked position of the two words that can form a collocation. Only the words matching one of the queries are then considered as co-occurrences for statistical computations. Each of the CQL queries is associated with a label that describes grammatical relationship between the particular two words. One grammatical label (or relation) can be assigned to multiple queries.

Figure 4.2 shows a simplified and incomplete example of a sketch grammar that captures only collocations in position of verb subject (relation "subject") and a modifier of a noun (relation "modifier"), for an English corpus morphologically tagged using the Penn Treebank tagset.[2] Grammatical relation names are on lines starting with "=" and are followed by CQL queries with particular positions marked as "1:" (this word will display as headword) and "2:" (this word will display as a collocate in a word sketch table with heading "subject").

The first subject query describes a noun, followed by 0–3 adverbs (tags matching "RB.?"), possibly by "be" and other 0–2 adverbs, and a verb not in form of passive participle. Such a query would match e.g. the expression "ideas furiously sleep" and if the corpus contains such expression, one hit of co-occurrence of "idea" and "sleep" within the subject relation would be recorded for the purpose of statistical computations.

---

1. A short CQL tutorial can be found at
`www.sketchengine.co.uk/documentation/wiki/SkE/CorpusQuerying`
2. `www.sketchengine.co.uk/documentation/wiki/tagsets/penn`

The CQL queries in the sketch grammar are evaluated against the whole corpus and for matching pairs of words, the statistics are computed and indexed in a specialized database. Access to the indexed data is provided by *WMap* library which is a part of Manatee. Processing of a 100 million word corpus with a sketch grammar of 38 CQL rules, in parallel on 8 CPU cores, takes about 17 minutes which probably makes it a fastest partial dependency parser in the world.

4.1.2 Processing directives

Several enhancements of this basic format have been developed, the usages of which are marked by keywords called *processing directives*. For example, the default definition introduced above is asymmetric, e.g. only "idea" is identified as a subject of "sleep" and not vice versa. Also, we would not want "sleep" to be identified as subject of "idea". The *DUAL processing directive makes the relationship bidirectional and makes it possible to name the inverse relationship differently. An example of a dual rule is shown in Figure 4.3.

Another possibility is to include a third word into the relationship, more precisely into a grammatical relation name. This is done by the *TRINARY directive – a third word can be labelled by "3:" within the CQL queries and this word replaces the string "%s" in the relation name, potentially creating a large number of different grammatical relations. For example, the *TRINARY rule in Figure 4.3 describing a noun followed by a preposition, possibly a possessive pronoun, more nouns and a non-noun, would match e.g. expression "test of my patience and", putting "patience" into relation with "test", under the label "after_of", and vice versa, under the label "before_of".

Another related directive *COLLOC enables collocations listed within the tables to capture more complex units than just words from the corpus. For example, we can concatenate lemmas of more words (by continuing labelling words in the CQL as "3:", "4:" etc.), or use a different attribute than (default) lemma, such as semantic tag, if available. Unlike the previous two, this directive relates to a particular CQL query, rather than to the relation, so the keyword is to be put before each relevant query. Examples in Figure 4.3 show both mentioned cases and the exact coding. The first rule would put items such as "positive-feature"[3] into the "semantic_modifier" relation

3. Of course, the exact values would depend on the semantic tagging present in the corpus. Here we assume that the corpus contains an attribute named "semantic" that would assign the "positive-feature" value to words like "cute".

```
*DUAL
=subject/subject_of

    2:[tag="N.*"]  [tag="RB.?"]{0,3}  [lemma="be"]?
        [tag="RB.?"]{0,2}  1:["V.[^N]?"]

    1:[tag="V.N"]  [tag="RB.?"]{0,2} [word="by"]
        [tag="N.*"]{0,2}  2:[tag="N.*"]  [tag!="N.*"]

*DUAL
=modifier/modifies

    2:[tag="(JJ|N).*"]  [tag="JJ.?"|tag="RB.?"]{0,3}
        [tag="N.*"]{0,2}  1:[tag="N.*"]  [tag!="N.*"]

*DUAL
*TRINARY
=after_%s/before_%s

    1:[tag="N.*"]  3:[tag="IN"] [tag="PP\$"]?
        [tag="N.*"]{0,2}  2:[tag="N.*"]  [tag!="N.*"]

=semantic_modifier
*COLLOC "%(2.semantic)"

    2:[tag="(JJ|N).*"]  [tag="JJ.?"|tag="RB.?"]{0,3}
        [tag="N.*"]{0,2}  1:[tag="N.*"]  [tag!="N.*"]

=two_word_modifier
*COLLOC "%(2.lemma)_3.lemma"

    [tag!="JJ.*"]  2:[tag="JJ.*"]  3:[tag="(JJ|N).*"]
        1:[tag="N.*"]  [tag!="N.*"]
```

Figure 4.3: An example of a sketch grammar with *DUAL, *TRINARY and *COLLOC relations.

with e.g. "cat", through expressions like "cute cat". The second example would gather "sweet_little" as a collocate for "cat", from expressions like "sweet little cat".

As we can see from the example, the directives may be combined arbitrarily.

There are more processing directives that make the work with sketch grammars easier and extend the default possibilities.[4] Here we will not go into more detail – we have introduced the main features of the sketch grammar formalism necessary for further discussion.

4.1.3 Coverage

Thanks to the simple concept, efficiency and worldwide intensive usage of the word sketch application (about 4,000 active users at the time), there are sketch grammars available for nearly 40 languages, mostly written by native speaker linguists. In our opinion, such a coverage is a strong support for our "worse is better" philosophy. Although it is not following any particular linguistic theory, and although it is a very simple approach which inherently makes many errors and is incomplete, the sketch grammar concept of syntactic analysis is very suitable for needs of thousands of people, unlike any other natural language parser in the world.

4.2 Extensions

Based on intensive usage of the word sketch application and user feature requests, the word sketch application has been extended to serve more applications than two-word collocation extraction. All of them are based on the features described in the previous section.

4.2.1 Multiword sketches

Since word sketches came to existence, there was demand for the same information for multiword units, like "take advantage" (which is very different from "take") or "climate change" (which is different from "change"). Technically trivial solution would be to tokenize the corpus according to the multiword units, however, it has serious drawbacks:

- with this solution, collocates would be multiwords as well which is not desirable

4. `www.sketchengine.co.uk/documentation/wiki/SkE/GrammarWriting`

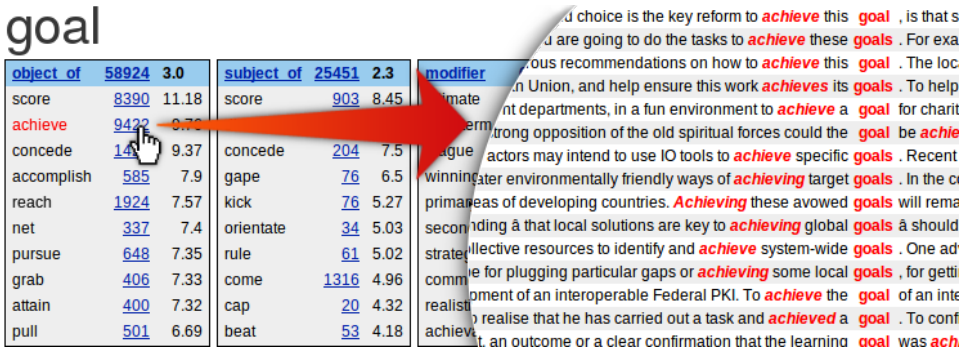


Figure 4.4: Illustration of transition from word sketch to corpus concordance (source: `www.sketchengine.co.uk`).

- morphological tagging of multiwords is problematic
- multiwords do not have to form connected phrases (e.g. English phrasal verbs like "look up")
- multiword boundaries are not clear – e.g. should we consider "new generation" as one unit or two?

We have implemented a method which allows users themselves to decide what expressions are interesting, and where no special tokenization of the corpus is needed [72].

The *WMap* library managing access to the indexed word sketch data, provides methods for filtering the word sketch information according to a particular list of corpus positions. Corpus positions enumerate word occurrences in the corpus – first word has position 0, second word 1, etc. So the list of positions may be e.g. result of a query, or first million words in the corpus, or words in a particular text type. At the same time, the word sketch database contains a reverse index for each word-relation-word triple that enables creating the list of corpus positions for the particular triple. This index is used for links from the word sketch tables to the actual occurrences of the collocation in the corpus, in form of a corpus concordance (see Figure 4.4 for illustration).

By combining the two features mentioned above – filtering the word sketch database, and the reverse index – it is possible to create e.g. a word sketch for "take" only for occurrences where it goes with "advantage": We use the index for the triple take-object-advantage to get the list of its positions and then filter the word sketch for "take" by this list.

By this procedure, we can get collocations for "take" but not for "advantage". To be able to show e.g. "full" in the list of collocations, to capture the

frequent pattern "take full advantage", we need to do this symmetrically, by combining word sketch for "take" filtered by "advantage" with word sketch for "advantage" filtered by "take".

This approach does not require a special tokenization nor multiword detection algorithms, and it is language independent, except the basic sketch grammar. Also, multiwords do not need to form a connected part of the sentence (but they may, depending on the CQL rules in the sketch grammar).

The end users of the multiword sketch application have two options of getting to a multiword sketch. The first follows the filtering procedure as introduced above, using links connected to particular collocates, as illustrated in Figure 4.5: User e.g. calls a word sketch for "water", then they click on "hot" to get a word sketch for "hot water". The process is iterative, so they may continue by clicking on "soapy" to get a word sketch for "hot soapy water". The second option is faster and probably more convenient for users – they can enter directly "young man" as input and a special heuristic procedure tries to guess the correct grammatical relation which is then used for filtering, based on frequency and part of speech selected by user.

4.2.2 Bilingual word sketches

Since 2005, there has been demand for bilingual word sketches by the lexicographic community, which would help lexicographers in compiling bilingual dictionaries, and language learners in finding translations of particular collocations easily. However, there was never a concrete specification and it was not clear how such application should look like. We have developed three approaches to the bilingual word sketch challenge.

The first and most straightforward is referred to as **BiM** (bilingual **m**anual) – manual means that users themselves specify the translation of the headword into the target language. Result is a word sketch in two languages, with aligned compatible grammatical relation columns, as illustrated in Figure 4.6. There is no attempt to align the individual collocations.

To be able to align compatible relations, either they must have same names, in corpora of different languages (i.e. we need compatible sketch grammars), or a mapping must be created between the relations to specify which of them are compatible (this is the usual scenario). We have selected to map relation names to a common base, rather than mapping each language pair individually, as it reduces the amount of necessary manual work from n^2 to n, with respect to the number of the languages (or sketch grammars). The mappings are defined in the sketch grammar using a newly in-

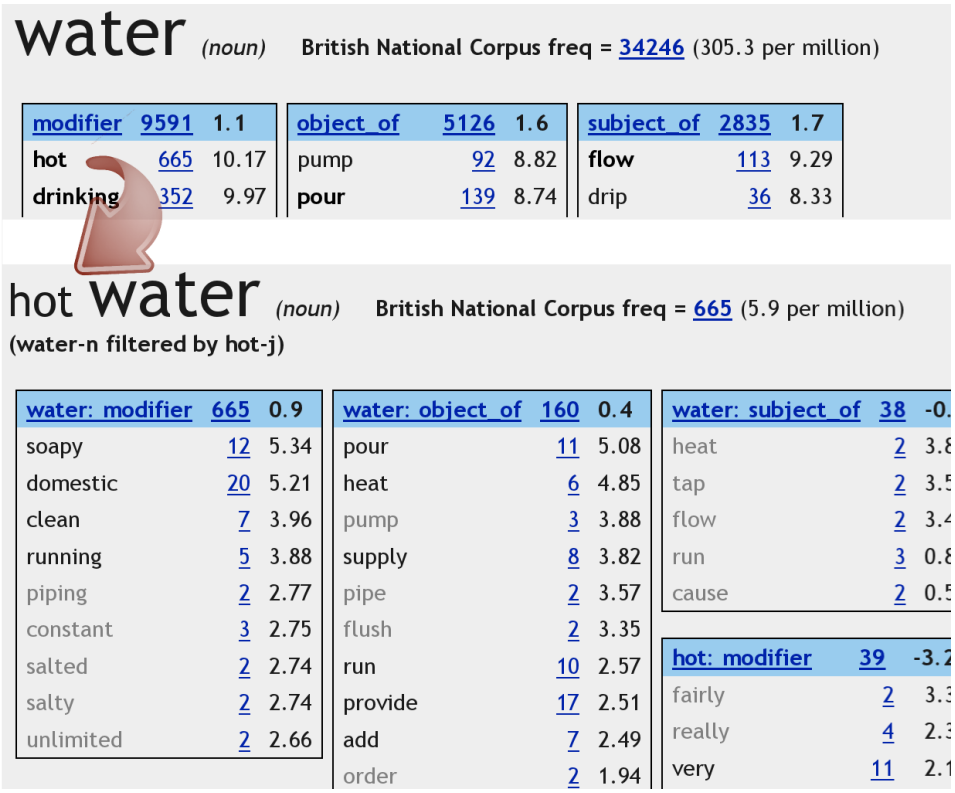


Figure 4.5: Transition from a word sketch to multiword sketch.

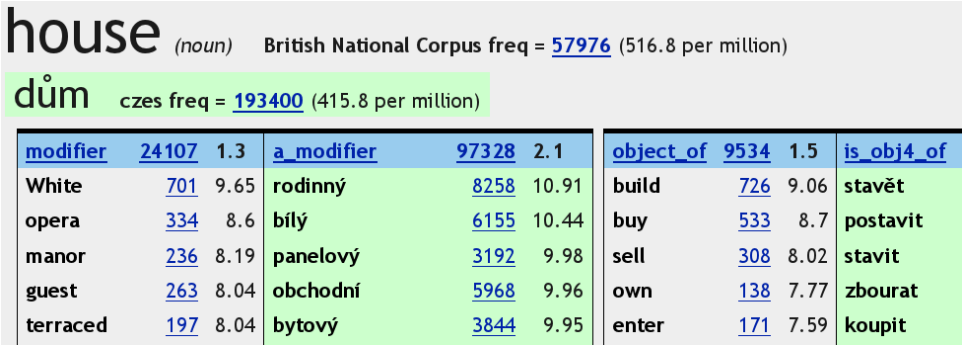


Figure 4.6: Bilingual manual word sketch.

```
*DUAL
=objet/objet_de
*UNIMAP object/object_of
...
```

Figure 4.7: Example of using the *UNIMAP directive.

troduced processing directive *UNIMAP. The definition in Figure 4.7, from a French sketch grammar says that relation "objet" should be aligned to "object" and "objet_de" should be aligned to "object_of" (or the appropriate relations in other languages). We have created the mappings of the core grammatical relations for 10 major languages, in collaboration with native speakers of the respective languages. Once the mapping is available, this functionality can be used with any pair of corpora of the two particular languages.

The other two approaches to the bilingual word sketch challenge find the headword translation to the target language automatically, and align particular collocations with their equivalents in the target language. For this task, we need a pair of comparable corpora (this variant is referred to as **BiC** – bilingual **c**omparable) or a parallel corpus (this variant is referred to as **BiP** – bilingual **p**arallel) which is implemented as two independent corpora aligned to each other, within the Sketch Engine.[5]

For **BiC**, we need a bilingual dictionary with a very good language coverage and as many translation equivalents as possible, for each word. Initially we were experimenting with Google dictionaries API, then the API stopped to be freely available, so currently we try to build the dictionaries from different available sources such as Wictionary[6] (the quality of the dictionary is much less important for this task than coverage).

Having a dictionary, we select the most salient (or the first, if there is no information on salience) translation of the headword and create a word sketch for this translation. We also show links to other possible translations, in case that the first translation is not the desired one. Then we have two word sketches: In the source ($ws1$) and target language ($ws2$). For each word in $ws1$, we seek for its translations in $ws2$, regardless grammatical relations. If one or more possible translations are found in $ws2$, the particular $ws2$ collocates are aligned under the particular $ws1$ collocate, as illustrated in Figure 4.8. For better readability, we do not include grammatical relation

5. www.sketchengine.co.uk/documentation/wiki/SkE/Parallel
6. www.wiktionary.org

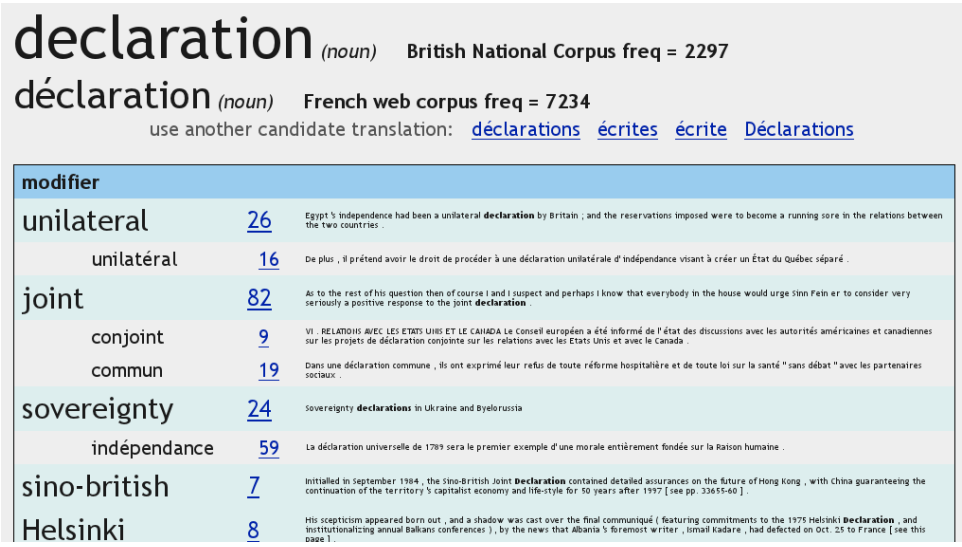


Figure 4.8: Example of a BiC word sketch.

names from $ws2$; rather than that, we supplement each collocation with an example of its usage (obtained asynchronously, using the mentioned reverse index).

The **BiP** approach is very similar to **BiC**, with two exceptions. First, the external dictionary is not needed. Thanks to the alignment (usually by sentences), we can compute probability that word X translates as Y, for each pair of words. Then we sort all possible translations for each word according to this probability and save the top n to a dictionary (currently we use 10). In fact, we do not even need to calculate probability, a ranking is sufficient – currently we are using the same logDice co-occurrence coefficient that is used for scoring word sketch collocations. In the future, we plan to test algorithms implemented within the well-known GIZA++ tool [133], however, we do not assume big differences, as the quality of the dictionary is not crucial for the task, as we have already mentioned.

The second difference is that we can verify the correctness of the collocation alignment to a larger extent. Thanks to the corpus alignment and mentioned word sketch reverse index, we are able to create a list of positions for both source and target language collocation pairs, and check if at least some of them are aligned to each other. If not, the particular alignment is probably a mistake in the collocation dictionary or a miss in the alignment algorithm, and it will be hidden from the user. This check is computationally demanding (we need to do it for all collocation pairs), so we

have implemented this checking (again) asynchronously within the Sketch Engine.

4.2.3 Word sketches for terminology extraction

Based on a requirement from a partner institution, we have also adapted the word sketch functionality for terminology extraction. Using the sketch grammar was not the first idea for terminology extraction we had – previously we tried working with n-grams – but the empirical evaluation by the partner showed that accommodating sketch grammar machinery for this purpose works better.

The Sketch Engine has a well developed functionality of finding keywords (single words specific to one corpus or subcorpus, with respect to another, general, or reference one), based on a straightforward formula called Simple Maths [70]:

$$keyness = \frac{freq\_per\_million + n}{freq\_per\_million\_in\_reference\_corpus + n}$$

where *freq* may be any type of frequency (within the Sketch Engine, it is possible to use raw frequency, average reduced frequency [157] or document frequency), and $n > 0$ is a smoothing constant – higher values prefer frequent words, lower values prefer rare words. We have a very good experience with this type of statistics; it is very transparent and it does not have significant drawbacks.

