MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Rule-Based Parsing of Morphologically Rich Languages

PH.D. THESIS

Miloš Jakubíček

Brno, 2017

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 thank all the people around me that allowed me to work on this thesis and helped me to finish it.

Ten years ago, my decision to start learning about natural language processing and my subsequent career orientation was largely influenced by the warm welcome to the Natural Language Processing Centre by its members, in the first place my current advisor Aleš Horák, my former advisor and head of the centre Karel Pala, my colleague Pavel Rychlý, as well as many other team members.

I would only wish to be able to thank to Adam Kilgarriff, who died in 2015 and who I have learnt a lot from during my work with the Sketch Engine team at Lexical Computing, the company he has founded. Adam has open many doors for me and has significantly shaped my research views. I do miss him a lot.

I would like to thank all my team mates at Lexical Computing for bearing with me during the time I spent on writing the thesis.

I would like to thank my wife Baška for taking the best care of our son during that time as well as my dear friend Terka who never stopped pushing me to finish.

Special thanks go to Aleš Horák, Vojta Kovář and Pavel Rychlý for their useful comments and discussions on this thesis, as well as to Vojtěch Řehák, my internal reviewer at the Masaryk University, for his significant contribution to making the text more comprehensible.

I was always confident that large portion of what I do sustains of building on what others have done before, picking up the right pieces and throwing away the bad ones, yet they were to be discovered. Thank you all.

Abstract

In this thesis I discuss parsing of morphologically-rich languages, particularly Czech. I start with a rather critical review of the state-of-the-art in terms of the dominant paradigm (statistical parsing), its general applicability and usability as well as mainstream evaluation practices. I put the discussion into a historical context which, I believe, is a key for its understanding.

I argue for more flexible and more practical approaches to parsing that might have a rule-based backbone. I argue for theory-agnostic extrinsic evaluation to be used in parsing to increase its practical usability.

I show two particular approaches and implemented systems targeting different tasks that I have been working on. The first focuses on large-scale parsing with the aims of extracting linguistic knowledge from large amounts of texts, and is evaluated on a collocation extraction task for Czech. The other focuses on parsing of isolated sentences and is evaluated on its contribution to part-of-speech tagging of Czech.

I discuss the theoretical background as well as practical implementation and scalability of both systems and their suitability for parsing morphologically-rich languages.

Keywords

parsing, syntactic analysis, syntax, grammar, morphologically-rich language, word sketch, Synt, natural language processing, computational linguistics

Contents

1 **Introduction** 1
1.1 *Syntax in Natural Language Processing: an Overview* 1
1.2 *A Historical Account* 2
1.2.1 Syntax and Parsing: the Theory and the Practice . . . 2
1.2.2 From Formal to Natural – and back 4
1.2.3 . . . and does it matter? 6
2 **Syntax** 8
2.1 *Overview* 8
2.2 *Phrase structure syntax* 8
2.3 *Dependency syntax* 10
2.4 *Phrase structure vs. dependency syntax* 11
2.4.1 Discontinuity and non-projectivity 12
2.4.2 Coordinating dependencies 13
2.4.3 Phrase Structures vs. Dependencies: Conclusions . . 13
2.5 *Advanced Syntactic Formalisms* 15
2.5.1 Generalized and Head-Driven Phrase Structure Grammar 15
2.5.2 Lexical Functional Grammar 16
2.5.3 Tree-Adjoining Grammar 17
2.5.4 Combinatory Categorial Grammar 18
2.5.5 Link Grammar 19
2.5.6 Conclusions 19
3 **Parsing** 21
3.1 *Overview* 21
3.2 *Rule-Based Parsing* 22
3.3 *Statistical Parsing* 23
3.4 *The Purpose of Parsing* 23
3.5 *Parsing Evaluation* 26
3.5.1 Intrinsic evaluation 26
3.5.2 Agreement among linguists 27
3.5.3 Agreement among annotators 28
3.5.4 Comparing trees: similarity metrics 31

3.5.5 Extrinsic evaluation . . . 31
3.6 *Criticism of Statistical Parsing* . . . 32
3.7 *On Writing Rules* . . . 33
3.8 *Parsing Systems* . . . 35
3.8.1 English and multi-language parsers . . . 35
3.8.2 Czech parsers . . . 37
3.8.2.1 Dependency parsers . . . 37
3.8.2.2 Phrase structure parsers . . . 37
3.9 *Parsing of morphologically-rich languages* . . . 38
3.10 *Parsing large vs. parsing small* . . . 39
4 **Parsing large: word sketches** . . . 41
4.1 *Overview* . . . 41
4.2 *The concept of word sketches* . . . 41
4.3 *CQL* . . . 44
4.4 *Sketch grammar formalism* . . . 45
4.5 *Handling rich-morphology* . . . 47
4.6 *Implementation* . . . 48
4.6.1 Manatee . . . 48
4.6.2 Generating Word Sketches . . . 51
4.7 *Scalability: Parallelization* . . . 52
4.7.1 Corpus compilation . . . 52
4.7.1.1 Corpus virtualization . . . 52
4.7.1.2 Parallelization through virtualization . . . 54
4.7.2 Word sketch compilation . . . 55
4.8 *Scalability: Optimization of regular expressions* . . . 58
4.8.1 Basic strategies . . . 58
4.8.2 n-gram prefetching . . . 59
4.8.3 n-gram indexing . . . 60
4.9 *Scalability: Multilinguality* . . . 65
4.10 *Word Sketch Accuracy Evaluation* . . . 67
4.10.1 2010: precision-only . . . 67
4.10.2 2012: precision and recall . . . 67
4.10.3 2016: precision-only . . . 69
5 **Parsing small: Synt** . . . 72
5.1 *Overview* . . . 72
5.2 *The Synt parser* . . . 72
5.3 *Grammar and Metagrammar* . . . 74
5.4 *Phrase structure grammar with dependency marking* . . . 76
5.5 *Chart and Forest of Values* . . . 76
5.6 *Handling rich-morphology* . . . 77

5.7 *Implementation* . . . 78
5.8 *Scalability: Multilinguality* . . . 80
5.9 *Facing ambiguity in parsing* . . . 81
5.9.1 Rule probabilities: Learning Probabilities From Treebanks . . . 81
5.9.2 Rule levels: Grammar Stratification Exploiting Competing PCFG Rules . . . 81
5.9.2.1 Chart-based Local Pruning . . . 82
5.9.2.2 Forest-based Non-local Pruning . . . 83
5.10 *Evaluation* . . . 84
5.10.1 Brno Phrasal Treebank . . . 85
5.10.2 Part-of-speech tagging . . . 86
5.10.2.1 Partial morphological disambiguation . . . 86
5.10.2.2 Comparison with part-of-speech taggers . . . 89
6 **Conclusions** . . . 98
Appendices . . . 100
A **List of author's publications** . . . 101
A.1 *Journal papers* . . . 101
A.2 *Book chapters* . . . 101
A.3 *Edited proceedings* . . . 102
A.4 *Conference papers (peer-reviewed)* . . . 102
A.5 *Conference papers (non-peer-reviewed)* . . . 106
A.6 *Other publications* . . . 109
Bibliography . . . 111

Chapter 1

Introduction

1.1 Syntax in Natural Language Processing: an Overview

This thesis focuses on syntactic analysis (parsing), a research area within the scope of natural language processing (NLP), a well-established research domain at the interface between computer science and linguistics dealing with natural languages. It bears various other names such as computational linguistics, human language processing and others, all of which are used in the following text interchangeably, referring to a field that aims at narrowing the still so wide gulf between natural language and computer systems.