For term extraction, no modifications of the sketch grammar formalism were needed; novel is the usage. We combined CQL queries recognizing multiword units that can be considered term candidates – mostly noun phrases – with extensive usage of the *COLLOC directive described earlier, to create and index the particular term strings. Then, we applied the existing Simple Maths statistics to the term candidates extracted from a specialized corpus and term candidates extracted from a general corpus. As a result, we get a list of term candidates, sorted by their "termhood", as illustrated in Figure 4.9.

4.3 Conclusions

The sketch grammar, together with related indexing tools, is a specialized shallow parsing engine that has been designed independently on any linguistic theories and treebanks, for a particular task. Despite that fact and despite the actual simplicity of its design, it fulfills its purpose very well

```
=terms
*COLLOC "%(2.lc)_%(1.lc)"

  2:[tag=="NN" | tag=="JJ" | tag=="VVG"]  1:[tag=="NN"]

*COLLOC "%(3.lc)_%(2.lc)_%(1.lc)"

  3:[tag=="NN" | tag=="JJ" | tag=="VVG"]
        2:[tag=="NN" | tag=="JJ" | tag=="VVG"]
        1:[tag=="NN"]
```

Term	Frequency	Freq/mill	Score
carbon dioxide	373	3864.3	37.5
global warming	317	3284.1	30.8
water vapor	71	735.6	8.3
greenhouse effect	69	714.8	8.1
greenhouse gas	71	735.6	8.0
climate change	78	808.1	7.6
industrial ecology	27	279.7	3.8
fossil fuel	26	269.4	3.6
surface temperature	20	207.2	3.1
carbon cycle	19	196.8	3.0

Figure 4.9: Example of two CQL rules from the English term grammar, and part of the result term list for a domain environment corpus.

and serves thousands of users worldwide. We have shown that it has potential to be extended and adjusted to other specialized practical tasks, namely collocation extraction for multiword units, bilingual collocation extraction, and extraction of terminology. Also, there is ongoing work in this direction trying to combine term extraction with bilingual word sketches, to address the problem of automatic bilingual terminology extraction.

The story of sketch grammars, including our contribution to it, forms another support point for our general methodology considerations – namely that design simplicity is the most important value, and that it is necessary to start with a practical application when designing a parsing system, rather than a language theory.

Chapter 5

SET – a light-weight parsing system

The SET parsing system,[1] firstly introduced in [90], was designed by the author according to the software development principles introduced in Section 2.5.5. Namely, design simplicity was always the highest priority that we took into account in all phases of development.

5.1 Initial considerations

The system was firstly developed for Czech and we originally tuned it against the Prague Dependency Treebank data with interesting results, top among rule-based parsers and not very far from the best statistical parsers [90], as summarized in Table 5.1.

Then, because of discovering the methodology problems discussed in Section 2.5, we changed the direction of development towards the needs of applications. For the bushbank project described in Chapter 3 (and few others), we needed phrase detection, so we prepared a few versions of the grammar for phrase detection. For grammar checking, a different type of output was needed, etc. We have dropped the idea of good results against a treebank, and aimed the development at being able to provide what applications need, with simplicity in design as the highest value. And it is that simplicity, thanks to which the parser is relatively easy to adapt to a large variety of requirements. Dependency precision of the current version of SET, compared to the PDT data, is around 55 percent.

The two other crucial values that were prioritized in the design, are usability, and readability of the results. As discussed earlier, we consider the traditional dependency and phrasal formalisms too cryptic for a user who is not familiar with them in detail, and enforcing a lot of unexploited and non-intuitive data. We have designed a new tree format that we call hybrid tree that combines dependency and phrasal components. We describe this

1. SET is an abbreviation of "syntactic engineering tool" and accidentally, it is also a god of the desert, storms, disorder, violence and foreigners in ancient Egyptian religion.

Testing set	Average precision (%)	Median precision (%)
PDT e-test	76.14	78.26
PDT 2000	83.02	87.50

Table 5.1: Historical results of the SET parser against the testing part of the Prague Dependency Treebank, using unlabelled dependency precision. These results were measured with SET version 0.3. The "PDT 2000" set consists of 2000 sentences randomly selected from PDT that were accepted by the grammar of the Synt parser [83] – by this experiment, we wanted to measure accuracy on well formed sentences.

format in the next section; as we will see, its expressivity is the same as in case of the traditional formalisms, with much more intuitive reading.

Apart from that, many output options have been implemented, including regular dependency and phrasal trees, a bush output (with unambiguous phrases and dependencies), output of all detected phrases and collocation information, so that it is easier to adapt the output of the parser to a particular application.

For the sake of both simplicity and usability, we have selected the Python programming language for the implementation. Python is a very powerful and readable language which itself makes orientation in the program code easy, thanks to which the development speed is very high (which relates closely to possibilities of flexible changing of the system behaviour and fast implementation of new features [104]). As Python is an interpreted language, its disadvantage is worse performance compared to languages like C, C++ or Java. On the other hand, the application speed is not that important in a research-aimed tool; rather than that, the speed of development and straightforward usability is crucial. Once the tool is built into a practical application, it can be further optimized, if needed. Also, despite lower Python efficiency in general, the parser's speed (about 7 sentences per second, on one CPU core) is comparable to most state of the art parsers.

Python also enables a "download and run" usage; no compilation and no installation is needed, which further increases usability of the tool. One exception is a dependency of system's graphical output on Python bindings for the Qt4 library that are not part of standard installations of Python.

5.2 Hybrid trees

Let $S = (w_1, \ldots, w_n)$ be a sequence of words that forms a sentence. Let N be an (arbitrary) set of nonterminals and L be a set of *syntactic labels*, that will express the syntactic role of the word in the sentence. *Hybrid syntactic tree* is defined as a tuple (S, N, L, d, l) where

$$d : \{w_1, \ldots, w_n\} \cup N \ \rightarrow \ \{w_1, \ldots, w_n\} \cup N$$

is the dependency function and

$$l : \{w_1, \ldots, w_n\} \cup N \ \rightarrow \ L$$

is the labelling function assigning a string from L, to each word or nonterminal (analogously to dependency trees). One string from L is dedicated for marking a special, phrasal type of dependency, that will be displayed in a different colour in the hybrid tree illustrations.

The main difference between dependency and hybrid trees is the presence of nonterminals (we also refer to them as phrasal nodes) and phrasal dependencies. The nodes connected to a phrasal node by phrasal dependencies are considered components of the particular phrase, in the same sense as in phrase structure formalism. It is allowed for the phrasal node to be a target of both phrasal and common dependencies.

As always, the readability and usability of particular trees depends on the particular implementation of the formal constraints, especially particular content of the sets N and L. The L set remains the same as in case of dependency trees, except the special symbol for phrasal dependency. As for set N, within our current schema we use the following nonterminals:

- `<sentence>` – the top symbol that covers the whole input sentence.
- `<clause>` that marks clauses within the sentence.
- `<inter>`, covering segments on the level of clauses or immediately below it (e.g. segments interrupting clauses) that do not contain verb and therefore cannot be marked as clauses, e.g. expressions in brackets, comma-delimited or in quotation marks.
- `<attr>`, a special case of `<inter>` marking comma-delimited modifiers in apposition.

- <coord>, probably the most important, for marking the coordination relationships where all parts of the coordination are on the same level of importance.
- Various non-terminals for named entity type of expressions, such as <name>, <date>, <code>, <number>. Such kind of low-level phrasal nodes can be also used e.g. for multiword prepositions like "instead of" (the nonterminal here could be <prep>). However, it is debatable to what extent should be these items identified by syntactic analysis and if it should not be rather part of the preprocessing, done by the named entity recognition tool, or even a tokenizer. We are currently experimenting with external named entity recognition tools.

Thanks to the hybrid tree format, we can combine the advantages of the dependency and phrasal formalisms discussed in Section 2.2.3. The coordination nonterminals were the main motivation for the proposal of hybrid trees, as the pure dependency trees are not able to record the coordination phenomenon in a natural way, which subsequently leads to confusing annotations and makes coordinations hard to be recognized correctly in dependency parsing. On the other hand, hybrid trees allow easy annotation of non-projective constructions by the dependency means. (This idea is not new [56] but the current state of the art prefers rather purity of the formalism.)

Also, we consider dependency formalism better for describing complex noun phrases: In Section 2.2.3, analysis of phrases like "new generation of fighters" is discussed as a disadvantage of the dependency formalism, however, in real sentences it is usually hard to decide the right bracketing, e.g. if "(low speed) of transfer" is better than "low (speed of transfer)", so it would rather bring a lot of unnecessary, probably often inconsistent decisions, and noise into annotation.

Of course, the introduced implementation is not to be taken as a dogma: our belief is that it should be always the final application who decides the exact implementation and the formalism. We have already slightly changed the particular schema several times during the development of SET, and e.g. for an application for detection of Czech analytical verb forms, phrasal nodes were used to mark the verb phrases (as reported in [8]).

The most important feature of the hybrid trees is their ability of combining advantages of dependency and phrasal formalisms; the particular setting is secondary. The feature helps with readability of the trees, as illustrated in Figure 5.1, and we believe it can help with the parsing process as

well, in case of thoughtful usage, as different nature of different syntactic phenomena can be taken into account.

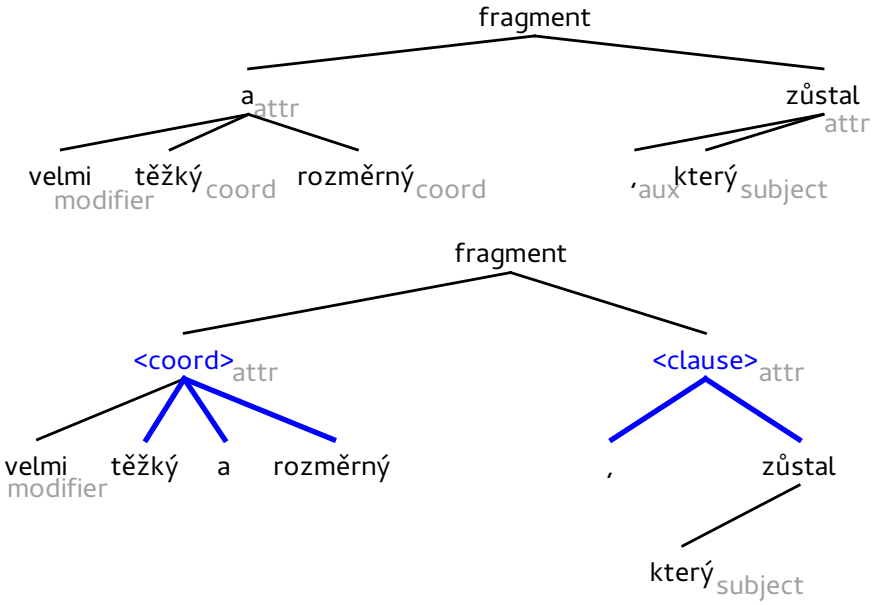


Figure 5.1: Comparison of dependency and hybrid tree on sentence part *"velmi těžký a rozměrný fragment, který zůstal"* (*"very heavy and (very) bulky fragment, that stayed"*).

5.3 Analysis by pattern matching

We have decided to use the pattern matching approach to describe the rules according to which the system analyses the input sentences. Pattern matching belongs to techniques that are widely used in natural language processing. Its main advantage is transparency and simplicity which is in agreement with our general values. Applications based on pattern matching are usually fast, understandable for people and suitable for further extensions.

In context of syntax processing, pattern matching is often used in partial parsing and morphological disambiguation [1, 15]. Word sketches, a successful commercial application already described in Chapter 4, are based on the pattern matching principle as well.

We have adapted the notion of a pattern to describe probabilistic linking rules that are able to cope with the unrestricted word order of the Czech language. Let us have a finite number of known syntactic constructions –

patterns. Then, we consider the process of analysis as searching patterns in the input text, sorting them according to their estimated probability, and finally selecting the most probable non-conflicting ones, to build a syntactic tree.

Our patterns are similar to CQL queries in the sketch grammar formalism, as introduced in Chapter 4: each pattern is a sequence of word class restrictions that describes a syntactic phenomenon. The simplest example is an adjective followed by a noun, in Czech with agreement in case, number and gender, forming a noun phrase (or alternatively, determining a dependency $adjective \rightarrow noun$). The patterns can contain gaps for tokens that are not relevant for the particular phenomenon, e.g. the symbolic pattern

$$preposition \ \ldots \ noun$$

(in Czech with case-agreement) for describing a basic prepositional phrase has a gap for any number of adjectives, determiners etc. More complicated patterns can also be used, e.g.

$$verb \ \ldots \ comma \ \ subord\_conjunction \ \ldots \ verb$$

for description of a subordinate clause.

5.3.1 Parsing algorithm

The realisation of a pattern in the input sentence (the particular words covered by the pattern) is then called a *match*. In the parsing process, only a limited number of matches is selected to obtain complete and unambiguous sentence analysis.[2] This selection is driven by a weight assigned manually to each pattern, and length of the resulting match, e.g. the distance between governing and dependent word, or between members of the coordination, according to the simple formula

$$selection\_weight = \frac{pattern\_weight}{match\_length}$$

We have tested enriching this simple weighting by considering match position in the input sentence, and values of external collocation statistics for words in the particular matches. These two have been implemented and

2. Ambiguous sentences could be described with more options of how the matches are selected; currently one of the output formats is displaying all found matches with their ranks, so the tool offers also a type of ambiguous analysis.

can be used, however, we did not spot any significant improvements by any tests (rather the opposite), so we returned to the simplest and most transparent weighting.

Given the set of patterns, the parsing algorithm consists in identifying the patterns in the input and selecting the best matches. Searching the matches in the input text can be compared with regular expression matching.

There is a theoretical objection to applying regular language machinery to natural languages, namely that the natural language structure is at least context-free, maybe even context-sensitive [161]. However, we do not care about this too much, because:

- We do not address the decision problem – if the particular sentence is grammatical or not – but just try to find most probable analysis provided the sentence is correct. The parser assumes every input sentence to be grammatical.
- The available proofs regarding non-regularity or non-context-freeness of natural languages always consider a nesting of some type, to an arbitrary depth. However, human brain has no infinite stack which is necessary for understanding arbitrary nestings. Corpus sentences provide evidence that a common level of nesting is 1 or 2 and the limit lies somewhere around 5 or 7 (may relate to psycholinguistic findings [114]). Apart from that, as we will see, our implementation of the parsing algorithm allows arbitrary nestings of some types.
- As we have shown on the sketch grammar example, there are successful approximative approaches to language analysis that work sufficiently, and are successful thanks to their simplicity rather than complete description of the language. This is the way of development we want to follow.

For the usage in the parsing algorithm, the patterns are divided into groups (or layers) according to the nature of the described phenomena; all layers are matched simultaneously and compete with each other, but the parser handles each of them in a slightly different way. Currently, SET contains 3 pattern layers:

- **relative clauses** – this layer contains rules identifying phrasal nodes for clauses, insertions and apposition attributes – `<clause>`, `<inter>` and `<attr>`. When such a node is identified, the matching process is ran once again (matches with higher priority from the previ-

ous run are kept), but words under the identified phrase are made invisible for that run. This behaviour enables nesting of relative clauses, and correct analysis of long distance links in case of insertions of one clause into another which is quite common in Czech. For example, in case of sentence "A man, who laughed, died" it would not be possible to capture the dependency $man \rightarrow died$, as it would be always beaten by $man \rightarrow laughed$. When the clause ", who died," is identified, its words are skipped in the next round of matching and the dependency $man \rightarrow died$ can be discovered.

- **coordinations** – this layer identifies the `<coord>` phrases. Its special evaluation is, if two coordinations are identified next to each other, they are joined to one bigger coordination. This enables analysis of coordinations with any number of members, with only binary rules, which simplifies the pattern matching grammar significantly.
- **dependencies** – all the other rules, mostly for finding dependencies, as the name suggests, but can contain also phrasal rules on lower levels than coordinations, such as multiword prepositions or named entities.[3] There is no special behaviour on this level, except one detail: The parser allows only one dependent node for each preposition (recognized by the "prep" alias, as described below), as it is not possible for a preposition to have more than one object, and it is not possible to express this restriction within the grammar itself.

5.3.2 Rule syntax

SET includes a readable and expressive definition format for the patterns to be transparent and easily editable. It is similar to sketch grammar CQL queries but developed independently and more oriented on syntactic analysis, rather than querying a corpus. In this section, we describe the main aspects of the rule syntax; a full and up-to-date reference is available on the project page.[4]

Each pattern definition consists of two parts, a template and a set of actions. If a template is matched during parsing, the corresponding actions are executed.

The **template**, introduced by "TMPL:" keyword, defines the pattern: a sequence of word classes involved, including gaps. A word class is de-

3. SET can be used for recognizing named entities. However, currently there is no comprehensive rule set for named entities available and their analysis relies on external tools.
4. `nlp.fi.muni.cz/projects/set`

scribed with a word form, lemma and/or a morphological information, matched against a tag from the preceding morphological analysis. Using labelled predefined restrictions, e.g. "infinitive" or "noun", is also possible.

The **actions** instruct the parser how to interpret the particular match, e.g. they can contain instructions for marking a dependency relation between two words. They can also specify the weight of the pattern and additional match restrictions such as morphological agreements.

In the following example:

```
TMPL: numeral ... noun
MARK 0   DEP 2
```

the template on the first line specifies a numeral-noun pair, probably within a single noun phrase, such as "three dogs" or "three beautiful dogs". In the action part on the second line, the parser is instructed to mark a $numeral \rightarrow noun$ dependency, in case the pattern is used (MARK identifies the dependent word, DEP the governing word – see below for precise description).

There are several possibilities of expressing restrictions within a template. The basic one is a **single condition**, a pair of input attribute (word, lemma or tag) and description of its value, enclosed in round brackets, for example:

- (lemma world) – will match input words with lemma *world*.
- (word and|or|so) – will match input words *and*, *or* or *so*.
- (tag k[123].*c2) – will match any substantive, adjective or pronoun in genitive (using the Ajka tagset [159], see also below).

As can be understood from the examples, disjunction (logical or) of values is allowed, denoted by a vertical bar (|), and in case of the tag attribute, it is possible to use regular expressions.

A **named variable** enables reusing a condition in multiple templates, and express a restriction on more input attributes at the same time. It is expressed by dollar sign followed by a variable name that can contain alphanumeric symbols, dots or underscores. If such a variable is found, its definition is looked up below the template where it was used. The definition uses the following format:

$NAME(attribute): list of values (constraints) separated by whitespace

Only the first following definition applies. More constraints within the definition line are interpreted as their discunction (logical or). If more definition lines immediately following each other are found, it is interpreted as a conjunction (logical and). Negative constraints can be created using the "not" keyword. The following example:

```
$CONJ(word): k8.*xC k8.*xS
$CONJ(word not): and
```

defines a named variable `$CONJ` which would match any word with morphological tag matching one of the two regular expressions on the first line, but not if the word is "and".

A more complicated variant of named variables can be also used for interconnecting restrictions. Imagine you want create a rule for matching a coordination consisting either of two nouns or two adjectives (but not noun and adjective). This can be achieved by using the following syntax with the `MATCH` keyword:

```
$C1 (word and) $C2
...
MATCH $C1(tag) $C2(tag)
k1 k1
k2 k2
END
```

This construction defines both `$C1` and `$C2` at the same time and specifies that either both of them are nouns (tag "k1") or both of them are adjectives (tag "k2"), otherwise the template does not match.

Last option of expressing a restriction is a **global alias** (different from local named variable) that we have already used in the illustrative examples at the beginning of this section. It is defined by the `CLASS` keyword, e.g.

```
CLASS prep (tag k7)
```

allows using string "prep" instead of "(tag k7)" throughout the grammar. Some of these definitions have also impact on the parsing process, e.g. the preposition definition above determines which tokens should be allowed

to be the target of only one dependency, as explained in the end of Section 5.3.1.

Also, two special restrictions similar to aliases were implemented – we named them "bound" and "rbound" – that can match the beginning and the end of the sentence respectively, as well as punctuation, or a specified conjuction.

We have already shown that the rule format allows gaps matching any number of any words, marked as three dots. It is also possible to restrict the content of the gaps. A named variable whose name ends with an asterisk is interpreted as any number of words matching the definition of the variable. For example,

```
$ADJS*(tag): k2
```

defines a variable `$ADJS*` matching any number (including zero) of adjectives.

Actions follow the template specification in the rule. An action is a function taking one or more arguments that often refer to restrictions in the template, by indices starting with 0. The following actions can be used:

- `MARK` is used for marking one or more words. It is used in two cases:
  - Adding a phrasal node: in this case, the last argument must be the name of the new node. Example: "`MARK 0 2 4 <coord>`" creates a phrasal coordination node consisting of tokens with indices 0, 2 and 4.
  - Marking a single word, which is then assigned a dependency using the `DEP` action (see below).
- `AGREE` is used for testing grammatical agreement. It takes 3 arguments, the first two are the token indices, the third describes the morphological categories to be tested. Example: "AGREE 0 2 cng" tests a case-number-gender agreement on tokens with indices 0 and 2, using attributive morphological tagging used by the morphological analyzer Ajka [159], where lowercase letters encode morphological attributes (c = case, g = gender, n = number) and uppercase letters or numbers denote their values, e.g. P stands for plural, if preceded by n. The action requires the values of the given attributes to be the same. Similar agreement restrictions can be implemented for other types of morphological tags.

- `DEP` determines the dependency for the result of the `MARK` action. It takes a single argument describing the token index that the `MARK` word (or a newly created phrasal node) will depend on.
- `HEAD` marks the head of a phrasal node created by the `MARK` action. It takes a single argument – the token index to be marked as the head. This is important namely by converting hybrid trees into dependency ones.
- `LABEL` assigns a syntactic label of the word. Its argument is any string describing a syntactic function, e.g. *subject*, *object*.
- `PROB` assigns a weight (although named according to probability, it does not need to meet the probability axioms) to the pattern. It takes a single argument – the weight which is any positive number. If the action is not used, the weight defaults to 100. The `PROB` keyword can be also used within the `MATCH` block, with the same effect – preferring some of the pairs to the others. If we modify the example above:

```
$C1 (word and) $C2
...
MATCH $C1(tag) $C2(tag)
k1 k1
k2 k2  PROB 50
END
```

  the rule will prefer nouns in coordination to adjectives. Weight of adjectives is twice as low as the default weight of nouns, so in case of a conflict, the rule would select adjective coordination only in case it is at least twice as short as the conflicting noun coordination.

Two more illustrative examples follow:

```
TMPL: verb  ...  $AND  ...  verb
      MARK 0 2 4 <coord>  PROB 500  HEAD 2
$AND(word): , a ani nebo
```

The template in this example describes a coordination of two verbs using one of the Czech conjunctions *a*, *ani*, *nebo* (and, neither, or), or a comma.

The actions say that the relevant tokens should be marked as a coordination with the conjunction being the head of this constituent. Also, the weight of this pattern is increased from default 100 to 500.

The last example:

```
TMPL: noun  $...*  comma  (tag k3.*yR)  $...*  verb
      $...*  rbound
      MARK 2 5 7 <clause>    DEP 0    AGREE 0 3 gn
$...*(tag not): k3.*yR
```

can be used for matching relative clauses. The template matches a sequence consisting of a noun followed (possibly) by a gap (which cannot contain a relative pronoun, as specified on the last line), a comma, a relative pronoun, again a possible (restriced) gap, a verb followed by another gap and a special rbound restriction matching the end of the sentence, one of the selected conjunctions, or a punctuation. Such a relative clause (its border words and the main word – the verb) is enclosed under a `<clause>` phrasal node which is made dependent on the preceding noun, if the agreement in gender and number between the noun and the relative pronoun is confirmed.

5.4 Usage

The system expects a vertical file as input (one word per line, tab-delimited columns for word, lemma and morphological tag). The preloaded grammar for Czech uses the Ajka tagset [159], however, the positional tagset used e.g. in the Prague Dependency Treebank, can be also used with this grammar – the conversion was implemented within the tool and can be activated by a command line switch. The preloaded English grammar uses the Penn Treebank tagset [103]. Ambiguous comma-delimited lemmas and tags are allowed in the input. Vertical files containing more sentences can also be processed: in this case, the "`<s>`" mark on a separate line is to be used as sentence delimiter. The tool processes the input as a stream, so analysis of big files (e.g. corpora) is possible, provided they contain sentence delimiters.

Apart from the two default grammars for Czech and English, a grammar for Slovak has been developed [111], and grammar dedicated for detection of verb phrases [8], as defined in the bushbank project. The latter achieved best results in verb phrase detection, against the bushbank data, as reported in Section 3.4. Both of them were primarily created by bachelor students

which illustrates that the grammar format is comprehensible and easy to use, even for non-technical people (the author of the verb phrase grammar was a student of Czech at a faculty of arts).

The parser can be run right after download from a command line:

```
./set.py   [OPTIONS]   file
```

If file is not given, the standard input is read. Important options include:

- `-g` – graphical output (Python Qt4 bindings are required)
- `-d` – output in form of dependency trees
- `-p` – output in form of phrasal trees
- `-b` – output bush
- `-v` – verbose: output analysis details (all found matches with their weights)
- `--cout` – output in form of collocations
- `--phrases` – output all found phrases (includes all nested phrases)
- `--short-phrases` – output selected phrases, as needed for bush-bank annotation
- `--long-phrases` – the same as above, but do not split the phrases on prepositions (trust the parser can resolve PP-attachment well)
- `--sconll` – output dependency trees in simplified CONLL format[5]
- `--postags` – use Czech positional tags on input with the default grammar
- `--maxlen=<num>` – parse only sentences shorter than `<num>` words (helps efficiency; recommended limit is 100)
- `--grammar=<path>` – use an alternative grammar

Output is always provided in the text form; if `-g` was used, a graphical window is displayed in addition. For all types of syntactic trees, we use the following textual coding (it is more or less a guideline for drawing the tree, for the graphical environment):

- ID of the node (integer number)
- name of the node (e.g. word form or `<coord>`)
- ID of the governor node (integer number)
- dependency type – “p” or “d”, for phrasal (blue) or dependency (black) edge

5. `nextens.uvt.nl/depparse-wiki/DataFormat`

- dependency label of the node (optional)

A web interface to the tool named wwwSET (linked from the project page)[6] has been developed which gives the most important types of results together on one page. As input, it allows a plain text sentence (in that case, the sentence is tagged automatically using the Desamb tagger [171]) or a file in the vertical format. It is also possible to upload an alternative grammar that will be used instead of the default one.

5.5 Conclusions

We have introduced SET, light-weight parsing system with a pattern matching grammar and rich output possibilities. Its development was based on the principles declared in Section 2.5.5, namely design simplicity and usability were the main values. During the development, the evaluation of the grammar against treebank data was abandoned, and a strong emphasis on usage in applications was introduced.

This approach has already its first visible results: although still in development, and although it is significantly worse than the best statistical parsers according to testing against treebank data, the analyser has been used in more collaborative projects than is usual for this type of tool [126, 125, 4, 3, 132, 152]. We describe some of the particular applications in the next chapter.

6. `nlp.fi.muni.cz/projects/set`

Chapter 6

Applications

In this last chapter, we discuss how the automatic syntactic analysis can be accommodated to serve various real-world NLP applications immediately, neither as a base for further, theoretical, semantic processing, nor in the far future when it will (maybe) be perfect. In most cases we refer to the sketch grammars and to the SET system, as described in the previous chapters, and we show prototypes of the real-world applications with syntactic analysis as part of the process. We will see that very important part of the accommodation is that the syntactic analysis tool is flexible, easily adjustable and simple which disqualifies the statistical tools to significant extent, at least in the initial phases of the research.

6.1 Information extraction for Czech

Unlike Miyao [118] whom we referred to earlier, we aim at the "extract all we can" version of the information extraction task: given an input text, convert as much information as we can to a formal database format. The following text describes a joint work with Vít Baisa, within the scope of a applied research project for the Czech ministry of interior. Author's contribution to this work is ca. 70 percent.

We have implemented an information extraction system for Czech – EFA (abbreviation form "extraction of facts") based on pipeline processing of the input plain text: the process involves tokenization, sentence boundary detection, morphological analysis and disambiguation, syntactic analysis, semantic noun and prepositional phrase classification, and output representation. We have discussed the processes below the level of syntactic analysis in Section 2.1, so we will skip their description within this section.

We use the bush output of the SET parser described in Chapter 5. This type of output contains noun, prepositional, verb phrases and clauses, as illustrated in Figure 6.1, together with dependencies among them. We further classify these phrases semantically, according to what type of infor-

```
<s>Myslím , že skinheadi v noci do města nechodí .
<clause> Myslím
<vp> Myslím
<clause> skinheadi v noci do města nechodí
<clauseconj> (k8xS): že
<vp>  nechodí
<phr> skinheadi
<phr> v noci
<phr> do města
</s>
```

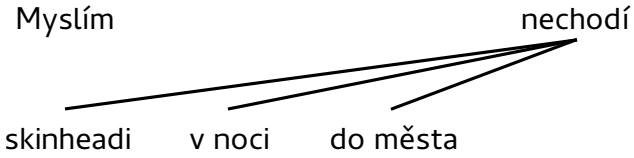


Figure 6.1: Simplified SET output in the bush format, for sentence *"Myslím, že skinheadi v noci do města nechodí."* (*"(I) think that skinheads do not go in the city in the night"*).

mation they contain. We have designed a set of semantic labels, that are mostly answers to wh-questions, together with an algorithm for mapping the syntactic phrases to these semantic labels. The result is then turned into a relational database record, as illustrated in Figure 6.2. The list of semantic labels follows:

- **přísudek** (predicate)
- **podmět**, kdo/co (subject – who, what – nominative)
- **komu/čemu** (whom, what – dative)
- **koho/co** (whom, what – accusative)
- **instrument**, kým/čím (by whom, by what)
- **kde** (where)
- **kam** (where to)
- **odkud** (where from)
- **kdy** (when)
- **od kdy** (when from)
- **do kdy** (when to)
- **důvod, proč** (why)
- **způsob, jak** (manner – how)

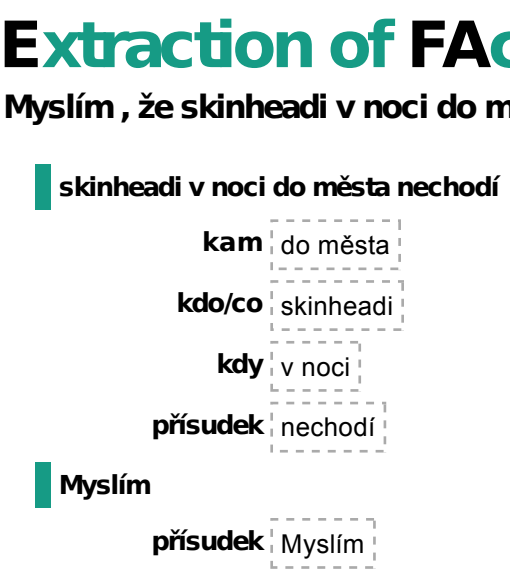


Figure 6.2: Result of Extraction of facts for sentence *"Myslím, že skinheadi v noci do města nechodí."* (*"(I) think that skinheads do not go in the city in the night"*).

accuracy of phrase detection (F-measure)	87.7 %
accuracy of phrase classification	79.7 %
overal accuracy	**69.9 %**

Table 6.1: Results of EFA initial evaluation.

The mapping algorithm is straightforward in case of noun phrases – in Czech, the wh-questions mostly correspond to the case of the phrase. In case of prepositional phrases, this correspondence is more complicated; almost all of the preposition-case pairs can correspond to more semantic classes. We have used lexical semantic information, namely a list of specific hypernyms, from the Czech WordNet [138] for approximation of the phrase semantics, thanks to which we are able to distinguish e.g. locations ("in the city") from time expressions ("in the next hour") [3].

Initial evaluation on 50 randomly selected sentences from news group texts showed that the overall accuracy of the system is around 70 percent, see Table 6.1. This is probably not enough for precise gathering information from texts e.g. for automatic reasoning. However, it is enough for highlighting certain types of information (e.g. by different colours, as illustrated

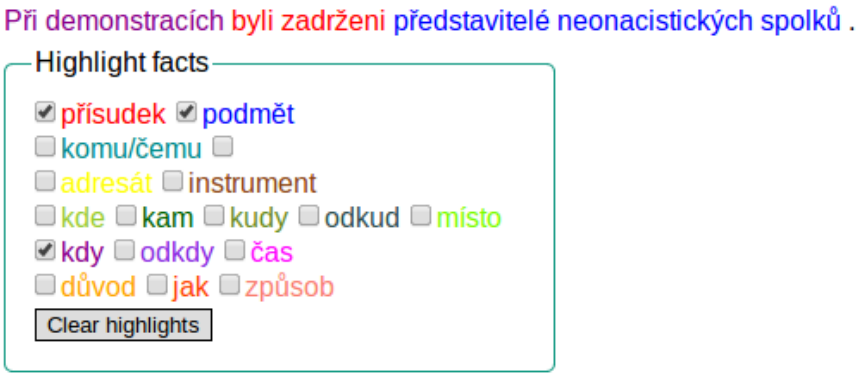


Figure 6.3: Illustration of EFA highlighting mode.

in Figure 6.3) which will help people scan texts quickly rather than read it, and focus only on interesting information, e.g. when looking for potentially dangerous posts on internet forums. This was the original purpose of the development, and the final application is currently running in testing mode.

We have also developed a web interface of the application for demonstration purposes[1] which takes a plain text or a web page URL as input, and returns the extracted information. The result can be displayed in three formats: visually structured tables (as in Figure 6.2), XML, or coloured text, as illustrated in Figure 6.3.

Future work, besides increasing precision of all components in the pipeline, includes applying the same type of processing in the information retrieval task and question answering: a query (including its wh-words) will be analysed using the introduced semantic labels, and compared with the documents within the database, analysed in the same way. This approach should solve some of the problems outlined in the introduction. Both of these applications are currently in development, but so far without salient results.

Also, for technical reasons, we want to integrate the phrase classification into the syntactic analysis process, as labels within the SET grammar, which will result into more "semantic" syntactic analysis. We believe this will lead to better maintainability and flexibility of the whole pipeline, which will enable faster enhancements in the future.

1. `nlp.fi.muni.cz/projekty/set/efa/wwwefa.cgi/first_page`

```
<inference type="effect" verb="dochutit"
                                        mean="to_flavour">
<ruleset id="taste_like" inf_verb="chutnat"
                                         negation="False">
  <rule case="c4" prep="" inf_case="c1" inf_prep=""/>
  <rule case="c7" prep="" inf_case="c6" inf_prep="po"/>
</ruleset>
</inference>
```

Figure 6.4: Inference rule saying that to flavour X (expressed by accusative = c4) with Y (instrumental = c7) has an effect that X (nominative – c1) tastes like Y (expressed by preposition "po" and locative = c6). Source: [126].

6.2 Automatic reasoning for Czech

Automatic reasoning, or textual entailment for Czech is being explored by Zuzana Nevěřilová [126]. Her prototype system uses the bush output of the SET parser for identifying the structure of the sentence, so the syntactic level of this project is relevant to the topic of this thesis.

The textual entailment project has introduced a concept of transformation rules, describing algorithms for generating new valid sentences (also called hypotheses), provided the input sentence is valid. The rules are written in XML, as illustrated in Figure 6.4, and work over the bush syntactic representation of the sentence. They can be classified into 4 groups [126]:

- **effect** – after performing the activity described in sentence, the hypothesis holds
- **precondition** – before performing the activity described in sentence, the hypothesis holds
- **near synonymy** – the propositions are judged as equivalent
- **conversion between active and passive verb forms**

The original set of rules is hand-written, however, it can be bootstrapped automatically, e.g. using verb synonyms, as reported in [126].

An evaluation of two sets of the transformation rules was performed, with very good results, as summarized in Table 6.2. As source data, 2400 sentences from the domain of cooking recipes were used. Firstly, only the set of 175 manually created rules was used, which was able to generate 1783 hypotheses, out of which 1533 was correct. Then, the group was automatically extended using synonym information from the Verbalex lexicon [44],

Rule set	# sentences generated	# syntactically correct	# semantically correct
175 manual rules	1783	1633 (91.6%)	1533 (86.0%)
599 (276) rules	2826	2598 (91.9%)	2443 (86.4%)

Table 6.2: Results of evaluation of the transformation rules.

to 599, out of which 276 were used (matched in the input data). From the same group of sentences, this set of rules generated 2826 hypotheses, out of which 2443 were correct. Both of these results indicate 86 percent precision; recall cannot be well estimated, as it is probably impossible to list all the correct hypotheses following from a sentence.

We consider this result very important in context of information extraction and question answering, where it can help much to address the problem of data sparseness.

A rigorous evaluation set was also built which presents a different approach to evaluation of the textual entailment [124]. It contains pairs (text, hypothesis) from reading comprehension tests for children and adults of various ages, together with indication of correctness of the hypotheses. This data set will allow evaluation of both precision and recall of the system using a practical task, in a way which is different from the precision evaluation described above. However, the current reasoning system has not been evaluated against this data yet.

6.3 Authorship recognition of Czech texts

Within the project for the Czech ministry of interior, that we have already mentioned, a tool for authorship recognition and verification is being developed and tested [151]. The tool is designed for both authorship attribution (assigning an author from a given group of authors, to a particular text) and authorship verification (deciding if the two texts are written by the same author, without knowing the set of authors in advance). The work is directed at an automatic or semi-automatic forensic linguistic tool, such as the ALIAS software[2] for English [21].

This section describes a collaborative work with Jan Rygl and Kristýna Zemková. Author's contribution to this work, besides parser development, is ca. 40 percent.

2. aliastechnology.com

6.3.1 Current approach

Algorithms for both authorship attribution and authorship verification are based on machine learning, with the following attributes, also referred to as stylometric features [151]:

- **relative frequency of important morphological tags** – 64 selected morphological tags have been used
- **relative frequency of important parts of speech (PoS)** – considered separately at the beginning, at the end, and in the middle of sentence
- **relative frequency of important PoS bigrams** – considered in the same way as single parts of speech
- **average sentence length** – expressed in number of alphanumeric words
- **relative frequency of important punctuation characters** like commas, semicolons, colons, question marks...
- **relative frequency of stoplist words** – as stoplist we considered 75 most frequent Czech words
- **relative volume of repeated words** considered separately for different parts of speech

These text attributes yield a high number of features (e.g. the frequency of stoplist words gains 75 features, one for each word) that are later used as input for machine learning. Vector consisting of these features, derived from available texts of a particular author, is then referred to as the author's stylome (analogy to genome in biology).

Two different approaches have been tested within the project. One-layer machine learning which compares the two stylomes using the support vector machine (SVM) method [167], simpler and faster in terms of both training phase and actual processing, compared to the second one. The second approach groups the stylometric features into several sub-vectors, or substylomes, on which similarities are computed. Then, another layer of machine learning is used, learning on the computed similarities [150]. This approach is called two-layer machine learning.

6.3.2 Syntactic features

We have performed extensive experiments with including features based on syntactic analysis in the authorship verification machine learning pipeline. Some similar experiments have been earlier done for English [49] with

unclear interpretation; we have implemented a different approach, regarding the syntactic features. We have designed two sets of features based on syntactic analysis within the SET parser, the first (further referred to as *syntax1*) being:

- **average maximum dependency tree depth** – longest path from the root to a leaf in the dependency tree gained from the `-d` output of the SET parser
- **average maximum dependency tree branching factor** – maximum number of children of any node in the dependency tree
- **relative frequency of dependency tree labels** – 12 dependency labels available in the SET dependency output were used
- **relative frequency of phrasal tree non-terminals** – 19 non-terminals available in the SET `-p` output were used

We believe these features may contribute to the author's stylome, as e.g. maximum dependency tree depth or its maximum branching factor capture the abstract way which authors use to building their sentences, to a certain extent, and dependency labels or phrasal non-terminals can tell us about frequency of certain types of phrases. The results of initial experiments with this set of features is described in [152].

Later, another set of syntactic features has been added to the first one (the combination is further referred to as *syntax2*):

- **grammar patterns within phrasal tree** implemented as sequences of non-terminals from SET `-p` output
- **relative frequency of syntactic bigrams of morphological tags** – syntactic bigram is a pair of words connected by a dependency; only the part of speech and case were considered among all morphological features, to reduce the option space

Clearly, these two new features can describe elements of an author's style as well – patterns can e.g. capture author's common clause constructions, bigrams can record frequent common bindings that the author uses.