The bold goals of NLP can be structured from two different angles: from a rather theoretical one, consisting of formal understanding and modelling of natural language, exhibited as formal (and therefore automatable) analysis and synthesis, or from a practical one that addresses particular tasks and applications of NLP.

The theoretical structure based on analysis and synthesis usually follows traditional linguistic view of language as a communication mechanism that can be structured into different hierarchical levels, such as phonetics, phonology, morphology, syntax, semantics and pragmatics. Aside of these levels NLP contributes to research in language acquisition, knowledge reasoning and description of other language-based cognitive processes. While the classification of linguistic levels certainly represents a useful concept for approaching formal description of a language, builds on some elementary differences of language use (i.e. differentiating between spoken and written language, understanding how words are formed and combined into sentences etc.) and coins fundamental concepts of linguistics (such as a word or a sentence), it does not give a straightforward recipe for language analysis. All efforts on formalizing these linguistic levels show that their interplay is very complex, their borders are very thin and it is extremely difficult to establish formal interfaces between the isolated levels. For these and other, more practical reasons (such as propagation of errors from the analysis of a lower level into the analysis of an upper level) one can see a clear trend to

overcome these difficulties using methods that apply cross-level analysis or synthesis.[1]

The practical account of NLP is to see the field structured by all task-oriented applications such as information extraction and retrieval, machine translation, question answering, computer lexicography etc. Theoretically these tasks always involve processing at all linguistic levels, in practice this is often limited just to few of them.

Where does syntax and syntactic analysis fit into this view? In theory it is a part of any tool and any language processing that goes beyond morphology (word forms). In practice this is however not true most of the time, for reasons such as low accuracy and high complexity of many parsing solutions. There are many successful NLP applications (e. g. in the area of information retrieval or machine translation) that use zero or only very limited syntactic processing. Many other tools are intentionally bypassing any syntactic processing, and rather use simple rule-based or statistic methods that rely on morphology only. Nevertheless syntactic information *might* be of great use for a large variety of NLP applications.

In this thesis I propose solutions that focus on syntactic processing of morphologically rich inflective languages, especially on Czech. I describe two systems and argue why the proposed solutions are suitable for these languages and why the dominant paradigm of (purely) statistical parsing does not contribute to larger exploitations of parsers in practical applications. I discuss both theoretical background and practical implementation of both parsing solutions together with evaluation on practical tasks.

1.2 A Historical Account

1.2.1 Syntax and Parsing: the Theory and the Practice

In linguistics syntax is usually defined as a linguistic discipline studying relationship among words in a sentence and recovering the sentence structure. The Oxford Dictionary of English defines syntax as "The arrangement of words and phrases to create well-formed sentences in a language."

The notion of well-formedness (*grammaticality*) is somewhat problematic in the field. Clearly non well-formed sentences do exist, but the judgement on whether a sentence is well-formed or not is not always an easy one, and there is limited agreement both among lay native speakers and professional linguists (as noted already in [Hill, 1961]). The question what is the syn-

1. In case of parsing, a recent work on English parsing without using morphological information is e.g. [Vinyals et al., 2015].

tactic structure of a sentence, and whether there actually is such a mental structure used by humans remains still open and subject to research in psycholinguistics and neurolinguistics. Experiments with reading times and eye-tracking show that human processing of sentences has probabilistic nature, and when reading sentences like "Man held at the station died" there is something like a re-analysis occurring at the point reaching the (unlikely) end of the sentence with the word *died* (a good overview is provided e.g. in [Traxler et al., 2002])

The syntax and parsing stand to each other like the theory to practice. In the past hundred years linguists have intensively studied formal properties of natural language syntax (with Noam Chomsky [Chomsky, 1957] being with no doubts one of the most influential syntacticians): how words combine to phrases, phrases to clauses and clauses finally build up a sentence. How morphology is propagated into syntax and what is the interface between syntax and semantics. How far syntactic properties can be universal among natural languages. Independently the advances in computer technologies brought practical parsing techniques, focusing primarily on formal languages – to parse communication protocols and programming languages.

However, there were soon attempts (e.g. [Kay, 1967, Earley, 1970]) to use the same or similar formalisms also for natural languages, whose inbuilt ambiguity required substantially different treatment and algorithms than those used for formal languages which strived for efficient lookahead methods to overcome local ambiguities.

Crocker classifies[2] strategies to deal with syntactic ambiguity (technically exhibited as the potential of a single sentence to yield multiple parses) in parsing as follows:

1. deterministic parsing
   Multiple parses are ignored: the decision process makes greedily local choices and end up with a single interpretation.

2. underspecification
   Syntactic phaenomena causing ambiguity (such as PP-attachment of verb valencies) are underspecified, i.e. undecided. This necessarily requires a representation that accounts for underspecification.

3. backtracking
   Multiple parses are retrieved subsequently using a backtracking

2. e.g. in http://www.coli.uni-saarland.de/~crocker/courses/comp_psych/documents/Lecture3_000.pdf

strategy in the decision-making process of parsing. Not surprisingly, this approach used to be tightly bound with prototypes implemented in Prolog with limited focus on efficiency.

4. parallelization
Multiple parses are retrieved concurrently during the analysis, usually by applying various dynamic programming techniques, yielding an ambiguous syntactic structure, typically a graph called *chart*, after which this approach is often called *chard-based parsing*. The ambiguous structure may be subject to scoring (by assigning probabilities, or generally weights) of individual parses for selecting the best parse afterwards.

Indeed I believe these are the only four (abstract) options for handling syntactic ambiguity. Many early parsing systems used to use backtracking, then later these evolved into parallel chart-based systems (rather evolutionary than revolutionary, and merely to overcome the time complexity of backtracking) and later were overcome by many purely statistical systems approaching parsing as a machine learning task, in many cases (especially those based on neural networks) leaving no space for global ambiguity and basically performing deterministic parsing.