6.3.3 Evaluation

The evaluation data were blog texts from 2012 automatically downloaded from a public Czech blogging site[3] and classified for authorship, according to the html markup. 5368 documents were used, with average length

3. `blog.ihned.cz`

of 300 sentences (and with relatively high variance – length of individual documents ranges from 20 to 500 sentences).

Evaluation was driven by the standard methodology in the stylometry area [79, 49], except our testing set consisted of internet blog texts, rather than literary works, as our application aims at this text type. For both training and testing, pairs of documents were used with the same proportion of positive and negative examples. Authors from the training part of the data were always different from authors in the testing part. Accuracy defined as number of correctly judged pairs divided by number of all pairs was used as the evaluation score.

Two training and testing configurations were used:

- **small data** with 680 training and 216 testing pairs of documents, which enabled more efficient evaluation of all the combinations of features, for indication of which ones are the most promising candidates
- **big data** with 3656 training and 960 testing pairs of documents; this set was expected to provide more reliable results, however, the computational complexity allowed us to evaluate only the most perspective feature combinations

The results of the evaluation are summarized in Table 6.3. From all combination of features, only the best results are shown; however, during the experiment, all combinations of the feature groups introduced above were examined. The baseline is 50 percent because of equal proportion of positive and negative examples in the testing set.

In case of one-layer machine learning, the original 7 groups of features gained 60.7% precision; syntactic features themselves were more successful, even more than combination of syntactic features with the original 7 feature groups. Best result was achieved using a sub-group of the original set of features together with all the syntactic features.

In case of one-layer learning and big data, the combination of original features with the first group of syntactic features was the best. Sub-group of the original set of features together with one group of the syntactic features win also in case of two-layer machine learning.

The two-layer machine learning is mildly more successful and the small set of documents seem to be somehow "easier" than the big one. It is disturbing that there are accuracy fluctuations for all the used feature sets, for different evaluation sets and machine learning methods. It is not easy to say which one of them is the best, although the big data set is probably more representative than the small set.

features	one-layer small data	one-layer big data	two-layer small data	two-layer big data
original7	60.7	63.5	65.0	64.2
syntax1	56.9	58.5	65.5	60.8
syntax2	65.3	56.2	58.9	52.7
original7 + syntax1	60.2	63.9	65.9	63.2
original7 + syntax2	63.9	61.0	62.8	64.2
combination1	67.2	60.0	65.5	**66.4**
combination2	65.2	62.9	**66.4**	64.1
combination3	56.5	61.7	**69.9**	66.0
combination4	63.4	62.1	57.9	**66.8**

Table 6.3: Results of the authorship verification experiment. Within the table, we name the combination of morphological tags, PoS bigrams, sentence length, stoplist, repeating words and syntax2 as **combination1**, combination of morphological tags, PoS, punctuation, sentence length, repeating words and syntax1 as **combination2**. Combination of punctuation, repeating words and syntax1 is denoted as **combination3**, and combination of morphological tags, punctuation, sentence length, repeating words and syntax2 as **combination4**.

However, the proposed syntactic features are contained in all the winning combinations, which can be interpreted as a clear benefit of the syntactic information provided by the SET parser to the authorship recognition problem.

6.4 Grammar checking for Czech

In this section, we introduce two case studies of partial grammar checking for Czech: new methods for punctuation correction, and detection of subject-predicate agreement errors in Czech. Both of the studies exploit the SET parser significantly.

6.4.1 Related work

There are two commercial systems for grammar checking of Czech: The Grammar checker built into the Microsoft Office, developed by the Institute of the Czech language [134], and the Grammaticon checker created by the Lingea company [165]. Not much has been published about the principles

these are based on; most of the available materials are Czech-only and have rather advertising character. According to available information, both of the tools are trying to describe negative (wrong) constructions and minimize number of false alerts, i.e. prefer precision over recall significantly (frequent false alerts bother users and make them stop using the tool). The available tests of these tools [136, 6] (available only in Czech) indicate that the tools are able to fix 25–35 percent of errors, with the number of false alerts 6–30 percent.

The Czech parsing community also contributed to the grammar checking problem. Holan et al. [46] proposed using automatic dependency parsing, however, authors conclude that the results have only a prototype character and much work is still needed to achieve practically usable product. Jakubíček and Horák [58] reported on using the Synt parser [51], together with a specialized grammar for Czech to detect punctuation in sentences. They report over 80 percent precision and recall in punctuation detection which means that the system fills in the commas into the text without commas (rather than into a text with errors). 80 percent in detection roughly means that every 5th comma is missing and every 5th is wrong. It is not completely clear how the system would behave on real erroneous texts and it is not possible to re-test, as the tool is not available at the moment.

6.4.2 Punctuation detection with the SET system

Within the SET parser, we have built a specialized grammar for punctuation detection, together with an added special output function which prints a comma before each word marked by a special phrasal token (we used <c>, as illustrated in the examples). The grammar contains 10 rules for analysing the most important patterns where a missing punctuation should be added, that are used for building a reduced tree where the only important information are the tokens marked with <c>.

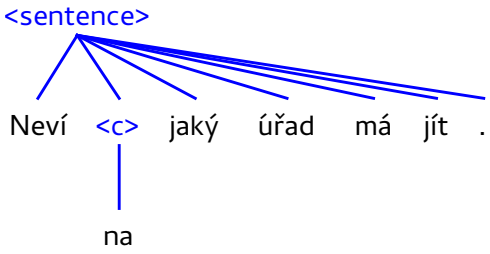
This approach is (again) deliberately approximative, and follows the more straightforward pattern matching idea of Grammaticon and Grammar checker, rather than the full syntactic analysis introduced by Jakubíček and Horák [58]. However, it is one of our future goals to combine the added functionality with the full power of the standard SET grammar and compare the results with the shallow approach.

Examples of a punctuation rule, a reduced syntactic tree for a sentence with missing punctuation, and the resulting sentence with completed punctuation, are given in Figures 6.5 and 6.6. As we can see, the "syntactic tree" on the SET output contains practically no syntactic information, except the

```
TMPL: $NEG $PREP $REL  MARK 1 <c>  HEAD 1
    $REL(tag): k3.*y[RQ] k6.*y[RQ]
    $PREP(tag): k7
    $NEG(tag not): k7 k3.*y[RQ] k6.*y[RQ] k8
    $NEG(word not): a * " tak přitom
```

Figure 6.5: One of the punctuation detection rules in SET, matching preposition and relative pronoun (k3.*y[RQ]) or adverb (k6.*y[RQ]), not preceded by preposition or conjunction or relative pronoun/adverb.

```
Input: Neví na jaký úřad má jít.
```




```
Output: Neví, na jaký úřad má jít.
```

Figure 6.6: Illustration of SET punctuation analysis – reduced tree and the output sentence with completed punctuation. The rule from Figure 6.5 was matched. Sentence: *"Neví na jaký úřad má jít."* (missing comma before *"na"* – *"(He) does not know what bureau to go in."*).

<c> guidelines for completing the sentence punctuation – rather than that, the SET parser is used as an economical pattern matching engine.

Evaluation of the functionality was performed using the DESAM corpus [137], using the same methodology as Jakubíček and Horák [58] – deleting all commas from the input sentences and comparing their original with the output of the parser.

The results are summarized in Table 6.4. We have distinguished a sample of first 500 sentences from the corpus, and the whole corpus of 50,000 sentences; also, we worked with both automatic and correct manual morphological tagging. We can see that the results are very similar for all the testing sets, and it can be concluded that errors in automatic tagging do not influence punctuation detection significantly.

The system shows very high, nearly 95 percent precision, which is very good as it minimizes the number of false alerts. Recall is rather low and means that the system is able to find only about 50 percent of errors. Speed of the analysis was in all cases rather high – 313 sentences per second, on a single Intel Xeon 2.66 GHz core.

We have performed a manual investigation of the differences between the parser output and the correct punctuation, on first 150 sentences of the testing data. This insight showed that many of the parser errors are actually not errors – in Czech, in some places the comma is not necessary but writing it is not a mistake. From the missed commas, 21.4 percent were not necessary according to the Czech writing rules (most frequent real errors were in coordinations). From the false positives, 50 percent were actually correctly placed commas. If we extrapolate these percentages to the whole Desam testing set, we get the numbers as in the Extrapolation row.

Testing set	Precision (%)	Recall (%)	F-measure (%)
Desam 500	94.7	47.3	63.1
Desam full	94.1	45.0	60.9
Desam 500 tagged	95.3	45.4	61.5
Extrapolation	97.1	56.8	71.6

Table 6.4: Results of punctuation detection within the SET system.

Our system outperforms the general reported results for Grammaticon and Czech grammar checker, in terms of both precision and recall – number of false alerts below 3% is very good compared to them, and also the recall is slightly better. Jakubíček and Horák [58] reported better recall but lower precision. We are confident that the precision is more important here, due to the bothering character of false alerts, and any tool with precision lower than 90–95 percent is not suitable for practical usage. Thanks to its results, our tool is ready to be built into a grammar checking application.

6.4.3 Subject-predicate agreement with the SET system

Unlike the previous case study, detecting errors in subject-predicate agreement in Czech sentences uses the full standard SET grammar. The rules detecting subjects of clauses (labelling them as "subject" and adding their dependency on the verb) were differentiated to correct subjects that agree with the detected verb in gender and number, and the salient candidates for subject that do not fulfill the agreement condition. The latter ones were

```
TMPL: $MAINVERB $...* $LIKESUBJ   AGREE 0 2 gn
         MARK 2  DEP 0  PROB 602  LABEL subject
TMPL: $MAINVERB $...* $LIKESUBJ
         MARK 2  DEP 0  PROB 601  LABEL subject-bad

MATCH $MAINVERB(tag) $MAINVERB(tag)
    k5.*mF  k5.*mF  PROB 110
...

MATCH $LIKESUBJ(tag) $LIKESUBJ(tag)
    k1.*c1  k1.*c1
    k3.*c1.*xP  k3.*c1.*xP
...
```

Figure 6.7: One of the SET subject rules, and its twin detecting bad subject-predicate agreement.

labelled as "subject-bad", for marking the difference. Example of the SET rules is given in Figure 6.7, and the output trees are illustrated in Figure 6.8.

Again, the current rules within the SET grammar cover the most frequent patterns. There are more complicated cases where the subject consists of a complex coordination, error in which would not be detected by our solution, in certain cases. However, according to the YAGNI principle introduced in Section 2.5.5, we first implement and test the straightforward approach, then identify the real drawbacks and then plan how to fix them, rather than devising a complete solution at the beginning and suppose that

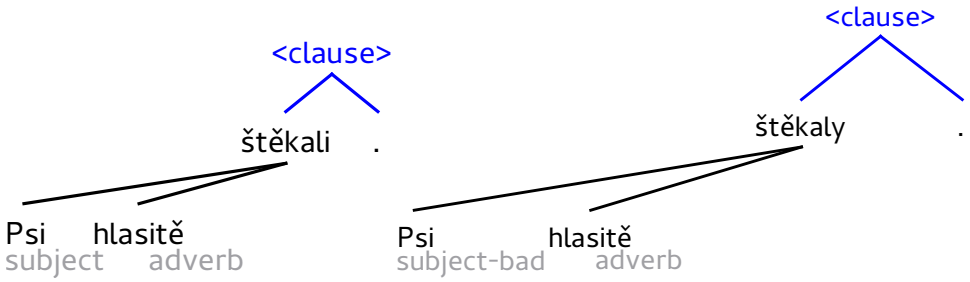


Figure 6.8: SET output tree for correct and incorrect version of sentence *"Psi hlasitě štěkali." ("Dogs loudly barked.")*.

we are able to anticipate possible problems. Correctness of the YAGNI principle showed very early in this case.

As there is no large available database of frequent Czech subject-predicate agreement errors, we have decided to use a small set of sentences from a Czech primary school dictation, where frequent errors were manually identified and classified [170]. The set contained 26 sentences with 11 subject-predicate errors. Although the testing set is small, from Table 6.5 we can clearly see that there is a problem in automatic morphological tagging. The difference in recall between the manual and automatic version is immense, and the reason is that the subjects in the erroneous clauses were tagged as non-subjects, e.g. as accusative instead of nominative (there is very frequent nominative-accusative homonymy in Czech), and therefore they were not recognized as subjects by the parser. This is probably caused by the fact that the tagger (Desamb [171]), as it is usual for taggers, was trained on correct texts and the non-agreement between subject and predicate is so rare in these texts, that it rather chooses another option. Actually, most of the tagging errors resulted in syntactically correct Czech sentences, sometimes even semantically correct, although not suitable in the given context. This is a complex problem that will require a new approach to Czech tagging.

# sentences	26
# errors	11
# errors spotted (automatic tagging)	2 (18%)
# false alerts	0
# errors spotted after tagging correction	7 (64%)

Table 6.5: Results of punctuation detection within the SET system.

Another problem are sentences with unvoiced subject (usually present in the previous sentence) – this was in 3 of the 11 sentences. Solution to this problem requires quality anaphora resolution, and we did not attempt to solve it within this case study.

Notable is the 100 percent precision that we have obtained in case of both manual and automatic tagging – there was no false alert. Thanks to this, although the recall is very low due to the automatic morphological tagging, the system can be immediately employed in a grammar checker as well.

6.5 Collocation extraction

Within the work on word sketches, we are constantly interested in monitoring their quality. Two evaluations of the word sketch functionality were performed, mainly with respect to their usability for a lexicographer or language learner. This section describes a collaborative work with Adam Kilgarriff, Miloš Jakubíček, Vít Baisa and Lucia Kocincová, with author's contribution ca. 50 percent.

6.5.1 Word sketch evaluation I

The first experiment took place in 2010, and word sketches for Dutch, English, Japanese and Slovene were evaluated [69]. A special web interface was developed for this purpose and 42 sample headwords were randomly selected from 3 different frequency ranges. Top 20 collocations for each of the sample headwords were judged by two different annotators for being good or bad, where the criterion for "good" was a positive answer to the question "would the collocation have been suitable for including at this entry in a dictionary like Oxford Collocation Dictionary?". All of the annotators were native speakers of the respective languages, and all of them were experts in lexicography with previous annotation experience, so they were model users of the word sketch application and very competent to perform the evaluation. Standard Sketch Engine corpora for the particular languages were used, and standard processing pipelines for tokenization and tagging [69].

The results of the evaluation are summarized in Table 6.6. We can see that the inter-annotator agreement was not too high; it is natural in these types of tasks – the actual feature that we wanted to evaluate can hardly be specified more precisely without significant bias. On the other hand, it is a practically defined task – collocation dictionaries do exist and are needed, so the evaluation is meaningful. For the calculation of the final percentages, we took only the cases into account where the annotators agreed with each other. The results indicate that across languages, around 70 percent of the top collocates are useful for recording in a dictionary, and therefore also relevant for language learners. The Japanese result was even better, and the details of Japanese evaluation were further discussed in [164].

By the method described above, we are able to assess precision but not recall – in other words, we are unable to find out how many relevant collocations were missed from the top word sketch results. We aimed at addressing this issue in a recent experiment, where we built gold standard

Language	Total #collocations	Agreed	Good	Bad	% good
Dutch	782	501	332	169	66.3
English	794	519	367	152	70.7
Japanese	747	690	600	90	87.0
Slovene	800	550	391	159	71.1

Table 6.6: Results of the word sketch precision evaluation from 2010.

data for evaluation of collocation extraction. This second experiment was performed for English and Czech.

6.5.2 Gold standard and word sketches from parsers

To build the gold standard data that would allow evaluation of both precision and recall, we needed to put together as many sources of collocations as possible. Again, sample headwords were randomly selected from 3 different frequency ranges, this time the sample size was 105. For each of the headwords, we have inspected all the available corpora for the language, processed with all available processing pipelines, to gather 500 best collocations (or 250 in case of medium frequency words, or 125 for low frequency words). For English, we completed this set with data from available dictionaries of English [29, 146, 7], some of them in their online versions,[4] and WordNet [115].

For Czech, we were unable to find similarly rich and reliable sources of collocations; to lower the risk of sparse data, we decided to add collocations found by available parsers. For this purpose, SET, Synt [51], MST Parser [110] and MaltParser [127] for Czech, both trained on the Prague Dependency Treebank data, were used for creating word sketches, instead of standard CQL sketch grammars. It has been also a significant opportunity to compare the parsers on a practically defined task, and to test if full parsing would help to gain better collocation lists in form of word sketches.

Then, this set of collocations was ordered randomly and evaluated manually by the same method as above, for collocation appropriateness for a dictionary. There were 3 annotators for each of the languages. Again, the inter-annotator agreement was not too high, with respect to the fact that the evaluation set was significantly unbalanced ("bad" was far more frequent

4. Oxford Dictionary of English at `oxforddictionaries.com`,
Collins English Dictionary at `collinsdictionary.com`,
and Merriam Webster at `merriam-webster.com`

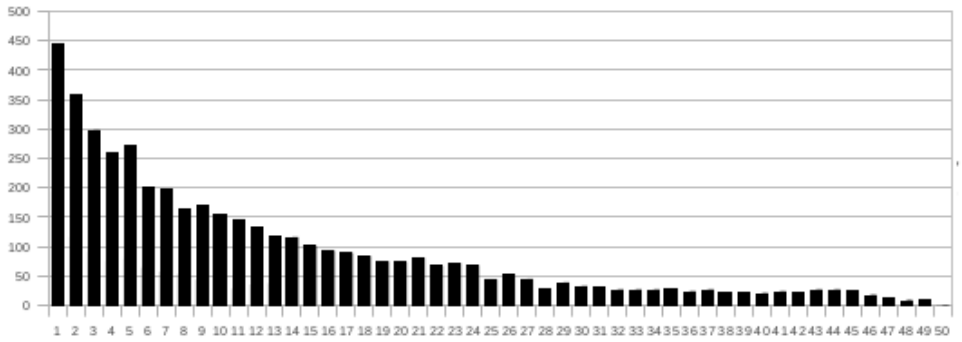


Figure 6.9: Distribution of collocations selected into the gold standard set, ordered by score.

answer) – between 73 and 90 percent pairwise, Cohen's kappa [26] below 0.5. However, we do not consider this a problem, as the task is to find a reference set of *good* collocations, as complete as possible, the purpose of which is to enable comparing corpora, processing tools and settings, rather than to serve as an ideal output of a software tool (such ideal is, of course, unreachable because of the limits in agreement among annotators).

We have selected the desired set by taking annotations where all the annotators but one answered "good". All the rest is bad (or grey zone) which does not have to be included in the experiments.

As for the completeness of the gold standard set, and its appropriateness for estimating recall: We expected that with the length of the collocation lists, with decreasing scores and frequencies of the collocations, less and less collocations will be annotated as good, and at the end of the 500–250–125 lists, there will be no good collocations found. That would indicate that if we evaluated longer lists, we would gain no more good collocations. (Of course, we would still miss the collocations that are neither present in the corpora, nor in the dictionaries… but then it is impossible for us to find them, and it is a philosophical question if such non-occurring collocations should be considered good at any circumstances – if they do not occur in texts, who needs to know about them?) This expectation has been confirmed to some extent, as illustrated in Figure 6.9 – at the end of the 500–250–125 lists, only a very little number of good collocations is found which indicates that we have probably missed some good collocations but their number is very small.

As result, gold standard data set for the collocation extraction task was created and made public, for both Czech and English, containing ca. 5,000

Corpus	Settings	# collocs	Prec (%)	Rec (%)	F-5 (%)
BNC	frq/hl/10	1,122	58.5	12.3	12.7
Model	frq/20/10	969	55.7	10.1	10.5
enTenTen12	frq/hl/10	1,456	52.7	14.4	14.8
enClueWeb09	frq/-/3	89,174	5.4	90.1	56.6
enTenTen12	frq/-/3	66,499	7.2	89.5	62.1
enTenTen08	frq/-/3	52,018	8.9	87.1	65.1
enTenTen12	frq/hi/5	23,113	17.8	77.3	68.5
enTenTen08	frq/hi/3	22,794	18.0	77.1	68.5
enClueWeb09	frq/hi/10	23,651	17.4	77.0	68.0

Table 6.7: Best results of English word sketch evaluation with gold standard: 3 best according to precision, 3 best according to recall and 3 best according to F-5. **Settings legend:** sorting / max. no. collocates per headword / minimum frequency; **frq:** sorting collocations by frequency, **hl:** max. no. collocates is 25/12/6, according to headword frequency band, **hi:** max. no. collocates is 400/200/100, according to headword frequency band.

good collocations for each language. This dataset is suitable for extrinsic evaluation of corpus data, as well as corpus processing tools such as tokenizers, taggers, parsers or sketch grammars.