I also think that it is very regrettable that underspecification was always considered as something close to losing the battle on syntax, rather than a humble and practical step away from issues that one cannot resolve well enough (yet).

1.2.2 From Formal to Natural – and back

With no exaggeration one can say that the whole field was largely affected by attempts to apply computer science methods developed for formal languages to parsing of natural languages too, and vice versa by linguists trying to match natural languages with their formal counterparts.

It may be that this was largely caused by Chomsky being such an influential character for both theory of formal languages in computer science[3] and studying syntax in linguistics; but by all means it was not without mutual

3. It is, from today's perspective, surprising to see such a close connection between computer science and linguistics at that time. Knuth admits that: "I found the mathematical approach to grammar immediately appealing—so much so, in fact, that I must admit to taking a copy of Noam Chomsky's Syntactic Structures along with me on my honeymoon in 1961. [...] Here was a marvelous thing: a mathematical theory of language in which I could use a computer programmer's intuition!" [Knuth et al., 2003]

misunderstandings such as those around the discussion on which category of the Chomsky hierarchy of languages do natural languages (one or all) fall in. In the Chomsky hierarchy, natural languages have been mostly assigned to the class of context-free languages, to be described by a context-free grammar (CFG). The question whether CFGs have enough expressive power to describe natural languages has been often disputed in linguistic circles (mainly with regard to English) in the 70's and 80's, and it was claimed by many that it is not the case, mostly without any empirical evidence or mathematical proofs. Many of these assumptions were based merely on the misunderstanding of the phrase "context-free" not as a formally defined computer science concept, but simply "context disregarding" treatment of natural language syntax.

This has caused extreme claims:[4]

"

the girl saw the boy
$*$ *the girl kiss the boy*

this well-known syntactic phenomenon demonstrates clearly the inadequacy of ... context-free phrase-structure grammars..." [Grinder and Elgin, 1973]

"These examples show that to describe the facts of English number agreement is literally impossible using a simple agreement rule of the type given in a phrase-structure grammar, since we cannot guarantee that the noun phrase that determines the agreement will precede (or even be immediately adjacent) to the present-tense verb." [Bach, 1974]

"In general, for any phrase structure grammar containing a finite number of rules like (2.5), (2.52) and (2.54) it will always be possible to construct a sentence that the grammar will not generate. In fact, because of recursion there will always be an infinite number of such sentences. Hence, the phrase structure analysis will not be sufficient to generate English." [Culicover, 1976]

"in many cases the number of a verb agrees with that of a noun phrase at some distance from it... this type of syntactic dependency can extend as far as memory or patience permits... the distant type of agreement... cannot be adequately described even by context-sensitive phrase-structure rules, for the possible context is not correctly describable as a finite string of phrases." [Bresnan, 1977]

4. The author cordially thanks Shuly Wintner for providing all these historical examples; as well as for contributing to the fact that this historical momentum does not remain forgotten.

"Since there seem to be no way of using such PS rules to represent an obviously significant generalization about one language, namely, English, we can be sure that phrase structure grammars cannot possibly represent all the significant aspects of language structure. We must introduce a new kind of rule that will permit us to do so." [Akmajian and Heny, 1980]

1.2.3 . . . and does it matter?

Indeed it took quite a few years until linguists with sufficient mathematical background like Gerald Gazdar disproved these claims and provided an excellent over view in [Pullum and Gazdar, 1982]. Years later Gazdar was reviewing the status quo in linguistics at that time as follows:

"(we) . . . became aware that the arguments relating to CFG and well-formedness, arguments which were around then and which had been accepted for ten or fifteen years almost without question, were basically junk, . . .

. . . When Geoff and I started reviewing the arguments against the context-freeness of natural languages, we weren't working in a field that was full of people who thought the existing arguments were invalid. On the contrary – most linguists were teaching their students that the facts of English subject-verb agreement showed that English was a context-sensitive language. " [Briscoe and Gazdar, 2001]

Despite it has been later shown that indeed a few phenomena in some particular languages (e. g. Swiss German[5], see [Shieber, 1985]) exceed the expressivity of CFG even in terms of weak generative capacity (i. e. string equivalence) and the fact that *contextual phenomena* (which have absolutely nothing to do with the exact definition of context-free languages, of course) play generally an important role in natural language processing, CFG's remain at the core of many grammar-based systems, often extended to Probabilistic CFG (PCFG). This has well-known practical reasons, such as parsing time in $\mathcal{O}(n^3)$ (see e. g. [Younger, 1967]).

In fact, whether a natural language, or all of them, fit into a particular class of languages of the Chomsky (or any other) hierarchy, is from a practical point of view of natural language processing far less (if at all) important than one might think (apart of the strong theoretical assumption that one is taking here, that there is something like a formal grammar of a language *per sē*). First of all, one has to strictly differentiate between the weak and strong equivalence of two grammars. In case of the weak equivalence, two grammar are equivalent if they generate the same language (of strings, or

5. Which is a dialect not to be confused with Swiss Standard German.

rather sentences). In the case of strong equivalence, a given sentence also needs to have the same derivation according to both grammars.

In terms of weak equivalence, one can do fairly well even with a finite-state (regular) grammar – which is practically mirrored by the fact that major portions of NLP are with big success being based merely on regular expressions over plain or annotated text. But talking about syntax, the weak equivalence would not be very useful: the syntax is all about describing complex structures and relationship among words, by no means satisfiable with binary right-branching of regular trees. However, in terms of strong equivalence, the answer to the question of belonging to a context-free or any other class strictly depends on a particular syntactic theory that one follows. It is trivial to come up with a very complex theory implying trans-context-free structure, but practical matters ask for the opposite: how can syntax of a natural language be simplified so as to be able to formalize it consistently, and process it easily?

Chapter 2

Syntax

2.1 Overview

The two main flavours of looking at the description of natural language syntax differ in whether central role is put on clustering of the sentence in smaller chunks of words (constituents, non-terminals, phrases) in case of the *phrase structure* syntax or whether it is put on discovering relationships between individual words in case of the *dependency* syntax.

I take closer look at both in the following sections, with a brief mention of the more complex theories that were built on top of them. The primary viewpoint are implications of these theories on parsing and its evaluation, witch are then discussed in Chapter 3.

2.2 Phrase structure syntax

The modern era of phrase structure syntax has been largely dominated by the influence of Noam Chomsky, starting with his book Syntactic Structures [Chomsky, 1957]. Chomsky laid down many new principles for phrase structure syntax which he further developed into the theory of Transformational Grammar (and later, Generative Grammar and Minimalist Program). Lots of his efforts with regard to syntax were actually devoted to the hypothesis of universal grammar: Chomsky was looking for syntactic principles that would be universal and innate [Chomsky, 1995].