6.5.3 Word sketch evaluation II – using the gold standard

We have tested a large number of corpora and settings such as minimum frequency, collocation scoring function and maximum number of extracted collocations per headword. Against the set of good collocations, we have observed precision, recall and F-5 score which counts with both precision and recall but discriminates precision in favour of recall. Recall is more important for this type of application where the automatic procedure prepares data to be further processed by people, either lexicographers, or language learners – omission of some desired results is worse than including non-relevant results (or results in the grey zone). F-5 score is defined as

$$\frac{(1+25) * precision * recall}{25 * precision + recall}$$

Best results in terms of precision, recall and F5 score for both Czech and English are summarized in Tables 6.7 and 6.8. According to the data, the

Corpus	Settings	# collocs	Prec (%)	Rec (%)	F-5 (%)
CZES MST	log/hl/10	1,480	38.8	11.8	12.2
CZES Malt	log/hl/10	1,478	38.0	11.6	11.9
CZES	log/hl/10	1,447	36.8	11.0	11.3
czTenTen12	frq/-/3	58,462	7.2	86.7	60.9
SYN	frq/-/3	52,000	7.9	85.1	62.0
CZES	frq/-/3	32,419	11.0	73.4	60.3
czTenTen12	frq/-/10	40,654	9.9	82.9	64.6
SYN	frq/-/10	34,667	11.1	79.1	64.0
CZES	frq/-/3	32,419	11.0	73.4	60.3

Table 6.8: Best results of Czech word sketch evaluation with gold standard: 3 best according to precision, 3 best according to recall and 3 best according to F-5. **Settings legend:** sorting / max. no. collocates per headword / minimum frequency; **frq:** sorting collocations by frequency, **log:** sorting collocations by logDice, **hl:** max. no. collocates is 25/12/6, according to headword frequency band.

system with appropriate settings is able to achieve about 90 percent recall (however, with very low precisions and high numbers of extracted collocations) and nearly 60 percent precision (40 for Czech). However, the best precision results achieve only 11–14 percent recall, so we do not consider them practically usable. The F-5 measure provides a good compromise between these two.

Also note that the best precision percentages are lower than in case of the previous evaluation – the reason is that this new evaluation counts the grey zone of annotators' disagreement as bad collocations, whereas the previous evaluation took them out from the testing set.

For precision, corpora prepared by careful selection of sources score better, such as the 100 million word British National Corpus [16] or Model corpus.[5] For Czech, the parsed corpora achieved best precision, but the SYN part of the Czech National Corpus (2.2 billion words) [95] scored well on recall and F-5 measure.

In both recall and F-5 measure, "the bigger, the better" roughly holds. Big web corpora are the general winners according to the F-5 score – namely *enClueWeb*, the 83 billion word English part of the ClueWeb crawl [141]; the

5. `www.sketchengine.co.uk/documentation/wiki/Corpora/NewModelCorpus`

Corpus	Settings	# collocs	Prec (%)	Rec (%)	F-5 (%)
CZES MST	log/hl/10	1,480	38.8	11.8	12.2
CZES Malt	log/hl/10	1,478	38.0	11.6	11.9
CZES Synt	log/hl/10	1,447	36.8	11.0	11.3
CZES	log/hl/10	1,447	36.8	11.0	11.3
CZES SET	log/hl/10	1,474	36.7	11.2	11.5
CZES	frq/-/3	32,419	11.0	73.4	60.3
CZES SET	frq/-/3	35,729	9.7	71.2	57.2
CZES MST	frq/-/3	32,581	10.5	70.5	57.8
CZES Malt	frq/-/3	32,471	10.5	70.2	57.6
CZES Synt	frq/-/3	18,708	15.5	59.9	54.0

Table 6.9: Comparison of Czech parsers by collocation extraction – best results according to precision, and according to recall/F-5. **Settings legend:** sorting / max. no. collocates per headword / minimum frequency; **frq:** sorting collocations by frequency, **log:** sorting collocations by logDice, **hl:** max. no. collocates is 25/12/6, according to headword frequency band.

enTenTen pair of corpora (3 and 13 billion words) and the Czech member of the family, 5 billion word *czTenTen* [61]; and the CZES corpus of about 500 million words [55].

Another important (and surprising) observation is that the raw frequency is better than the logDice score for collocation salience estimations – it outperformed the logDice practically in all important measures.

6.5.4 Parser comparison

We were interested in the influence of full parsing on the word sketch collocations, and in comparing Czech parsers using the collocation extraction task. We have selected the CZES corpus (for its moderate size), processed it by SET, Synt, MST Parser and MaltParser and compared the resulting word sketches with the standard ones produced by the sketch grammar.

The results are summarized in Table 6.9. Rather surprisingly, they show that the word sketches from full parsing are noticeably worse than word sketches from sketch grammar. MST Parser and MaltParser achieved slightly higher maximum precision, but at the cost of insufficient recall. In terms of recall and F-5 score, sketch grammar outperformed all the full parsers. Differences among SET, MaltParser and MST Parser are small, and Synt is slightly worse.

Parser	PDT score (%)	collocation extraction F-5 (%)
Sketch grammar	N/A	60.3
Synt	N/A	54.0
SET	56.0	57.2
MST Parser	84.7	57.8
MaltParser	85.8	57.6

Table 6.10: Comparison of unlabelled dependency precision score on Prague Dependency Treebank testing section, and F-5 score for collocation extraction.

These results substantially support our claim that specialized parsing methods designed directly for particular applications are better than general, mostly statistical parsers developed and tested according to the "treebank philosophy".

And again – although the MST Parser and MaltParser are considered much better than Synt and SET, according to the scores against the Prague Dependency Treebank data, there is no correlation with the results of collocation extraction testing, as illustrated in Table 6.10. This is another evidence supporting the SET design approach: Although its concept is much simpler than in case of the other three, the differences on a particular practically oriented task are minimal.

We believe that the parsers can be modified in order to score better on the collocation extraction task and outperform the shallow sketch grammar. However, as we have illustrated on the Czech example, this is not the case right now, even when using the best available parsers. We would like to conduct more experiments with parsers and collocation extraction in the future, namely to modify the parsers so that their output will better suit the collocation extraction task. It is clear that such modifications will be much easier in case of the SET parser with its transparent design and small rule-based grammar, than in case of the statistical parsers trained on a treebank and based on complex probabilistic models.

6.6 Terminology extraction

Terminology extraction using the Simple Maths formula and the sketch grammar analysis has been described in Section 4.2.3. Up to now, it has been implemented for 9 languages, in collaboration with native speakers of the

respective languages: English, French, Japanese, Korean, Chinese, Spanish, German, Portuguese and Russian.

6.6.1 Evaluations

The first 200 extraction results for the first 5 languages were manually evaluated by our partner – World Intellectual Property Organisation (WIPO) – as the automatic extraction will help them create an international term database ("termbase") for texts of patents and other resources they have access to. Only precision has been taken into account during the evaluation, and relatively strict and very specialized criteria were set, where e.g. each term candidate that contained word "method" or "device" was judged as bad. The data for the extraction were small (500 thousand to 5 million words) sample texts provided by WIPO. The results were as follows:

- English: 40%
- French: 17%
- Japanese: 20%
- Korean: 2%
- Chinese: 37%

which was good enough to win a contract, among (allegedly) 10 other candidate systems. Later, we modified the process in collaboration with the partner to make the results better: especially we have involved blacklist of unwanted words within the terms, and slightly modified the term grammars. A second round of manual evaluation was performed, for 100 best term candidates for each of the 5 languages. The results are summarized in Table 6.11, together with the legend provided by WIPO.

We have also performed an initial independent evaluation for English, using the GENIA corpus [74], in which all terms have been manually identified. Both keyword and term extraction was ran to obtain the top 2000 keywords and top 1000 multi-word terms. Terms manually annotated in GENIA as well as terms extracted by our tool were normalized before comparison (lowercase, spaces and hyphens removed) and the most frequent GENIA terms were looked up in the extraction results.

As for recall, 61 of the top 100, 275 of the top 500, and 489 of the top 1000 GENIA terms were found by the system. The terms from the top 100 that were not found did not consist of English words: most of them were acronyms, e.g. "EGR1", "STAT-6". Concerning precision, 53 out of the top 100 term candidates extracted by the system were annotated as terms in the GENIA corpus.

Language	Suitable	Not suitable	N/A
English	79%	5%	16%
Japanese	71%	29%	0%
French	70%	22%	8%
Chinese	51%	33%	16%
Korean	29%	68%	3%

Table 6.11: Results of the second round of manual evaluation of terminology extraction. **Suitable:** the term candidate is a relevant term in the given subject-field and is suitable for inclusion in the termbase. **Not suitable:** the term candidate is not a valid term in the given subject-field and is not suitable for inclusion in the termbase. Cases of truncated terms for which the correct full form was not extracted by the software were also scored as "not suitable". **N/A:** the term in question is truncated; however, the correct full form is also extracted by the software and the term is included somewhere in the list in the first 500 terms. As far as Korean is concerned, 37% of the terms marked as "non suitable" were proper terms, however, not belonging to the domain under investigation which indicates wrong selection of texts.

With respect to the fact that the system has not been tuned in any way according to the GENIA data and annotation style, we consider it a very promising result.

6.6.2 Bilingual term extraction

Within the WIPO project, we have also experimented with bilingual term extraction – finding translation candidates for extracted terms, using bilingual parallel data, aligned on sentences or sentence groups. Our approach to this task was combining the bilingual parallel word sketch procedure, as described in Section 4.2.2, with the term extraction machinery.

Firstly, we extract all the term candidates and mark them in the texts. Then a stochastic dictionary of translation candidates is created by the same method as described in Section 4.2.2, using the logDice co-occurrence score, but with terms as the basic units rather than words. The result is a dictionary with n best translations of each term.

This method has been evaluated manually by WIPO, for 4 language pairs: English-French, English-Japanese, English-Korean and English-Chinese. Again, small (500 thousands to 5 million words) sample texts provided by WIPO were used for creating the dictionaries. 100 random En-

glish terms were given and the system returned 10 best translations from the stochastic dictionary, for each of them, in each of the 4 languages. If the correct translation was among the candidates, it was considered right, otherwise wrong. The accuracy according to this methodology was as follows:

- French: 66%
- Japanese: 88%
- Korean: 35%
- Chinese: 79%

6.7 Automatic extraction of lexical semantic information

A lexical semantic resource Sholva is being developed by Marek Grác [37], together with a method for automatic detection of semantic classes for unknown words. The method is based on tracking collocations of the unknown word, within given syntactic relations, and comparing it with the semantic resource records – if one or more collocates of the unknown word occur predominantly with a given semantic class, then the unknown word is assigned to this class, too [37, pp. 73–81].

Experiments were conducted with Sholva and Czech WordNet [138], and with the *modifier* relation obtained from output of SET, Synt and Czech sketch grammar. Precision, recall and F-score of the automatic semantic class assignment were measured. The SET parser in combination with the Sholva semantic network produced the best results, with precision up to 80.1% (depending on other settings), recall up to 59.9% and best F-score 53.3%.

Later, the experiment was repeated using syntactic information from MaltParser and MST Parser for Czech, none of which outperformed the previous SET results. Again, an allegedly worse parser outperformed the top state-of-the-art ones, in a practically oriented task.

The identified semantic classes were then manually checked and added into the semantic network, raising its coverage on Czech nouns from 68.3% to 83.1%. The process is currently used for further extensions of the semantic network and for annotation of new semantic classes.

6.8 Valency frame induction

Lexicons of semantic valencies have been always considered crucial for natural language understanding – this is why big projects like FrameNet [5] or VerbaLex [44] were introduced, aimed at recording the semantic valency

Figure 6.10: Correspondence between manual VerbaLex verb frame and LDA automatic verb frame. *"člověk, rodič, žena, ..."* (*"man, parent, woman, ..."*) correspond to *"person:1"*, *"pivo, alkohol, jídlo, ..."* (*"beer, alcohol, food, ..."*) correspond to *"food:1"*.

information for individual words. Such resources have been criticised by empiric linguists for being based on introspection rather than corpus data. Also, they usually lack coverage and are expensive to build.

A novel method is in development by Jiří Materna which enables automatic induction of the word frames from corpus data, based on latent dirichlet allocation (LDA) [106]. The method has been partly evaluated for English, with promising results [106, 105]. A demo application has been created which provides a web interface to the available automatic valency lexicons.[6]

The LDA method requires parsed data on the input, in form of word tuples for the confident instances of the valency frames. For English, the Stanford parser was used [75], for Czech, it was the SET parser described in Chapter 5. For this purpose, a special option to the SET parser has been added for output suitable for this application. Although the Czech results have not been evaluated yet, the frames generally seem to be correct and consistent with the valency information provided by manually created VerbaLex lexicon, as illustrated in Figure 6.10.

6. `nlp.fi.muni.cz/projekty/lda-frames/view_frames.psp`

6.9 Czech phrase declension

Synthesis of texts in Czech is, thanks to the rich morphology and analytical character of the language, relatively complex task. However, it is needed by many subsequent applications, some of them of general use: generating user-friendly reports from databases or other formal records, transformation of sentences needed e.g. by the textual entailment application [126], localization of interfaces...

One of the necessary assumptions for correct sentence generation in Czech is proper declension of noun and prepositional phrases. This task has been addressed by Zuzana Nevěřilová [125] in context of the textual entailment application development mentioned above. She shows that for complex phrases, information from syntactic analysis is needed for detection of the phrase head, as declension of the words before and after the syntactic head works differently. The analysis by the SET parser was used for this task – no special additions were needed, as the head of a phrase is always the top word in the dependency tree for the phrase.

Manual evaluation of the phrase declension application was performed on a sample of 286 phrases [125], showing 90.6% accuracy. Based on this application, a more complex tool for generating Czech sentences based on formal JSON encoding was designed. Both of the applications have corresponding web interfaces.[7]

6.10 Anaphora resolution

Anaphora resolution for Czech is addressed by 3 different systems [132, 99, 131]. The accuracy of these systems is evaluated against anaphora manual annotation contained in the so called tectogrammatical layer of the Prague Dependency Treebank. The overall accuracy of all the systems seems to be around, or slightly above 40 percent, for all three systems. However, the available evaluations seem to test slightly different variants of the task (problems in evaluation are discussed in [132]) and the results are probably not directly comparable.

The task is usually solved in two steps:

- finding the anaphoric expressions and their possible antecedents (so called *markables*)
- resolving the coreference of markables

7. `nlp.fi.muni.cz/projekty/declension`
`nlp.fi.muni.cz/projekty/ner/generate`

For the first step, syntactic analysis is needed, as the markables form noun phrases or similar syntactic units – adjective phrases, clauses or sentences and sentence groups. One of the tools, Saara designed by Václav Němčík, is a framework that allows running several universal anaphora resolution algorithms [132]. It uses SET as its primary and default source of syntactic information, although processing pipeline for the Synt parser has also been implemented. A web version of this application, accepting Czech free text, and with Saara configured to resolve anaphora for personal pronouns, is freely available online.[8]

Recently, the development of yet another anaphora resolution application was started by Zuzana Nevěřilová, because of dissatisfaction with the available tools, Aara.[9] This tool is also based upon the syntactic information from the SET parser.

6.11 Conclusions

In this chapter, we have introduced the particular usages of the tools presented in the previous chapters. Where available, we have provided evaluations of accuracy, so that other tools, possibly exploiting different syntactic processing, can be compared with our results.

In some cases (e.g. noun phrase declension), it is relatively straightforward to replace the used parser with another one, although there are technical complications. In other cases, e.g. grammar checking or extraction of facts, our methods are tailored to the processing chain of the SET parser, and replacing it with another syntactic tool would be very complicated and only approximative, due to different character of the output information.

There are several other projects that we have not described in detail, aimed at applying natural language parsing to practical problems, where our input was rather limited, or the research is in an early stage yet, namely:

- morphological disambiguation for Czech within the Synt parser, reported in [59]
- automatic theme-rheme identification for Czech [139] exploiting the SET parser
- intrinsic corpus evaluation using the SET parser [4]
- translation of Czech sentences to formulae in transparent intensional logic, using the Synt parser [53, 89]

8. `nlp.fi.muni.cz/projekty/anaphora_resolution/saara/demo/`
9. So far, only a web demo is available:
`nlp.fi.muni.cz/projekty/watsonson/aara`

- question answering system for Czech, based on the same principle as the extraction of facts introduced in Section 6.1
- syntactic information retrieval system for Czech, based on the same principle

The variety of applications which use the syntactic analysis designed according to principles introduced in Section 2.5.5 and the SET parser in particular, confirms that our strategy, aimed at developing natural language parsing for real-world applications, is correct. According to the state of the art evaluation methodology, both of the parsing methods introduced in this work are substantially worse than the competition winners. In spite of that, they are much more usable and their results improve (or even make possible) functionality of useful applications. The SET parser is used in more practical applications than any other parser of Czech, including the supposedly best ones, and the results of an application based comparison show that its results are not significantly worse.

6.11.1 Note on parsing evaluation methodology

As we have shown in case of collocation extraction, accuracy according to the state of the art methodology does not correlate with the usability for a particular application. This result is consistent with previously cited studies [118, 68]. Together, they reveal that the state of the art evaluation methodology is obsolete and leads to harmful effects – instead of being useful for applications, syntactic analysis research aims at mimicking the testing data that have nothing in common with the needs of real applications.

We propose changing this methodology. Automatic syntactic analysis should not serve just for itself but to improve the following applications – and the evaluation of parsing should respect this fact. We propose that the evaluation of natural language parsers against treebanks is abandoned and replaced by extrinsic evaluation of the applications exploiting the parsers, in the same way as we performed it for collocation extraction in Section 6.5.4, and as reported in [118] and [68].

For the same reason, the natural language parsers should be developed primarily for the particular applications, not as general tools. Emphasis should be put on developing a suitable representation of the syntactic information needed (which is relative to each particular application), besides the parsing precision – as we have shown earlier, the sentence trees are far from being universal syntactic representation suitable for all applications. The very specific tree representations created according to treebank anno-

tation manuals are important for the linguistic theory but not for usage of syntactic information within applications.

Chapter 7

Conclusions

This work describes an application oriented approach to automatic natural language syntactic analysis. We have analyzed the state of the art methodology and identified problems leading to low usage of automatic syntactic analysis, despite large amount of work that has been done in this domain.

In Section 2.5, we have criticized evaluations of natural language parsers by comparing their output with manually annotated treebanks, and explained how this evaluation methodology leads to development of parsers unusable in practice. We have proposed an alternative methodology based on well known principles of software engineering where simplicity in design is the crucial key to usability and flexibility of the system. Usage of the parsing systems, and their flexibility in adding new procedures and output types, are necessary conditions for determining next meaningful research directions and moving the whole natural language processing field forward.

The next chapters describe our original research within the scope of the new approach. Chapter 3 introduces bushbank, an alternative format of syntactic annotation which overcomes some of the most painful problems of treebanks. Namely, it allows ambiguity in syntactic annotations and distinguishes between straightforward and complicated syntactic phenomena, as perceived by human beings. The particular implementation of the formalism is application-oriented – we aimed at recording syntactic phenomena directly usable by applications, and avoiding technical and non-intuitive annotations forced by the treebank formalisms. Last but not least, we have described the annotation process which is 10 times faster when compared to treebank annotation. According to the principles from Section 2.5.5, we have emphasized simplicity in all aspects of the annotation design – the guidelines for annotators are smaller by 2 orders of magnitude than guidelines for treebanks, with competitive inter-annotator agreement, and the annotation is easily readable by people not educated in theoretical linguistics.

The following chapter describes word sketches, a corpus-based syntactic engine for collocation extraction originally introduced by Pavel Rychlý and Adam Kilgarriff. We have described our work on word sketches leading to new applications of the formalism – collocation extraction for multiwords, bilingual collocation extraction, and terminology extraction. Chapter 5 introduces a new full parsing system SET based on pattern matching and a set of manually written rules. The rule format provides very economic, easy to maintain and readable description of syntactic phenomena.