Even though his theories have been later criticised [Johnson and Lappin, 1997, Sampson, 2001, Sampson, 2005] as missing empirical evidence, and may be considered overcome to large extent, he was one of the most influential syntacticians of the 20th century and his theories still represent a mainstream in linguistics, despite the opposition getting stronger, and many people not hesitating to express sharp opinions on Chomsky:

"...The field (linguistics) has clearly been damaged by the presence of a charismatic leader who has led it badly. ..." [Briscoe and Gazdar, 2001]

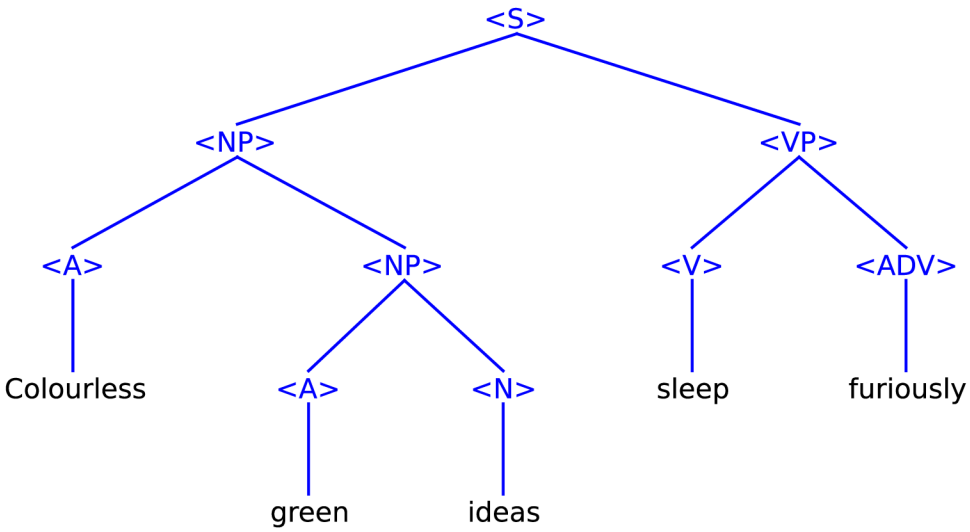


Figure 2.1: Example phrase structure tree.

There were many research efforts in phrase structure syntax apart of (or following up on) Chomsky, leading to advanced syntactic formalisms like GPSG, CCG, HPSG, TAG or LFG, which are briefly described in further text.

There is no doubt that phrase structure syntax, as a syntactic model based on the principle of constituency, independent of any particular theory, is by all means a very useful concept, both for linguistics and natural language processing. What I consider uncertain (both particularly for phrase structure syntax and generally for any syntactic theory) is whether the dominant default form of the result of syntactic analysis – a syntactic tree, which in case of phrase structure syntax often corresponds to the derivation tree according to the underlying grammar – is indeed the best default from the perspective of usability, difficulties in evaluation as for comparing trees or intelligibility by end users.

In Figure 2.1 I provide a derivation tree for Chomsky's famous sentence *Colourless green ideas sleep furiously* which Chomsky used as an argument for the separation of syntax from semantics. A slight modification to the sentence reveals one of the many structural ambiguities that need to be resolved during the parsing process: if it were *Green ideas without colour*, which of the representations provided in Figure 2.2 is the correct one? While this will be obvious in most cases, there remain many where any single right decision is hard if not impossible to justify objectively.[1]

1. Stageberg gave already in 1968 an overview of 73 categories of English noun phrases

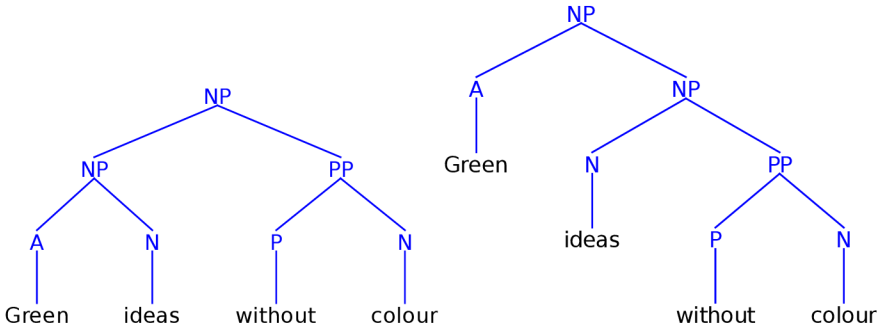


Figure 2.2: Noun-phrase ambiguity illustrated on the examples of the phrase *Green ideas without colour*.

This applies also to the perhaps most famous ambiguity in phrase structure syntax, namely the PP-attachment problem. Notoriously well known examples like *I saw a man with a telescope* or *Charles talked about sex with Britney Spears* showing PP-attachment instances requiring extra-sentential context are followed with many where the main obstacle lies elsewhere:

"The core difficulty in this enterprise is to define the notion of argument precisely enough that it can be used automatically. There is a consensus in linguistics that arguments and adjuncts are different both with respect to their function in the sentence and in the way they themselves are interpreted [...] Unfortunately, the linguistic diagnostics that are used to determine whether a PP is an adjunct or an argument are not accurate in all circumstances, they often partition the set of the examples differently, and they give rise to relative, and not absolute, acceptability judgments" [Merlo and Ferrer, 2006, p. 4]

2.3 Dependency syntax

On contrary to the phrase structure syntax which places adjacency (constituency) as the dominant syntactic principle, dependency syntax relies on the notion (as the name suggests) of a dependency, a binary (and, usually, asymmetric and anti-transitive) relation between words. The work on dependency syntax is historically attributed to Tesnière [Tesnière, 1959] and has been further independently developed by multiple linguistic groups, resulting into a range of dependency syntax theories, such as Hudson's Word Grammar [Hudson, 1984], Meaning-text theory (MTT) developed by Mel'čuk [Mel'čuk, 1988], and for Czech the work done by Šmilauer [Šmilauer, 1969] and later partially adopted by the Prague linguistic cir-

leading to structural ambiguities (see [Stageberg, 1968]).

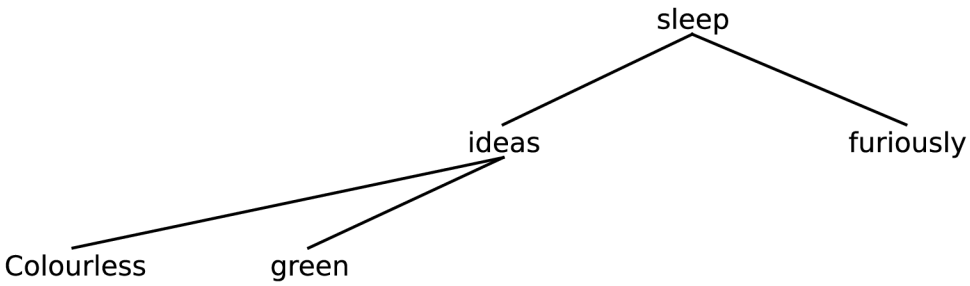


Figure 2.3: Example unlabeled dependency syntax tree for *Coulourless green ideas sleep furiously*.

cle and exploited in the theory coined as Functional Generative Description (FDG) [Sgall et al., 1986].