Chapter 6 demonstrates the usability and flexibility of the proposed parsing engines, by describing a wide variety of NLP applications where the tools have been used. Evaluations of these applications show that despite their deliberate simplicity, the analysers are competitive with the state of the art tools, concerning the accuracy of the applications. Regarding usability and flexibility, measured by the number of applications, they have already overtaken all the current parsers for Czech.

We have constrasted evaluations based on applications and on bushbank, with the state of the art evaluation, which showed that they do not correlate well. This clearly indicates that the current methodology based on treebank evaluations does not measure the usefulness of the parsers with regard to practical applications.

The main message of this work can be summarized as: *"Let's base natural language syntactic processing purely on needs of practical applications that demand it, not the artificial syntactic descriptions contained in treebanks."*

We have provided evidence that the state of the art evaluation methodology is counterproductive, proposed new approach, and described results proving that the approach is meaningful. Of course, there is still much work to be done in the future but we believe that the ideas described in this thesis will contribute to faster and more intelligent natural language processing development.

Bibliography

[1] Steven Abney. Part-of-speech tagging and partial parsing. *Corpus-Based Methods in Language and Speech Processing*, 2:118–136, 1997.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*. ACM press New York, 1999.

[3] Vít Baisa and Vojtěch Kovář. Information extraction for Czech based on syntactic analysis. In *Proceedings of the 5th Language & Technology Conference*, pages 466–470, Poznań, Poland, 2011. Funcacja Universytetu im. A. Mickiewicza.

[4] Vít Baisa and Vít Suchomel. Intrinsic methods for comparison of corpora. In *Proceedings of Seventh Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 51–58, Brno, 2013. Tribun EU.

[5] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley Framenet project. In *Proceedings of the 17th international conference on Computational linguistics – Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[6] Dalibor Behún. Kontrola české gramatiky pro MS Office - konec korektorů v Čechách?, 2005.
`interval.cz/clanky/kontrola-ceske-gramatiky-pro-\`
`ms-office-konec-korektoru-v-cechach`.

[7] Morton Benson, Evelyn Benson, and Robert Ilson. *The BBI combinatory dictionary of English, 3rd edition*. John Benjamins, Amsterdam, Philadelphia, 1986.

[8] Iveta Beranová. Tvorba syntaktických pravidel pro detekci frází v rámci analyzátoru SET. Bachelor thesis, Masaryk University, 2013.
`is.muni.cz/th/382676/ff_b`.

[9] Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary A. Marcinkiewicz, and Britta Schasberger. Bracketing guidelines for treebank II style Penn treebank project, 1995. `languagelog.ldc.upenn.edu/myl/PennTreebank1995.pdf`.

[10] Joan Bresnan. *Lexical-functional syntax*. Blackwell, 2001.

[11] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-intensive question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC)*, pages 393–400, Gaithersburg, MD, USA, 2001. Department of Commerce, National Institute of Standards and Technology.

[12] Ted Briscoe and John Carroll. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1499–1504, Las Palmas, Gran Canaria, 2002. European Language Resources Association.

[13] Ted Briscoe, John Carroll, and Rebecca Watson. The second release of the RASP system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80. Association for Computational Linguistics, 2006.

[14] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics, 2006.

[15] Aleksander Buczyński and Adam Przepiórkowski. Spejd: A shallow processing and morphological disambiguation tool. In *Human Language Technology. Challenges of the Information Society*, pages 131–141. Springer, Berlin, 2009.

[16] Lou Burnard. *Users Reference Guide British National Corpus Version 1.0*. Oxford University Computing Services, UK, 1995.

[17] Jean Carletta. Assessing agreement on classification tasks: The kappa statistic. *Computational linguistics*, 22(2):249–254, 1996.

[18] Jean Carletta, Stefan Evert, Ulrich Heid, and Jonathan Kilgour. The NITE XML toolkit: data model and query language. *Language resources and evaluation*, 39(4):313–334, 2005.

[19] John Carroll. Parsing and real-world applications. In *Proceedings of Text, Speech and Dialogue, 13th International Conference, TSD 2010*, Lecture Notes in Computer Science, pages 3–5, Berlin, 2010. Springer.

[20] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Morgan Kaufmann Publishers Inc., 2000.

[21] Carole E. Chaski. The computational-linguistic approach to forensic authorship attribution. In *Law and Language: Theory and Society*, pages 119–144, Düsseldorf, 2008. Düsseldorf University Press.

[22] Zhiyi Chi. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160, 1999.

[23] Noam Chomsky. *Syntactic structures*. Janua linguarum: Series minor. Mouton, 1957.

[24] Stephen Clark and James R. Curran. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552, 2007.

[25] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog. Second edition*. Springer, 1984.

[26] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[27] Michael Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003.

[28] Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. A statistical parser for Czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 505–512. Association for Computational Linguistics, 1999.

[29] Jonathan Crowther, Sheila Dignen, and Diana Lea. *Oxford Collocations Dictionary for Students of English*. Oxford University Press, 2002.

[30] Bojan Djordjevic, James R. Curran, and Stephen Clark. Improving the efficiency of a wide-coverage CCG parser. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 39–47. Association for Computational Linguistics, 2007.

[31] Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1):61–103, 2009.

[32] Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. *Statistical methods for rates and proportions*. John Wiley & Sons, 2013.

[33] Richard P. Gabriel. Lisp: Good news, bad news, how to win big. *AI Expert*, 6:30–39, 1991.

[34] Richard P. Gabriel. Is worse really better? *Journal of Object-Oriented Programming (JOOP)*, 5(4):501–538, 1992.

[35] Marek Grác, Miloš Jakubíček, and Vojtěch Kovář. Through low-cost annotation to reliable parsing evaluation. In *PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 555–562, Sendai, Japan, 2010. Tohoku University.

[36] Ralph Grishman, Catherine Macleod, and John Sterling. Evaluating parsing strategies using standardized parse files. In *Proceedings of the third conference on Applied natural language processing*, pages 156–161. Association for Computational Linguistics, 1992.

[37] Marek Grác. *Rapid Development of Language Resources*. PhD thesis, Masaryk University, 2013.

[38] Jan Hajič. Complex corpus annotation: The Prague dependency treebank. *Insight into the Slovak and Czech Corpus Linguistics*, page 54, 2006.

[39] Jan Hajič, Jan Raab, Miroslav Spousta, et al. Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771. Association for Computational Linguistics, 2009.

[40] Eva Hajičová, Jan Hajič, Barbora Hladká, Martin Holub, Petr Pajas, Veronika Řezníčková, and Petr Sgall. The current status of the Prague dependency treebank. In *Proceedings of Text, Speech and Dialogue, 4th International Conference*, volume 2166 of *Lecture Notes in Computer Science*, pages 11–20, Berlin, 2001. Springer.

[41] Jan Hajič. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 106–132, Prague, 1998. Karolinum.

[42] Jan Hajič, Jarmila Panevová, Eva Buráňová, Zdeňka Urešová, Alla Bémová, Jan Štěpánek, Petr Pajas, and Jiří Kárník. Annotations at analytical level: Instructions for annotators, 2005. `ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/a-layer/pdf/a-man-en.pdf`.

[43] Ondřej Herman and Vojtěch Kovář. Methods for detection of word usage over time. In *Proceedings of Seventh Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 79–85, Brno, 2013. Tribun EU.

[44] Dana Hlaváčková and Aleš Horák. VerbaLex - new comprehensive lexicon of verb valencies for Czech. In *Computer Treatment of Slavic and East European Languages*, pages 107–115, Bratislava, Slovakia, 2006. Slovenský národný korpus.

[45] Jerry R. Hobbs. Discourse and inference: Magnum opus in progress, 2014. `www.isi.edu/~hobbs/disinf-tc.html`.

[46] Tomáš Holan, Vladislav Kuboň, and Martin Plátek. A prototype of a grammar checker for Czech. In *Proceedings of the 5th conference on Applied natural language processing*, pages 147–154. Association for Computational Linguistics, 1997.

[47] Tomáš Holan. Genetické učení závislostních analyzátorů. In *Sborník semináře ITAT 2005*, pages 47–54. UPJŠ, Košice, 2005.

[48] Tomáš Holan and Zdeněk Žabokrtský. Combining Czech dependency parsers. In *Proceedings of Text, Speech and Dialogue, 9th International Conference*, volume 4188 of *Lecture Notes in Computer Science*, pages 95–102, Berlin, 2006. Springer.

[49] Charles Hollingsworth. Using dependency-based annotations for authorship identification. In *Proceedings of Text, Speech and Dialogue, 15th International Conference*, volume 7499 of *Lecture Notes in Computer Science*, pages 314–319, Berlin, 2012. Springer.

[50] Aleš Horák. *The Normal Translation Algorithm in Transparent Intensional Logic for Czech*. PhD thesis, Masaryk University, 2002.

[51] Aleš Horák. *Computer Processing of Czech Syntax and Semantics*. Librix.eu, Brno, Czech Republic, 2008.

[52] Aleš Horák, Tomáš Holan, Vladimír Kadlec, and Vojtěch Kovář. Dependency and phrasal parsers of the Czech language: A comparison. In *Proceedings of Text, Speech and Dialogue, 10th International Conference*, volume 4629 of *Lecture Notes in Computer Science*, pages 76–84, Berlin, 2007. Springer.

[53] Aleš Horák, Miloš Jakubíček, and Vojtěch Kovář. Analyzing time-related clauses in transparent intensional logic. In *Proceedings of Fifth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 3–9, Brno, 2011. Tribun EU.

[54] Aleš Horák, Miloš Jakubíček, and Vojtěch Kovář. Linguistic logical analysis of direct speech. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 51–59, Brno, 2012. Tribun EU.

[55] Aleš Horák, Pavel Rychlý, and Adam Kilgarriff. *Czech Word Sketch Relations with Full Syntax Parser*, pages 101–112. Masaryk University, Brno, 2009.

[56] Richard A. Hudson. *Word grammar*. Blackwell Oxford, 1984.

[57] Abhilash Inumella, Adam Kilgarriff, and Vojtěch Kovář. Associating collocations with dictionary senses. In *Proceedings of 6th Biennial Conference of the Asian Association for Lexicography*, pages 102–113, Bangkok, Thailand, 2009. Asian Association for Lexicography.

[58] Miloš Jakubíček and Aleš Horák. Punctuation detection with full syntactic parsing. *Research in Computing Science, Special issue: Natural Language Processing and its Applications*, 46:335–343, 2010.

[59] Miloš Jakubíček, Aleš Horák, and Vojtěch Kovář. Mining phrases from syntactic analysis. In *Proceedings of Text, Speech and Dialogue, 12th International Conference*, volume 5729 of *Lecture Notes in Computer Science*, pages 124–130, Berlin, 2009. Springer.

[60] Miloš Jakubíček. Rule-based parsing of morphologically rich languages. Dissertation proposal, Masaryk University, 2012. `is.muni.cz/th/172962/fi_r`.

[61] Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlý, and Vít Suchomel. The TenTen corpus family. In *7th International Corpus Linguistics Conference CL 2013*, pages 125–127, Lancaster, 2013. Lancaster University.

[62] Miloš Jakubíček and Vojtěch Kovář. CzechParl: Corpus of stenographic protocols from Czech parliament. In *Proceedings of Fourth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 41–46, Brno, 2010. Masaryk University.

[63] Miloš Jakubíček and Vojtěch Kovář. Enhancing Czech parsing with verb valency frames. In *Computational Linguistics and Intelligent Text Processing - 14th International Conference, CICLing 2013*, volume 7816 of *Lecture Notes in Computer Science*, pages 282–293, Greece, 2013. Springer.

[64] Miloš Jakubíček, Vojtěch Kovář, and Aleš Horák. Measuring coverage of a valency lexicon using full syntactic analysis. In *Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 75–79, Brno, 2009. Masaryk University.

[65] Miloš Jakubíček, Vojtěch Kovář, and Pavel Šmerk. Czech morphological tagset revisited. In *Proceedings of Fifth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 29–42, Brno, 2011. Tribun EU.

[66] Miloš Jakubíček, Pavel Rychlý, Adam Kilgarriff, and Diana McCarthy. Fast syntactic searching in very large corpora for many languages. In *PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 741–747, Sendai, Japan, 2010. Tohoku University.

[67] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer, 1997.

[68] Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. Training a parser for machine translation reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 183–192. Association for Computational Linguistics, 2011.

[69] A. Kilgarriff, V. Kovář, S. Krek, I. Srdanović, and C. Tiberius. A quantitative evaluation of Word Sketches. In *Proceedings of the XIV Eu-*

ralex International Congress, pages 372–379, Ljouwert, Netherlands, 2010. Fryske Akademy.

[70] Adam Kilgarriff. Simple maths for keywords. In *Proceedings of the Corpus Linguistics Conference*, Liverpool, 2009. University of Liverpool.

[71] Adam Kilgarriff, Vojtěch Kovář, and Pavel Rychlý. Tickbox lexicography. In *eLexicography in the 21st century: New challenges, new applications*, pages 411–418, Louvain la Neuve, Belgium, 2010. Presses universitaires de Louvain.

[72] Adam Kilgarriff, Pavel Rychlý, Vojtěch Kovář, and Vít Baisa. Finding multiwords of more than two words. In *Proceedings of the 15th EURALEX International Congress*, pages 693–700, Oslo, 2012. University of Oslo.

[73] Adam Kilgarriff, Pavel Rychlý, Pavel Smrž, and David Tugwell. The Sketch Engine. *Information Technology*, 105:116–127, 2004.

[74] Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. GENIA corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182, 2003.

[75] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10. MIT Press, 2003.

[76] Graham Klyne, Jeremy J. Carroll, and Brian McBride. Resource description framework (RDF): Concepts and abstract syntax. *W3C recommendation*, 10, 2004.

[77] Lucia Kocincová. Využití syntaktických analyzátorů pro získávání kolokací v korpusech. Bachelor thesis, Masaryk University, 2013. `is.muni.cz/th/374080/fi_b`.

[78] Miloslav Konopík and Ondřej Rohlík. Question answering for not yet semantic web. In *Proceedings of Text, Speech and Dialogue, 13th International Conference, TSD 2010*, Lecture Notes in Computer Science, pages 125–132, Berlin, 2010. Springer.

[79] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology*, 60(1):9–26, 2009.

[80] Iztok Kosem, Vít Baisa, Vojtěch Kovář, and Adam Kilgarriff. User-friendly interface of error/correction-annotated corpus for both teachers and researchers. In *Book of Abstracts LCR 2013*, pages 82–84, Bergen, 2013. University of Bergen.

[81] Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Syntactic analysis as pattern matching: The SET parsing system. In *Proceedings of 4th Language & Technology Conference*, pages 978–983, Poznań, Poland, 2009. Wydawnictwo Poznańskie.

[82] Vojtěch Kovář and Aleš Horák. Reducing the number of resulting parsing trees for the Czech language using the beautified chart method. In *Proceedings of the 3rd Language & Technology Conference*, pages 433–437, Poznań, Poland, 2007. Wydawnictwo Poznańskie.

[83] Vojtěch Kovář, Aleš Horák, and Vladimír Kadlec. New methods for pruning and ordering of syntax parsing trees. In *Proceedings of Text, Speech and Dialogue, 11th International Conference*, volume 5246 of *Lecture Notes in Computer Science*, pages 125–131, Berlin, 2008. Springer.

[84] Vojtěch Kovář and Miloš Jakubíček. Test suite for the Czech parser Synt. In *Proceedings of Second Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 63–70, Brno, 2008. Masaryk University.

[85] Vojtěch Kovář and Miloš Jakubíček. Prague dependency treebank annotation errors: A preliminary analysis. In *Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 101–108, Brno, 2009. Masaryk University.

[86] Vojtěch Kovář. Corpus query system Bonito - recent development. In *Proceedings of First Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 71–76, Brno, 2007. Masaryk University.

[87] Vojtěch Kovář and Aleš Horák. Power networks dialogues - automatic analysis and evaluation of a domain-specific text corpus. In *Proceedings of ELNET 2007*, pages 30–37, Ostrava, 2007. Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava.

[88] Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Power networks dialogs - enhancing domain-specific text processing techniques and resources. In *Proceedings of ELNET 2008*, pages 72–80, Ostrava, 2008. Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava.

[89] Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. How to analyze natural language with transparent intensional logic? In *Proceedings of Fourth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 69–76, Brno, 2010. Masaryk University.

[90] Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Syntactic analysis using finite patterns: A new parsing system for Czech. In *Human Language Technology. Challenges for Computer Science and Linguistics*, volume 6562 of *Lecture Notes in Computer Science*, pages 161–171, Berlin, 2011. Springer.

[91] Vojtěch Kovář, Miloš Jakubíček, and Jan Bušta. Czech vulgarisms in text corpora. In *After Half a Century of Slavonic Natural Language Processing*, pages 141–145, Brno, 2009. Tribun EU.

[92] Vojtěch Kovář, Miloš Jakubíček, and Aleš Horák. Syntactic parser SET, 2012. `nlp.fi.muni.cz/projects/set`.

[93] Vojtěch Kovář, Vladimír Kadlec, and Aleš Horák. Grammar development for Czech syntactic parser with corpus-based techniques. In *Proceedings of Corpus Linguistic 2006*, pages 159–165, Saint-Petersburg, 2006. Saint-Petersburg State University.

[94] Vladislav Kuboň, Markéta Lopatková, Martin Plátek, and Patrice Pognan. Segmentation of complex sentences. In *Proceedings of Text, Speech and Dialogue, 9th International Conference*, volume 4188 of *Lecture Notes in Computer Science*, pages 151–158, Berlin, 2006. Springer.

[95] Karel Kučera. The Czech National Corpus: principles, design, and results. *Literary and linguistic computing*, 17(2):245–257, 2002.

[96] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[97] Dekang Lin. PRINCIPAR: An efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th conference on Computa-*

tional linguistics – Volume 1, pages 482–488. Association for Computational Linguistics, 1994.

[98] Dekang Lin. Dependency-based evaluation of MINIPAR. In *Treebanks: Building and using parsed corpora*, pages 317–329. Springer, Berlin, 2003.

[99] Nguy Giang Linh. Návrh souboru pravidel pro analýzu anafor v českém jazyce. Master thesis, Charles University, 2006.

[100] Markéta Lopatková, Natalia Klyueva, and Petr Homola. Annotation of sentence structure: capturing the relationship among clauses in Czech sentences. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 74–81. Association for Computational Linguistics, 2009.

[101] Christopher D. Manning. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011*, pages 171–189. Springer, Berlin, 2011.

[102] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[103] Mitchell P. Marcus, Marry A. Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, 1993.

[104] James Martin. *Rapid application development*. Macmillan, 1991.

[105] Jiří Materna. Building a thesaurus using LDA-Frames. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 97 –103, Brno, 2012. Tribun EU.

[106] Jiří Materna. LDA-Frames: An unsupervised approach to generating semantic frames. In *Computational Linguistics and Intelligent Text Processing, 13th International Conference, CICLing 2012*, volume 7181 of *Lecture Notes in Computer Science*, pages 376–387, Berlin, 2012. Springer.

[107] Pavel Materna, Marie Duží, and Bjorn T. F. Jespersen. *Procedural Semantics for Hyperintensional Logic*. Springer, Berlin, 2010.

[108] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics, 2006.

[109] Ryan McDonald. *Discriminative learning and spanning tree algorithms for dependency parsing*. PhD thesis, University of Pennsylvania, 2006.

[110] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Vancouver, BC, Canada, 2005. Association for Computational Linguistics.

[111] Marek Medved', Miloš Jakubíček, and Vojtěch Kovář. Towards taggers and parsers for Slovak. In *Proceedings of 6th Language & Technology Conference*, pages 1–4, Poznań, Poland, 2013. Wydawnictwo Poznańskie.

[112] Marek Medved', Miloš Jakubíček, Vojtěch Kovář, and Václav Němčík. Adaptation of Czech parsers for Slovak. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 23–30, Brno, 2012. Tribun EU.

[113] Marie Mikulová and Jan Štěpánek. Annotation procedure in building the Prague Czech-English dependency treebank. In *Slovko 2009, NLP, Corpus Linguistics, Corpus Based Grammar Research*, pages 241–248, Bratislava, Slovakia, 2009. Slovenská akadémia vied.