Despite substantial differences among all these theories, the underlying idea is pretty much the same: syntactic structure is described by assigning each word – the dependant – to its syntactic head, i. e. the word that the dependant is governed by, together with a further specification of this relation, called usually dependency (or edge, arc) label. In Figure 2.3 I provide the same sentence as given in Figure 2.1 in the dependency notation. The set of syntactic relations describing the dependency (such as *being an object/subject/modifier*) is as important as the set of non-terminals in a phrase structure derivation tree, but dependency trees are often presented (and evaluated) without any labels.

Without any doubt the most important recent contribution to the dependency syntax is the Universal Dependencies movement. This project with very practical outcomes in the form of dozens of syntactically annotated text corpora (treebanks [Nivre et al., 2016]) with annotations following a single dependency scheme based on the original Universal Stanford Dependencies [de Marneffe et al., 2014].

2.4 Phrase structure vs. dependency syntax

Before moving forward to more advanced theories and grammatical frameworks, which are building on phrase structure and dependency syntax, and sometimes going far beyond the borders of syntax into semantics, I would like to do away with the question whether these two antagonist theories have a clear winner that outperforms its counterpart. I am not aware of any arguments suggesting that this would be the case. Both of these theories have their Achilles' heels bound to the principles they rely on and from a practical perspective the fitness of each should be always judged depending on the

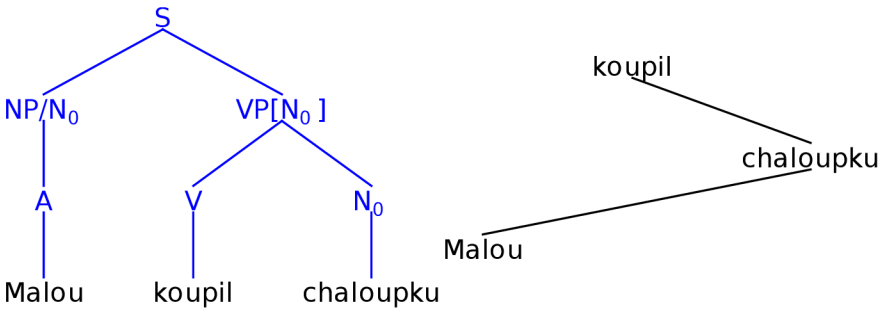


Figure 2.4: Comparison of discontinuity in phrase structure and dependency syntax trees for the sentence *Malou koupil chaloupku*.[2]

particular task that they are applied to. Some of the most notable practical differences are outlined as follows:

2.4.1 Discontinuity and non-projectivity

Sentences with discontinuous phrases like *Malou koupil chaloupku* represent a difficulty for any syntactic theory. While it is possible to tweak the phrase structure annotation to account for these (usually by means of co-indexing and introduction of slash non-terminals of the discontinued phrase, see [Gazdar et al., 1985] for detailed analysis of English), they are by all means easier to be represented within the dependency syntax where they yield a non-projective edge as can be seen from the comparison provided in Figure 2.4.

In Czech and many other free word order languages where non-projectivity is claimed to be not so rare as it is otherwise, the phaenomenon of non-projectivity has been paid a lot of attention (see e.g. [Hajičová et al., 2004, Havelka, 2007]), perhaps more than it deserves. In the example of the Prague Dependency Treebank (PDT) [Hajič, 1998], one can see that the number of non-projective edges in the treebank largely results from intentional design decisions about the annotation scheme and many of them would be avoidable, arguably with no loss in syntactic information.

Hajič et al. reported in Chapter 2 of [Hajič et al., 1998] only 1.8 % of all dependencies in PDT 1.0 to be non-projective, corresponding to almost 80 % of all sentences having no non-projective dependency and more than 93.8 % sentences having at most one non-projective dependency.

Czech is sometimes claimed to be one of those languages where due

2. in English *He bought a small cottage* given as *Malou/Small/AdjAcc koupil/bought/VPast chaloupku/cottage/NounAcc*.

to the free word order non-projectivity almost enforces the usage of dependency syntax (e.g. in [McDonald et al., 2005] or [Hall and Novák, 2005]), but in fact non-projectivity is still very rare even in Czech and can be often avoided. This claim can be supported by taking closer look at the analysis of non-projective dependencies in the PDT provided in [Hajičová et al., 2004] where according to the classification at least 70.5 % of the non-projective dependencies[3] could have a very straightforward projective variant (within the dependency notation) or continuous phrase structure representation.

To conclude: while non-projectivity is a valid issue in processing syntax of natural languages (for some of them more serious than for others), it is not a very strong argument against phrase structure syntax. The phaenomenon is rare, moreover non-projectivity can be expressed in dependency syntax with ease but cannot be automatically parsed easily. In fact non-projective parsing brings a lot of additional (design and computing) complexity, so that most dependency parsers actually carry out only projective parsing (for an overview see [Nivre and Nilsson, 2005]).

2.4.2 Coordinating dependencies

From the opposite direction there are valid objections that for some constructions (primarily coordinations) it is very hard to imagine what kind of a dependency relation there should be (e. g. between a conjunction and two members it is coordinating). Again, from the practical point of view this does not play a role at all, at least as long as one is able to reconstruct the coordinating structure. An overview of handling coordinating structures in dependency syntax can be found in [Popel et al., 2013].

2.4.3 Phrase Structures vs. Dependencies: Conclusions

From a theoretical perspective there is only little research as for the formal correspondence of dependency and phrase structure grammars. An interesting overview can be found in the (otherwise unpublished) manuscript [Abney, 1995] where the relationship between dependency and context-free grammars is discussed, concluding that depending on how the notion of headedness is treated, different kind of equivalence classes between these two formalisms arise (with regard to *headed context-free grammars* which

3. Namely the following categories in the referred paper: A1 – function words (21 %), A2 – prepositional groups with a focus sensitive particle (28 %), A3 – noun phrases with both preceding and succeeding genitive numerals (0.6 %), B4 – numerals with dislocated dependent (1.3 %), B6 – relatives and interrogatives (1.6 %) and B8 – particles (18 %).

Abney introduces), and neither subsume each other. The limited interest of computer scientists in this topic is likely to be explained by the fact that while phrase structure grammars find major exploitation in formal languages, this is not the case for the dependency system. In addition to that, the linguistic audience never showed much interest in taking this kind of research into account, as Abney himself politely summarizes:

"In closing, it is perhaps appropriate to discuss the import of the result. From a mathematical perspective, equivalences between types of representation provide useful tools. Some results are easier to discover or prove using one type of representation; some are easier to discover or prove using the other. The equivalence allows one to use either type of representation interchangeably. Judging from the literature, the linguist's attitude tends to be rather different. If one shows that two representations are equivalent in some respect, the immediate response is to seek other arguments for why one or the other representation is right. Either the correct structure is a phrase-structure tree, or it is a dependency tree, but it certainly cannot be both. Assigning both a phrase structure tree and a dependency tree to the sentence would be at best absurdly redundant. Accordingly, either DG is the correct grammatical formalism, or CFG, but not both." [Abney, 1995, p. 7]

Finally, an objective assessment of the relationship between from both formal and practical point of view can be found in closing of his paper:

"If the result reported here is of linguistic interest, it is because some constraints on syntactic structures are most easily established using projection dependency trees, and others are most easily established using characteristic trees. The result does not mean that phrase-structure trees are right and dependency trees are wrong. Indeed, if one is a true description of a sentence, the other is ipso facto true; though they may be differentially useful in different contexts." [Abney, 1995, p. 8]

This presents a view I would definitely subscribe to. In practice it is a fairly easy task to write an algorithm that will convert a phrase structure tree to its structurally equivalent dependency counterpart and vice versa. The real difficulties are in converting the annotation schemes that either of the formalisms used to exercise. In terms of computer processing, the significant notational simplicity of the dependency formalism over the phrase structure notation represents a great advantage, and allows for easy human readability and rather straightforward automated processing. Ultimately, the decision which of these two elementary syntactic approaches should be used always depends on the particular practical application. Thus, a parser should ideally be able to produce both.

2.5 Advanced Syntactic Formalisms

In the following I provide a very brief overview of advanced syntactic formalisms that adopt and extend some of the principles of phrase structure or dependency syntax. Rather than a syntactic formalism, some of them represent a full-fledged linguistic framework for the description of all linguistic levels. While this may suggest solid formal consistency, the complexity of these formalisms together with many additional assumptions they impose (both in terms of linguistic theory as well as practical usage) means a substantial obstacle in using them.[4]

2.5.1 Generalized and Head-Driven Phrase Structure Grammar

Head-driven phrase structure grammar (HPSG) [Pollard and Sag, 1994] is a formalism that adopts the basic phrase structure syntax and builds on the Generalized Phrase Structure Grammar (GPSG) [Gazdar et al., 1985]. The GPSG was a rather practically motivated initiative trying to provide a mechanism for describing English by means of a context-free grammar. GPSG introduced many useful concepts in grammar development, such as the usage of a metagrammar or the ID/LP notation, while formally pertaining to the context-free framework.

The idea of a metagrammar, i.e. a grammar using additional notation (operators and other syntactic markup) allowing for simpler, more readable and comprehensive grammar writing, was later practically implemented in many parsing systems including Synt which is described in Chapter 5.

The ID (immediate dominance)/LP (linear precedence) grammar notation decomposes the traditional phrase structure rule productions ($A \rightarrow BC$) into two independent descriptions: one focusing on syntactic immediate dominance (A immediately dominates B and C) and another one focusing on surface linear precedence in the sentence (B precedes C). An ID/LP-based grammar describes both of these principles separately, using different notation.

The ID/LP principle was later adopted by the (much more famous and widely-used) successor of GPSG, the Head-Driven Phrase Structure Grammar (HPSG). In addition to the ID/LP principles, HPSG is not a derivational, but a unification-based formalism – it exploits a sophisticated feature hi-

4. Throughout this section, illustrations of parse structures of HPSG, LFG, (L)TAG and CCG parsing were obtained from slides prepared by my former teachers Yi Zhang and Antske Fokkens for the Syntactic Theory course at the University of Saarland. Available from http://www.coli.uni-saarland.de/courses/syntactic-theory-09/.

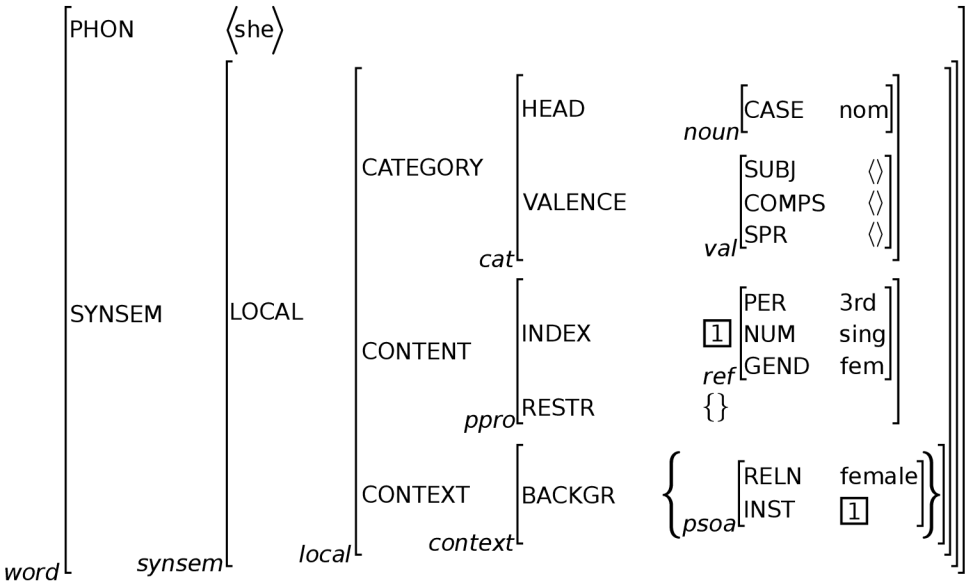


Figure 2.5: Sample lexical input for *She* in the form of an attribute-value matrix for HPSG parsing. Picture source: see note 4 on page 15.

erarchy attached to the phrase elements called *typed feature structure* and formalized as a bounded complete partial order (see Figure 2.5).[5]

HPSG is lexicalized (words and phrases are handled differently) and goes beyond syntax in that it also specifies semantic categories (such as agent, patient etc.). All these added features come, not surprisingly, at the cost of added complexity, both in terms of the underlying theory and parsing speed: it has been shown that the recognition problem for HPSG is NP-complete [Trautwein, 1995] and while it is likely that relaxing the problem (by restricting the grammars) may yield efficient (i.e. polynomial) solution, the efficiency in parsing of large volumes of data remains an issue.