[114] George A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[115] George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[116] Ruslan Mitkov. *Anaphora resolution*. Longman London, 2002.

[117] Yusuke Miyao, Alastair Butler, Kei Yoshimoto, and Jun'ichi Tsujii. A modular architecture for the wide-coverage translation of natural language texts into predicate logic formulas. In *PACLIC 24 Proceedings*

of the 24th Pacific Asia Conference on Language, Information and Computation, pages 481–488, Sendai, Japan, 2010. Tohoku University.

[118] Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun'ichi Tsujii. Evaluating contributions of natural language parsers to protein–protein interaction extraction. *Bioinformatics*, 25(3):394–400, 2009.

[119] Jaroslav Moravec, Vojtěch Kovář, Jan Bušta, and Miloš Jakubíček. OOCorr, 2010. `nlp.fi.muni.cz/projects/oocorr`.

[120] Andrea Moro and Roberto Navigli. Integrating syntactic and semantic analysis into the open information extraction paradigm. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 2148–2154. AAAI Press, 2013.

[121] Eva Mráková and Radek Sedláček. From Czech morphology through partial parsing to disambiguation. In *Computational Linguistics and Intelligent Text Processing - 11th International Conference, CICLing 2010*, volume 6008 of *Lecture Notes in Computer Science*, pages 126–135, Berlin, 2010. Springer.

[122] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[123] Tetsuji Nakagawa. Multilingual dependency parsing using global features. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 952–956. Association for Computational Linguistics, 2007.

[124] Zuzana Nevěřilová. Building evaluation dataset for textual entailment in Czech. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 53–58, Brno, 2012. Tribun EU.

[125] Zuzana Nevěřilová. Declension of Czech noun phrases. In *Actes du 31e Colloque International sur le Lexique et la Grammaire*, pages 134–138, České Budějovice, 2012. Université de Bohême du Sud à České Budějovice (République tchèque).

[126] Zuzana Nevěřilová and Marek Grác. Common sense inference using verb valency frames. In *Proceedings of Text, Speech and Dialogue, 15th International Conference*, volume 7499 of *Lecture Notes in Computer Science*, pages 328–335, Berlin, 2012. Springer.

[127] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359. Association for Computational Linguistics, 2009.

[128] Joakim Nivre, Johan Hall, and Jens Nilsson. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th international conference on Language Resources and Evaluation (LREC 2006)*, volume 6, Genoa, Italy, 2006. European Language Resource Association, Paris.

[129] Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106. Association for Computational Linguistics, 2005.

[130] Václav Novák and Zdeněk Žabokrtský. Feature engineering in maximum spanning tree dependency parser. In *Proceedings of Text, Speech and Dialogue, 10th International Conference*, volume 4629 of *Lecture Notes in Computer Science*, pages 92–98, Berlin, 2007. Springer.

[131] Michal Novák and Zdeněk Žabokrtský. Resolving noun phrase coreference in Czech. In *Anaphora Processing and Applications*, pages 24–34. Springer, Berlin, 2011.

[132] Václav Němčík. Saara: Anaphora resolution on free text in Czech. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 3–8, Brno, 2012. Tribun EU.

[133] Franz J. Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics, 2000.

[134] Karel Oliva, Vladimír Petkevič, and Microsoft s.r.o. Czech grammar checker, 2005. `office.microsoft.com/word`.

[135] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[136] Karel Pala. Pište dopisy konečně bez chyb – Český gramatický korektor pro Microsoft Office. *Computer*, pages 13–14, 2005.

[137] Karel Pala, Pavel Rychlý, and Pavel Smrž. DESAM — annotated corpus for Czech. In *Proceedings of SOFSEM'97*, pages 523–530, Berlin, 1997. Springer.

[138] Karel Pala and Pavel Smrž. Building Czech Wordnet. *Romanian Journal of Information Science and Technology*, 7:79–88, 2004.

[139] Karel Pala and Ondřej Svoboda. Semi-automatic theme-rheme identification. In *Proceedings of Seventh Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 39–48, Brno, 2013. Tribun EU.

[140] Carl Pollard. *Head-driven phrase structure grammar*. University of Chicago Press, 1994.

[141] Jan Pomikálek, Miloš Jakubíček, and Pavel Rychlý. Building a 70 billion word corpus of English from ClueWeb. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 502–506. European Language Resources Association, 2012.

[142] Martin Popel, David Mareček, Jan Štepánek, Daniel Zeman, and Zdeněk Žabokrtský. Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 517–527, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

[143] Karl Popper. *The logic of scientific discovery*. Routledge, 1959.

[144] Adam Radziszewski and Marek Grác. Using low-cost annotation to train a reliable Czech shallow parser. In *Proceedings of Text, Speech and Dialogue, 16th International Conference*, volume 8082 of *Lecture Notes in Computer Science*, pages 575–1156, Berlin, 2013. Springer.

[145] Adam Radziszewski and Maciej Piasecki. A preliminary noun phrase chunker for Polish. In *Intelligent Information Systems*, pages 169–180, Berlin, 2010. Springer.

[146] Michael Rundell. *Macmillan Collocations Dictionary*. Macmillan, 2010.

[147] Pavel Rychlý and Pavel Smrž. Manatee, Bonito and Word Sketches for Czech. In *Proceedings of the Second International Conference on Corpus Linguisitcs*, pages 124–132, Saint-Petersburg, 2004. Saint-Petersburg State University Press.

[148] Pavel Rychlý. A lexicographer-friendly association score. In *Proceedings of Second Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 6–9, Brno, 2008. Masaryk University.

[149] Pavel Rychlý and Vojtěch Kovář. Displaying bidirectional text concordances in KWIC format. In *Proceedings of 5th Biennial Conference of the Asian Association for Lexicography*, pages 61–66, Madras, India, 2007. University of Madras.

[150] Jan Rygl and Aleš Horák. Authorship attribution: Comparison of single-layer and double-layer machine learning. In *Proceedings of Text, Speech and Dialogue, 15th International Conference*, volume 7499 of *Lecture Notes in Computer Science*, pages 282–289, Berlin, 2012. Springer.

[151] Jan Rygl and Aleš Horák. Similarity ranking as attribute for machine learning approach to authorship identification. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, 2012. European Language Resources Association.

[152] Jan Rygl, Kristýna Zemková, and Vojtěch Kovář. Authorship verification based on syntax features. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 111–119, Brno, 2012. Tribun EU.

[153] Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. Multiword expressions: A pain in the neck for NLP. In *Computational Linguistics and Intelligent Text Processing*, pages 1–15, Berlin, 2002. Springer.

[154] Geoffrey Sampson. A proposal for improving the measurement of parse accuracy. *International Journal of Corpus Linguistics*, 5(01):53–68, 2000.

[155] Geoffrey Sampson and Anna Babarczy. A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, 9(04):365–380, 2003.

[156] Geoffrey Sampson and Anna Babarczy. Definitional and human constraints on structural annotation of English. *Natural Language Engineering*, 14(4):471–494, 2008.

[157] Petr Savický and Jaroslava Hlaváčová. Measures of word commonness. *Journal of Quantitative Linguistics*, 9(3):215–231, 2002.

[158] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12, pages 44–49. Manchester, UK, 1994.

[159] Radek Sedláček and Pavel Smrž. A new Czech morphological analyser ajka. In *Proceedings of Text, Speech and Dialogue, 4th International Conference*, volume 2166 of *Lecture Notes in Computer Science*, pages 100–107, Berlin, 2001. Springer.

[160] Petr Sgall. *Generativní popis jazyka a česká deklinace (Generative Description of the Language and the Czech Declension)*. Academia, Prague, 1967.

[161] Stuart M. Shieber. Evidence against the context-freeness of natural language. In *The Formal Complexity of Natural Language*, pages 320–334. Springer, Berlin, 1987.

[162] Arne Skjærholt. Influence of preprocessing on dependency syntax annotation: speed and agreement. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 28–32. Association for Computational Linguistics, 2013.

[163] Vladimír Šmilauer. *Novočeská skladba*. SPN, 1969.

[164] Irena Srdanović, Naomi Ida, Chikako Shigemori Bučar, Adam Kilgarriff, and Vojtěch Kovář. Japanese word sketches: Advances and problems. *Acta Linguistica Asiatica*, 1(2):63–82, 2011.

[165] Lingea s.r.o. Grammaticon, 2003. `www.lingea.cz/grammaticon.htm`.

[166] Mark Steedman and Jason Baldridge. Combinatory categorial grammar. In *Non-Transformational Syntax*, pages 181–224. Blackwell, 2011.

[167] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer, 2008.

[168] Asher Stern and Ido Dagan. The BIUTEE research platform for transformation-based textual entailment recognition. *Linguistic Issues in Language Technology*, 9:1–26, 2013.

[169] Lucien Tesnière. *Élements de syntaxe structurale*. C. Klincksieck, 1959.

[170] Barbora Trifanová. Analýza chyb v diktátech žáků po absolvování 1. stupně ZŠ. Bachelor thesis, Masaryk University, 2014. `is.muni.cz/th/382965/ff_b`.

[171] Pavel Šmerk. Unsupervised learning of rules for morphological disambiguation. In *Proceedings of Text, Speech and Dialogue, 7th International Conference*, volume 3206 of *Lecture Notes in Computer Science*, pages 211–216, Berlin, 2004. Springer.

[172] Eva Žáčková. *Parciální syntaktická analýza (češtiny)*. PhD thesis, Masaryk University, 2002.

[173] Yorick Wilks and Mark Stevenson. Sense tagging: Semantic tagging with a lexicon. In *Proceedings of the SIGLEX Workshop Tagging Text with Lexical Semantics: What, why and how*, pages 47–51. Association for Computational Linguistics, 1997.

[174] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

[175] Daniel Zeman. Neprojektivita v Pražském závislostním korpusu (PDT). Technical Report TR-2004-22, ÚFAL/CKL MFF UK, Prague, 2004.

[176] Daniel Zeman. *Parsing with a Statistical Dependency model*. PhD thesis, Charles University, MFF, 2004.

Appendix A

Czech bushbank manual for annotators

NP

- **NP je fráze, jejíž hlavou je podstatné či přídavné jméno, zájmeno, číslovka, předložka nebo zkratka (k[12347A]) včetně svých bezpředložkových rozvití**
  - např. zájmena ”který”, ”jaký”, ”se”, ”si” jsou NP
  - ”domnívám se” – NP je ”se”
  - ”vymínil si” – NP je ”si”
  - ”ceny tepla by mohly být i nižší” – NP jsou ”ceny tepla” a ”nižší”
  - ”opatření by nutilo ke změně chování spotřebitele paliv, kteří ještě stále investují...” – NP jsou ”opatření”, ”ke změně chování spotřebitele paliv”, ”kteří”
  - ”musíme určit, za jakých podmínek to uděláme” – NP jsou ”za jakých podmínek”, ”to”
- **do NP patří předložka včetně ”jako”, ”než”, ”okolo”, ”kolem” POUZE NA ZAČÁTKU, tzn. každá NP je oddělena předložkou**
  - ”jako tvrdé politické rozhodnutí vlády” je NP
  - ”mnohem vyšší než náklady v České republice” – ”než náklady” je NP, ”v České republice” je NP
  - ”s nimiž se čtenáři setkali jako spotřebitelé” – ”jako spotřebitelé” je NP
  - ”s návratností do 10 let” – NP jsou ”s návratností” a ”do 10 let”
- **ALE pokud po ”jako”, ”než” bezprostředně nenásleduje NP, není součástí NP**
  - ”jako byla cena rodinného domku” – NP je pouze ”cena rodinného domku”
- **do NP nepatří spojky (”aby, že, protože, jak...”) spojující jednotlivé věty**
- **NP může obsahovat spojku pouze uprostřed; a pouze takovou, která souřadně spojuje přívlastky**

- ″černá a bílá kočka″ je NP
- ″ministerstvo školství a mládeže″ je NP

- **víceslovné předložky rozdělujeme:**
  - ″se zřetelem na správní řízení″ – NP jsou ″se zřetelem″ a ″na správní řízení″
  - ″vzhledem ke komplikacím″ – NP jsou ″vzhledem″ a ″ke komplikacím″
- **a s jednoslovnými nevlastními předložkami zacházíme jako s předložkami vlastními:**
  - např. ″blízko″, ″doprostřed″, ″dovnitř″, ″nedaleko″, ″oproti″, ″poblíž″, ″uprostřed″, ″vprostřed″, ″zprostřed″ – tedy nevlastní předložky jednoslovné vzniklé z místních příslovcí
  - ″jsou poblíž nás″ – NP je ″poblíž nás″
  - ″uprostřed náměstí″ je NP
- **v případě, že jsou dvě NP vedle sebe bez předložky, musíme je rozdělit, pokud sémanticky nepatří k sobě, čili druhá není rozvitím první:**
  - ″dala mi to″ – NP jsou ″mi″ a ″to″
  - ″pošleme vám zboží″ – NP jsou ″zboží″a ″vám″
  - ″jak vypadal svět před 100 lety očima vesmíru″ – NP jsou ″svět″, ″před sto lety″ a ″očima vesmíru″
  - ″z nichž každý″ – správné NP jsou ″z nichž″ a ″každý″
- **NP nezačíná ani nekončí na interpunkci ani na žádné jiné divné znaky (hvězdička, závorka, pomlčka, uvozovky a jiné)**
  - ″na 5 tisíc Kč,″ ″příliš nákladné,″ ″*Michal Malý″ NEJSOU NP
- **správná je jen nejdelší možná NP podle pravidel výše**
  - ″zcela jiné starosti″ je správná NP, ″jiné starosti″ nebo ″starosti″ jsou špatné NP
  - ″okolo půl milionu″ je správná NP, ″milionu″ nebo ″půl milionu″ jsou špatné NP
  - ″nadějnost českého trhu″ je jedna NP
- **správné NP jsou také:**
  - ″10
  - ″100 Kč″
  - ″20 až 30 dnů″
  - ALE ″od 200 do 600 m″ – NP jsou ″od 200″ a ″do 600m″

- **přístavky a jiná interpunkcí oddělená bližší určení určujeme jako NP zvlášť**
  - ”Marie Nováková, učitelka na základní škole” – NP jsou ”Marie Nováková”, ”učitelka” a ”na základní škole”
  - ALE ”učitelka Marie Nováková” je jedna NP – ”Marie Nováková” je špatná NP
- **příslovce patří do NP pouze pokud před ně lze postavit předložku**
  - ”téměř tři hodiny” – můžeme říci ”za téměř tři hodiny”, tudíž téměř patří do NP
  - stejně tak i příslovce ”asi”, ”přibližně”, ”například nebo na příklad” (”například zelenina” je NP)
  - ALE nepatří ”už”, již” (”už dvě hodiny” – NP je ”dvě hodiny” bez už)
- **do NP nepatří slovesa, částice a citoslovce**
  - ”až 500 kč” – NP je pouze ”500 kč”

COORD

- **skládá se z několika NP + spojovací výrazy (čárka, spojka)**
  - ”džípy a obrněné vozy” je koordinace, ”džípy” a ”obrněné vozy” jsou NP
  - ”pes nebo kočka” je koordinace a ”pes” a ”kočka” jsou NP
  - ”originální rozhodnutí či certifikát” je koordinace, a ”originální rozhodnutí” a ”certifikát” jsou NP
- **aby šlo o koordinaci, je nutné, aby byl hlavní (řídící) člen NP několikanásobný**
  - ”na nákup chemikálií a vybavení” není koordinace, ale NP, protože řídící člen je ”nákup”
  - ”ministerstvo školství, mládeže a tělovýchovy” není koordinace – protože hlavní člen je ”ministerstvo”
- **pokud před několikanásobným větným členem stojí předložka, NENÍ to koordinace, protože hlavou fráze a řídícím členem je předložka**
  - ”mluvil o míru a svornosti” – ”míru a svornosti” závisí na předložce ”o”, není koordinace
  - ”bílá a černá kočka” NENÍ koordinace, muselo by být ”bílá kočka a černý pes”

- **nezačíná ani nekončí na interpunkci ani na žádné jiné divné znaky (hvězdička, závorka, pomlčka, uvozovky a jiné)**
- **pro případ typu "sice X, ale Y" jsou "Y" a "X" NP a "sice X, ale Y" je koordinace**
    - "sice dům, ale malý" je koordinace

VP

- **VP je slovesná fráze, obsahuje POUZE slovesné tvary a nic jiného**
    - včetně tvarů: bych, bys, by, bychom, bysme
    - včetně pomocných tvarů slovesa být a jejich negací: jsem, jsi, je, jsme, jste, jsou, budu, budeš, bude, budeme, budete, budou)
- **jedna VP obsahuje právě jedno významové sloveso =¿ jednoduchá věta může obsahovat víc VP**
    - "šel nakoupit" – VP jsou "šel" a "nakoupit"
    - "šel bys nakoupit" – správné VP jsou "šel bys" a "nakoupit"
    - "budu hrát fotbal" – VP je "budu hrát"
    - "vše jsme uplatnili" – VP je "jsme uplatnili"
- **spojky "aby", "kdyby" a jejich tvary, tj. "abych, abys, aby, abychom, abyste, aby..." do VP nepatří**
    - "abych šel" – VP je pouze "šel"
- **jedna VP také spolu s významovým slovesem obsahuje slovesa modální i jejich negace, tedy "chtít", "mít (povinnost)", "moci", "umět", "smět", "muset", "lze", "nelze", "nechtít"**
    - "může dojít k sejmutí klateb" – VP je "může dojít"
    - "rozhodnutí musí zrát" – VP je "musí zrát"
    - "má se stát prezidentem" – VP je "má stát"
    - "úkony nelze hradit" – VP je "nelze hradit"
    - "bude muset o odvolání uchazeče rozhodnout" – VP je "bude muset rozhodnout"
    - "mohlo by dojít k sejmutí klateb" – VP je "mohlo by dojít", "mohlo dojít" je špatná VP
- **fázová slovesa tvoří samostatnou VP**
    - "začít bruslit" – VP jsou "začít" a "bruslit"
    - "potřebuji spravit" – VP jsou "potřebuji" a "spravit"
    - "ošetřující lékař odmítl sdělit" – VP jsou "odmítl" a "sdělit"
- **VP NIKDY neobsahuje se/si**

  - ″domnívám se″ – VP je pouze ″domnívám″
- **několikanásobný přísudek rozdělujeme:**
  - ″musí ošetřovat a bodovat″ – VP jsou ″musí ošetřovat″ a ″bodovat″
- **VP u přísudku jmenného:**
  - ″je potřeba zakročit″ – VP jsou ″je″ a ″zakročit″; ″potřeba″ je NP
  - stejným způsobem například ″je dobré udělat″
- **jedna VP obsahuje i sloveso v příčestí trpném**
  - ″byl zabit″ – je VP
  - ″byl jeho post předurčen″ – VP je ″byl předurčen″
- **ALE pokud je tam adjektivum nebo pokud je příčestí trpné odvozeno od adjektiva (tj. nelze utvořit slovesný infinitiv), není součástí VP**
  - ″je ochoten″ – VP je pouze ″je″, ″ochoten″ je NP
  - ale ″byl zabit″ je VP
  - ″byl zabitý″ – VP je ″byl″, ″zabitý″ je NP
- **slovesná spojení typu ″mít uděláno″ jsou jedna VP, stejně tak příčestí trpné je jedna VP i když je pomocné ″je″ vypuštěno**
  - ″mít spočteno″ – jedna VP
  - ″sečteno a podtrženo″ – VP jsou ″sečteno″ a ″podtrženo″
- **VP nezačíná ani nekončí na interpunkci ani na žádné jiné divné znaky (hvězdička, závorka, pomlčka, uvozovky a jiné)**
  - ″*chci podpořit″ je špatná VP
- **VP určujeme i pokud je v závorce**
  - ″Firma (její jméno neuvádíme) se skládá ze dvou ekonomicky a obchodně samostatných divizí.″ – VP jsou ″neuvádíme″, ″skládá″
- **VP neobsahuje ″-li″**
  - ″pomineme-li bezdrátové telefony″ – VP je pouze ″pomineme″