2.5.2 Lexical Functional Grammar

Lexical functional grammar (LFG) [Bresnan, 2001] is another grammatical formalism building on phrase structure syntax. It does so in a very straightforward way: an LFG analysis consists of two steps, first a "standard" phrase structure analysis is performed, and the resulting derivation tree is called a *c-structure* as for the constituent structure. In the second step, the *c-structure* is

5. For illustration how complex the feature hierarchy may be readers are invited to have a look at the LinGO Grammar Matrix Type Hierarchy Chart prepared by Yi Zhang and available at http://www.coli.uni-saarland.de/~yzhang/downloads.php.

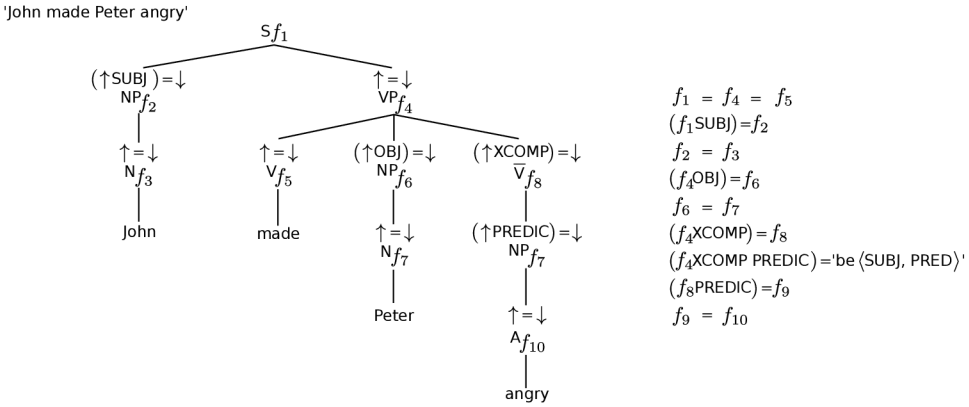


Figure 2.6: Sample analysis of the input sentence *John made Peter angry* in LFG: feature constraints are applied on top a phrase structure tree, hierarchically propagated and unified. Picture source: see note 4 on page 15.

exploited to build a so called *f-structure*, functional structure. This is done using a unification framework over a set of feature structures that hierarchically propagate over the *c-structure* tree. In this respect, LFG parsing is similar to HPSG parsing, and not surprisingly also NP-complete [Trautwein, 1995], though – again – efficient implementations were devised by constraining the underlying LFG grammar.[6] An example of an LFG analysis is provided in Figure 2.6.

2.5.3 Tree-Adjoining Grammar

Tree-adjoining grammar (TAG) [Joshi and Schabes, 1997] is a syntactic formalism based on phrase structure syntax where the building elements of the analysis are not words and non-terminals, but partially specified syntax trees that are subject to several admissible tree operations such as substitution or adjunction: the expansion of an inner-node non-terminal by the insertion of a recursive subtree (i.e. a tree which has a dedicated leaf node labeled with the same non-terminal as the root of the subtree).

Thanks to the adjunction operation (illustrated in Figure 2.7) TAG has a higher generative capacity than a context-free grammar, but lower than a context-sensitive grammar. The class of languages such a system may generate is usually called mildly context-sensitive languages, and it has been shown to have parsing time in $\mathcal{O}(n^6)$ time complexity [Satta, 1994]. This is a clear benefit over HPSG or LFG, but still significant slowdown from the $\mathcal{O}(n^3)$ parsing time of a context-free grammar. The formalism later evolved

6. See e.g. the XLE project, http://www2.parc.com/isl/groups/nltt/xle/

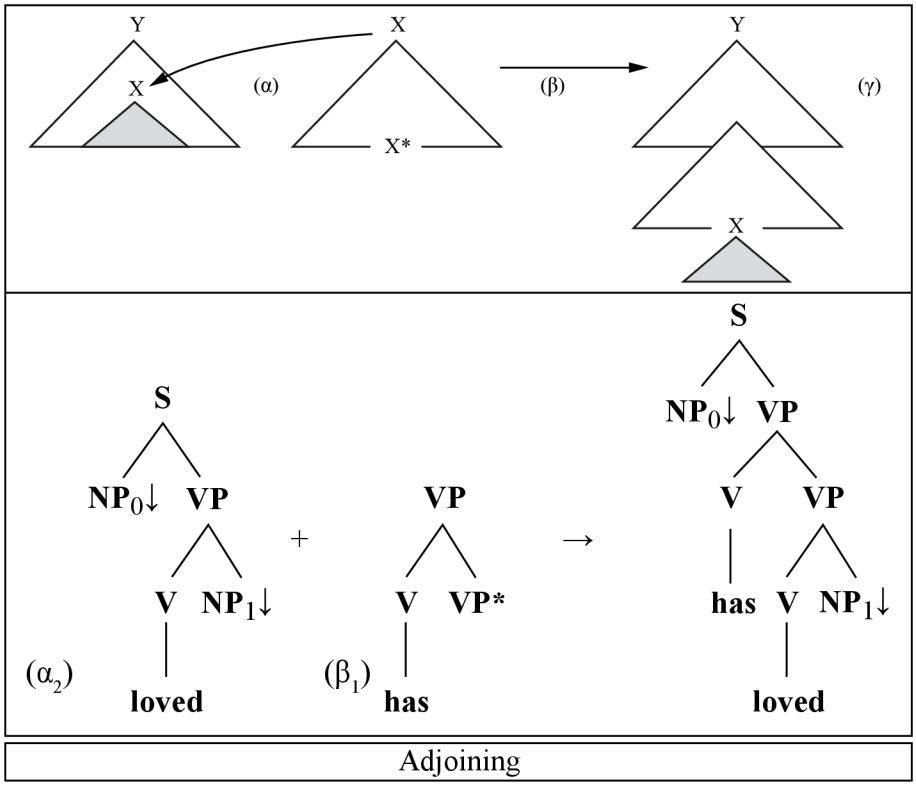


Figure 2.7: Combining an initial (α) and auxiliary (β) by adjoining the β tree into the α tree. Picture source: see note 4 on page 15.

into Lexicalized Tree-Adjoining Grammar (LTAG), having the elementary trees assigned a lexical anchor (i. e. word).