CLAUSE

- CLAUSE je jednoduchá věta (může obsahovat více VP)
- celá zobrazená věta je CLAUSE, pokud se nejedná o souvětí (nemusí obsahovat VP)
- podřadící spojka na začátku nepatří do CLAUSE (že, aby)
- souřadící spojky na začátku nepatří do clause
- relativní zájmena a příslovce tam patří (který, jenž, kolik, kdy, jak)

- CLAUSE nezačíná ani nekončí na interpukci (ale může ji obsahovat)
- nezačíná na 3), 15), §264, ...
- může obsahovat zátvorky, pokud v nich není další CLAUSE

RELATIONS

Navázání na NP:

- snažíme se vytvořit nejdelší platnou NP a navazujeme vždycky na řídící NP
  - ”muž s dalekohledem”: ”s dalekohledem” → ”noun: muž”
- je v pořádku nepřiřadit NP nikam
- když jsou NP součástí koordinace, tak se váže k slovesu nebo k jiné NP jen koordinace a NP se navazují na koordinaci
- ”pejsek a kočička běhali”:
  - ”pejsek a kočička” → ”verb: běhali”
  - ”pejsek” → ”noun: pejsek a kočička”
  - ”kočička” → ”noun: pejsek a kočička”
- pokud se NP nenavazuje nikam, použijte ”not identified yet” nebo neurčujte
- nikdy nenavazujeme na strukturu (VP, NP), která je označena znaménkem ”-”, tedy je špatně, ani nenavazujeme samotnou špatnou strukturu
- pokud může být daná NP závislá na více frázích (NP, VP – např. doplněk), určete ji podle významu (a sebe)

Navázání na VP:

- navazujeme vždy na VP, která se vyskytuje ve stejné jednoduché větě (pokud má věta více VP navazujeme k té, ke které se váže)
- ”se/si”, ”který”, ”jaký” jsou NP a vážeme je k příslušnému slovesu

Appendix B

English bushbank manual for annotators

NP

- **NP (noun phrase) consists of nouns, adjectives, pronouns, numerals, prepositions, abbreviation or participles or gerunds**
  - pronouns like which, what, that, his etc. are NP
- **participles count as part of NP as they act as adjectives**
  - "the long and winding road" is NP
  - in "all the research mentioned is on memory", "all the research mentioned" is NP, "on memory" is NP and "is" is VP
- **gerunds count as part of NP as they act as nouns**
  - in "eating a cake is easy", "eating a cake" is NP
- **NP are separated by relevance, generally indicated by a preposition – prepositions have to be the first word of NP**
  - ″over 100 million in six years″ – ″over 100 million″ and ″in six years″ are two separated NP (also ″in″ is a preposition)
  - ″like mobile telephones″ is NP
  - ″their popular line of mobile music players are called iPods″ – ″their popular line″, ″of mobile music players″ and ″iPods″ are NP
- **note that there are also multiple word prepositions, e.g. "because of", "as for", "according to" – they are considered single prepositions**
  - "because of a giant sheep" is one NP
- **words like ″near″, ″far″ we consider preposition (they can be only at the begining of NP)**
- **non-coordinating conjunctions, e.g. ″as″, ″if″, ″when″ are not NP**
- **coordinating conjunctions, e.g. "and", "or", "but", "yet" cannot be at the beginning or at the end of NP**
  - ″word and voice memos″ is one NP (″word″ and ″voice″ develop word ″memos″)

  - ”and the papers” is not NP
- **punctuation (or other symbols like asterisks, dashes, brackets, quotation marks etc.) cannot be at the begining or at the end of NP**
  - ”e-mail,” is not NP
- **article is part of NP (but not at the end of NP)**
  - ”the difference” is NP
- **appositions are independent NP only when separated by commas**
  - ”Tim Cook, Chief Executive Officer” – ”Tim Cook” and ”Chief Executive Officer” are two separated NP
- **only the longest possible NP is correct, e.g.:**
  - ”of the most popular products ever developed”
  - ”about 2,000 beautiful touching songs”
  - ”in February 13, 2008”
  - ”8 GB”
- **when a preposition is not part of a phrasal verb and the object of the preposition is missing or is a whole clause, only the preposition itself is a NP**
  - in “the thing to think of”, “the thing” is NP, “to think” is VP, “of” is NP
  - in “think about how we are doing”, “think” is VP, “about” is NP, “how we are doing” is another clause
  - in “He sleeps as the rain falls”, “as” is NP
  - in “I’m as tall as he is”, “as tall” is NP, “as” is NP

COORD

- **COORD consist of several NP (including conjuctions or comma)**
  - ”lead, mercury, cadmium, hexavalent chromium, and PBB” is COORD
  - remember inclusive or: in ”it allows researchers or clinicians”, ”researchers or clinicians” is COORD
  - ”learning and mental problems” IS NOT (!) COORD – word ”problem” is control member and ”learning” and ”mental” are developing members
- **if there is a preposition at the begining of multiple constituent, it is not COORD (a head of the phrase is the proposition)**

  - ”kinds of information and services” is not COORD
- **punctuation (or other symbols like asterisks, dashes, brackets, quotation marks etc.) cannot be at the begining or at the end of COORD**

VP

- **VP contains only verb forms**
- **one VP contains only one non-auxiliary verb =¿ there can be more VP in a phrase**
  - ”go swimming” – ”go” and ”swimming” are two VP
  - ”appears to be” – ”appears” and ”to be” are two VP
  - auxiliary verbs (be, do, have and their forms: am, is, being, had etc.) + verb are ONE VP
    - ”have explained”, “did not want”, “was working”, “is finished” are one VP
  - modal verbs (can, could, may, might, must, shall, should, would, will) + verb are ONE VP
    - ”could use them to make” – ”could use” is VP, ”them” is NP, ”to make” is VP
  - modal verbs with ”to”, e.g. have to, need to, aren’t considered modal verbs themselves: in “have to see”, “have” and “to see” are VP
- **the particle ”to” is a part of VP (in the infinitive verb form)**
  - ”to have” is one VP
- **”not” is part of VP**
  - ”did not want” is one VP
- **passive constructions are one VP**
  - ”is also related to linguistics” – ”is related” is VP, ”also” and ”to linguistics” are two NP
  - ”is called” is VP
  - ”it was first sold” – ”was sold” is VP
- **it is necessary to separate multiple verbs**
  - ”can take and show” – ”can take” and ”show” are two VP
- when the verb is followed by a preposition (and therefore a NP after it), the preposition is a part of NP, unless it is a phrasal verb according to `www.usingenglish.com/reference/phrasal-verbs` (+ fuck off, fuck up, piss off, ... :))

- in “what you are thinking of”, “are thinking” is VP, “of what” is NP
- in “(she) is looking after a baby”, “is looking after” is VP, “a baby” is NP
- in “(I) will have stood by John”, “will have stood by” is VP, “John” is NP
- in “(she) broke down”, “broke down” is VP
- in “(you) should think it over”, “should think over” is VP
- in “(I) am looking forward to the event”, “am looking forward to” is VP, “the event” is NP
- in “(they) had been sitting in for a neighbor”, “had been sitting in for” is VP, “a neighbor” is NP

CLAUSE

- **is a simple sentence (containing one finite verb), or a phrase, if there is no verb present**
  - ”the baby is crying loudly” is one clause
  - ”visual illusions can be caused by imagination” is one clause
  - ”The Curse of Mental Accounting” is one clause (it is a title)
- **one clause can contain more than one VP (several non-finite verbs)**
  - ”someone tried to refuse to accept the offer” is one clause
  - ”the history of the film turns out to be very interesting” is one clause
- **only the longest possible clause is correct**
  - ”to be about 66 meters per second” is NOT a clause (part of the clause is missing)
  - ”is in the december edition of The Psychologist” is NOT a clause (subject is missing)
- **punctuation (or other symbols like asterisks, dashes, brackets, quotation marks etc.) cannot be at the beginning or at the end of a clause**
  - 3), 15), §264,... are not clauses
- **conjunctions (and, if, but, as, or, because,...) cannot be at the beginning of a clause**
  - ”the water was warm, but I didn’t go swimming” – ”the water was warm” and ”I didn’t go swimming” are clauses
  - ”I see that you arrived” – ”I see” and ”you arrived” are clauses (that is a conjunction)

- **BUT relative pronouns and relative adverbs (which, who, that, what, when, where, why, whatever, wherever, ...) are part of a clause**
  - ″this is the boy who kissed her″ – ″this is the boy″ and ″who kissed her″ are clauses
  - ″I ate the food that I made″ – ″I ate the food″ and ″that I made″ are clauses (that is a relative pronoun)
- **the content in brackets is a separate clause, if there is a finite verb, otherwise it is a part of one clause**
  - ″acquired immunodeficiency syndrome (AIDS) is a disease of the human immune system″ is one clause
  - ″I was reading a book (it was an exciting story) in my living room″ – ″I was reading a book″ and ″it was an exciting story″ are clauses
- **a clause can be embedded into another clause**
  - ″there is a table, obviously made to appear old, and a chair″ – ″there is a table and a chair″ is the correct clause

RELATIONS

- **only the longest possible NP can be dependent on a superior NP or on VP**
  - ″man with a telescope″ – ″with a telescope″ → ″noun: man″
  - ″the beautiful young princess woke up″ – ″the beautiful young princess″ → ″verb: woke up″
- **if NP is not dependent on anything, use the option ″not identified yet″**
- **if NP is part of COORD, NP is dependent on the COORD and the whole COORD is dependent on relevant VP**
  - ″dog and cat decided to bake a cake″
  - ″dog and cat″ → ″verb: decided″
  - ″dog″ → ″noun: dog and cat″
  - ″cat″ → ″noun: dog and cat″
- **wrong phrases can't be related to anything and nothing can be dependent on them**
- **if NP can be dependent on more phrases, choose the one most suitable according to your opinion**
- **NP is dependent on VP which is in the same clause (choose the relevant VP, if there are more VPs in a clause)**

Appendix C

List of author's publications

C.1 Peer reviewed journal papers

1. Irena Srdanović, Naomi Ida, Chikako Shigemori Bučar, Adam Kilgarriff, and Vojtěch Kovář. Japanese word sketches: Advances and problems. *Acta Linguistica Asiatica*, 1(2):63–82, 2011
   [author's contribution: 15% – annotation software, automatic evaluations, partly design of the experiment]

C.2 Book chapters

1. Vojtěch Kovář, Miloš Jakubíček, and Jan Bušta. Czech vulgarisms in text corpora. In *After Half a Century of Slavonic Natural Language Processing*, pages 141–145, Brno, 2009. Tribun EU
   [author's contribution: 80% – majority of data collection, analysis and text writing]

C.3 Peer reviewed conference papers

1. Vojtěch Kovář, Vladimír Kadlec, and Aleš Horák. Grammar development for Czech syntactic parser with corpus-based techniques. In *Proceedings of Corpus Linguistic 2006*, pages 159–165, Saint-Petersburg, 2006. Saint-Petersburg State University
   [author's contribution: 60% – design and implementation of the central experiment]
2. Vojtěch Kovář and Aleš Horák. Reducing the number of resulting parsing trees for the Czech language using the beautified chart method. In *Proceedings of the 3rd Language & Technology Conference*, pages 433–437, Poznań, Poland, 2007. Wydawnictwo Poznańskie
   [author's contribution: 70% – implementation of the method, design and implementation of the experiments, majority of text writing]

3. Vojtěch Kovář and Aleš Horák. Power networks dialogues - automatic analysis and evaluation of a domain-specific text corpus. In *Proceedings of ELNET 2007*, pages 30–37, Ostrava, 2007. Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava
[author's contribution: 60% – implementation of the experiments, part of text writing]

4. Pavel Rychlý and Vojtěch Kovář. Displaying bidirectional text concordances in KWIC format. In *Proceedings of 5th Biennial Conference of the Asian Association for Lexicography*, pages 61–66, Madras, India, 2007. University of Madras
[author's contribution: 40% – part of implementation, majority of text writing]

5. Aleš Horák, Tomáš Holan, Vladimír Kadlec, and Vojtěch Kovář. Dependency and phrasal parsers of the Czech language: A comparison. In *Proceedings of Text, Speech and Dialogue, 10th International Conference*, volume 4629 of *Lecture Notes in Computer Science*, pages 76–84, Berlin, 2007. Springer
[author's contribution: 20% – design and implementation of one part of the experiment, minor part of text writing]

6. Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Power networks dialogs - enhancing domain-specific text processing techniques and resources. In *Proceedings of ELNET 2008*, pages 72–80, Ostrava, 2008. Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava
[author's contribution: 50% – implementation of experiments, part of text writing]

7. Vojtěch Kovář, Aleš Horák, and Vladimír Kadlec. New methods for pruning and ordering of syntax parsing trees. In *Proceedings of Text, Speech and Dialogue, 11th International Conference*, volume 5246 of *Lecture Notes in Computer Science*, pages 125–131, Berlin, 2008. Springer
[author's contribution: 50% – implementation and partly design of the experiments, part of text writing]

8. Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Syntactic analysis as pattern matching: The SET parsing system. In *Proceedings of 4th Language & Technology Conference*, pages 978–983, Poznań, Poland, 2009. Wydawnictwo Poznańskie

[author's contribution: 90% – system design, majority of implementation and text writing]

9. Miloš Jakubíček, Aleš Horák, and Vojtěch Kovář. Mining phrases from syntactic analysis. In *Proceedings of Text, Speech and Dialogue, 12th International Conference*, volume 5729 of *Lecture Notes in Computer Science*, pages 124–130, Berlin, 2009. Springer
[author's contribution: 10% – minor contributions to design and implementation of the method]

10. Abhilash Inumella, Adam Kilgarriff, and Vojtěch Kovář. Associating collocations with dictionary senses. In *Proceedings of 6th Biennial Conference of the Asian Association for Lexicography*, pages 102–113, Bangkok, Thailand, 2009. Asian Association for Lexicography
[author's contribution: 20% – partial contributions to design and implementation of presented methods]

11. Adam Kilgarriff, Vojtěch Kovář, and Pavel Rychlý. Tickbox lexicography. In *eLexicography in the 21st century: New challenges, new applications*, pages 411–418, Louvain la Neuve, Belgium, 2010. Presses universitaires de Louvain
[author's contribution: 40% – implementation and partly design of the method]

12. Marek Grác, Miloš Jakubíček, and Vojtěch Kovář. Through low-cost annotation to reliable parsing evaluation. In *PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 555–562, Sendai, Japan, 2010. Tohoku University
[author's contribution: 33% – parser development, partly methodology design and text writing]

13. A. Kilgarriff, V. Kovář, S. Krek, I. Srdanović, and C. Tiberius. A quantitative evaluation of Word Sketches. In *Proceedings of the XIV Euralex International Congress*, pages 372–379, Ljouwert, Netherlands, 2010. Fryske Akademy
[author's contribution: 30% – implementation and partly design of the experiment]

14. Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Syntactic analysis using finite patterns: A new parsing system for Czech. In *Human Language Technology. Challenges for Computer Science and Linguistics*, volume 6562 of *Lecture Notes in Computer Science*, pages 161–171, Berlin, 2011. Springer

[author's contribution: 90% – system design, majority of implementation and text writing]

15. Vít Baisa and Vojtěch Kovář. Information extraction for Czech based on syntactic analysis. In *Proceedings of the 5th Language & Technology Conference*, pages 466–470, Poznań, Poland, 2011. Funcacja Universytetu im. A. Mickiewicza
[author's contribution: 50% – system design and partly implementation, evaluation, part of text writing]

16. Adam Kilgarriff, Pavel Rychlý, Vojtěch Kovář, and Vít Baisa. Finding multiwords of more than two words. In *Proceedings of the 15th EURALEX International Congress*, pages 693–700, Oslo, 2012. University of Oslo
[author's contribution: 20% – implementation and partly design of one of the methods]

17. Iztok Kosem, Vít Baisa, Vojtěch Kovář, and Adam Kilgarriff. User-friendly interface of error/correction-annotated corpus for both teachers and researchers. In *Book of Abstracts LCR 2013*, pages 82–84, Bergen, 2013. University of Bergen
[author's contribution: 25% – contributions to implementation and interface design]

18. Marek Medved', Miloš Jakubíček, and Vojtěch Kovář. Towards taggers and parsers for Slovak. In *Proceedings of 6th Language & Technology Conference*, pages 1–4, Poznań, Poland, 2013. Wydawnictwo Poznańskie
[author's contribution: 10% – contributions to grammar engineering]

19. Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlý, and Vít Suchomel. The TenTen corpus family. In *7th International Corpus Linguistics Conference CL 2013*, pages 125–127, Lancaster, 2013. Lancaster University
[author's contribution: 15% – contributions to methodology and data analysis]

20. Miloš Jakubíček and Vojtěch Kovář. Enhancing Czech parsing with verb valency frames. In *Computational Linguistics and Intelligent Text Processing - 14th International Conference, CICLing 2013*, volume 7816 of *Lecture Notes in Computer Science*, pages 282–293, Greece, 2013. Springer
[author's contribution: 25% – valency data processing, partly text writing]

C.4 Other papers

1. Vojtěch Kovář. Corpus query system Bonito - recent development. In *Proceedings of First Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 71–76, Brno, 2007. Masaryk University
2. Vojtěch Kovář and Miloš Jakubíček. Test suite for the Czech parser Synt. In *Proceedings of Second Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 63–70, Brno, 2008. Masaryk University
   [author's contribution: 60% – majority of design and implementation, part of text writing]
3. Vojtěch Kovář and Miloš Jakubíček. Prague dependency treebank annotation errors: A preliminary analysis. In *Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 101–108, Brno, 2009. Masaryk University
   [author's contribution: 60% – majority of both data analysis and text writing]
4. Miloš Jakubíček, Vojtěch Kovář, and Aleš Horák. Measuring coverage of a valency lexicon using full syntactic analysis. In *Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 75–79, Brno, 2009. Masaryk University
   [author's contribution: 10% – minor contributions to design and implementation of the experiment]
5. Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. How to analyze natural language with transparent intensional logic? In *Proceedings of Fourth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 69–76, Brno, 2010. Masaryk University
   [author's contribution: 50% – substantial contribution to methodology design, implementation and text writing]
6. Miloš Jakubíček and Vojtěch Kovář. CzechParl: Corpus of stenographic protocols from Czech parliament. In *Proceedings of Fourth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 41–46, Brno, 2010. Masaryk University
   [author's contribution: 10% – minor contribution to data analysis]
7. Miloš Jakubíček, Vojtěch Kovář, and Pavel Šmerk. Czech morphological tagset revisited. In *Proceedings of Fifth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 29–42, Brno, 2011. Tribun EU

[author's contribution: 33% – contributions to analysis, modification proposals and text writing]

8. Aleš Horák, Miloš Jakubíček, and Vojtěch Kovář. Analyzing time-related clauses in transparent intensional logic. In *Proceedings of Fifth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 3–9, Brno, 2011. Tribun EU
   [author's contribution: 20% – contributions to implementation and text writing]
9. Aleš Horák, Miloš Jakubíček, and Vojtěch Kovář. Linguistic logical analysis of direct speech. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 51–59, Brno, 2012. Tribun EU
   [author's contribution: 20% – contributions to implementation and text writing]
10. Jan Rygl, Kristýna Zemková, and Vojtěch Kovář. Authorship verification based on syntax features. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 111–119, Brno, 2012. Tribun EU
    [author's contribution: 20% – contributions to method design and text writing]
11. Marek Medved', Miloš Jakubíček, Vojtěch Kovář, and Václav Němčík. Adaptation of Czech parsers for Slovak. In *Proceedings of Sixth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 23–30, Brno, 2012. Tribun EU
    [author's contribution: 10% – contributions to grammar engineering]
12. Ondřej Herman and Vojtěch Kovář. Methods for detection of word usage over time. In *Proceedings of Seventh Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 79–85, Brno, 2013. Tribun EU
    [author's contribution: 15% – contributions to method design]

C.5 Software

1. Jaroslav Moravec, Vojtěch Kovář, Jan Bušta, and Miloš Jakubíček. OOCorr, 2010. `nlp.fi.muni.cz/projects/oocorr`
   [author's contribution: 15% – minor contributions to design and implementation]

2. Vojtěch Kovář, Miloš Jakubíček, and Aleš Horák. Syntactic parser SET, 2012. `nlp.fi.muni.cz/projects/set`
   [author's contribution: 90% – system design, majority of implementation]