2.5.4 Combinatory Categorial Grammar

Combinatory categorial grammar (CCG) [Steedman, 2000] is a formalism that also uses phrase structure representation. It is not derivation-based, but operates instead using composition of functions (combinatorial operators such as application of a function to an argument, combination of two functions, associativity or type-raising). Formal background is adopted from typed lambda calculus. It has been proven that CCG is a weakly equivalent formalism (in terms of generating the same string language) to TAG [Vijay-Shanker and Weir, 1994] and from a psycholinguistically perspective, it is argued that CCG might represent a plausible model for human syntax parsing as it implicitly accounts for incremental processing of the input

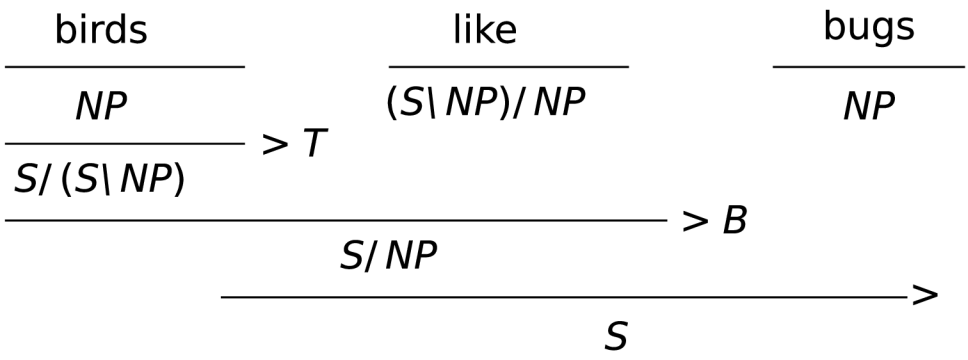


Figure 2.8: CCG analysis of the input sentence *Birds like bugs*. $>T$ is the type-raising operator, $>B$ is the composition operator, $>$ is the application operator. Picture source: see note 4 on page 15.

sentence. An example of a CCG analysis is given in Figure 2.8.

2.5.5 Link Grammar

The Link Grammar [Sleator and Temperley, 1993] attempts to merge the advantages of both phrase structure and dependency syntax. While not so widely adopted as the previously mentioned formalisms, and tightly bound with one particular implementation, the Link Parser, it represents and interesting project that is practically motivated and was exploited in various tools like the AbiWord processor. The result of the Link Grammar parser is called a linkage, and it is a set of binary relations on words, but not head-dependent relation, just *links* that are left-right directional, have a given length and usually have constituent interpretation. The parser also produces a constituent tree from the linkage.

2.5.6 Conclusions

This very brief overview presented above suits just to demonstrate the broad landscape of syntactic theories that were developed over the past decades. On one hand, there was a clear trend towards more complex theories, trying to extend the expressive power of the formalisms and involving semantics and syntax into one package, both being plausible from the point of view of numerous linguistic schools. On the other hand, recent efforts within the Universal Dependencies project show that addressing issues of practical applicability and multilinguality requires starting simple.

I find it being a misfortune that the development of various syntactic theories was not driven by the fact that existing ones would show to be *practically* insufficient for the purposes of NLP, as it contributed to unnecessary

pulverization of efforts in parsing and even to misleading the purpose of parsing to some extent, as is discussed further in this thesis.

From a linguistic point of view, some of these advanced formalisms (HPSG, LFG, TAG) assume having large lexical resources available (such as valency dictionaries, lexicons with semantic frames etc.). These resources are usually available just for a few languages, and even for those (and even for English) their coverage is often limited, and hence is limited also practical applicability of the tools that are based on them. Moreover, some of the linguistic concepts (such as verb valency, as discussed in Section 2.2) are rather well theoretically developed but when facing reality (i.e. evidence from large text corpora) such concepts sometimes fall apart because of frequent phaenomena like partial sentences or elliptic structures.

Finally, as I will discuss further, even basic syntactic formalisms like dependency or phrase structure syntax are still very far from having exhausted their potential as indicated by the performance of the state-of-the-art parsers, which is still often found being unsatisfactory.

Chapter 3

Parsing

3.1 Overview

Given a syntactic theory (or at least a certain formalism with elementary syntactic principles), what are the options to devise a tool that would automatically produce the analysis according to this theory for an input text?

While there are very many theories with many variants, the approaches for parsing generally follow the mainstream paradigm in natural language processing: the automation can be either manually defined (rule-based), or learnt from training data (statistical) – or take the advantage of both of these approaches. Early natural language processing systems were almost solely rule-based, and the field witnessed a revolution in the 90's towards using statistical methods.

This revolution was possible only thanks to the technological advances and sudden availability of enough natural language texts in the electronic form – text corpora – with internet being the biggest text corpus in certain sense. This revolution is still, evolutionally, ongoing[1], and while it brought many fruits and caused major breakthrough in some NLP tasks (such as part-of-speech tagging, machine translation or speech recognition), the absolute commitment of the mainstream community towards (only) statistical processing is occasionally being subject to criticism.

Perhaps the most comprehensive criticism was provided by Church, a leading figure of the statistical revolution of 90's, in his 2007 paper *A Pendulum Swung Too Far*. He reviews the 50-years-old history of computational linguistics, and records the ideological shifts from Rationalism to Empiricism about every 20 years. Because of that, he emphasizes his concerns especially in the context of university education as the need to teach about the opposition that is soon to come back ruling:

"*Students need to learn how to use popular approximations effectively. [...] Many of these limitations are obvious (by construction), but even so, the debate,*

1. And now being exploited by business marketing terms coining it as Big Data.

both pro and con, has been heated at times. And sometimes, one side of the debate is written out of the textbooks and forgotten, only to be revived/reinvented by the next generation. We should encourage the next generation to learn the arguments on both sides of these debates, even if they choose to take one side or the other. "

But his criticism goes much further: he acknowledges his own generation for picking up "low-hanging fruit" available through statistical (empirical) methods with no deep language understanding, and warns that this low-hanging fruit will be gone at some point.

"When we revived empiricism in the 1990s, we chose to reject the position of our teachers for pragmatic reasons. Data had become available like never before. What could we do with it? We argued that it is better to do something simple than nothing at all. Let's go pick some low hanging fruit. [...] That argument made a lot of sense in the 1990s, especially given unrealistic expectations that had been raised during the previous boom. But today's students might be faced with a very different set of challenges in the not-too-distant future. What should they do when most of the low hanging fruit has been pretty much picked over? "

Church hypothesizes whether the next ideological revolution is to happen around 2010. As of 2017, it has not come – rather vice versa, the revival of neural networks (an "old technology" resurrected thanks to a real progress: the introduction of the word embeddings by [Mikolov et al., 2013]) supported by a massive PR of major technological companies like Google, Facebook and others, has affirmed the statistical paradigm even in business management.

3.2 Rule-Based Parsing

Rule-based parsing of natural languages is tightly connected with the times when foundations of the theory of formal languages have been laid down. The task has been seen as follows: give me a grammar and a sentence, and I will tell you whether this is a well-formed sentence according to the grammar, and what would a derivation tree look like. The massive ambiguity of natural languages required multiple adjustments to the core algorithms that were used to parse simple and deterministic communication protocols and programming languages. This was a tempting task for computer science which made a great contribution to the parsing algorithms, using standard techniques such as dynamic programming and memoization, and resulting into various practically usable algorithms for efficient parsing of context-free grammars (see e. g. [Kay, 1986]).

There are however considerable theoretical as well as practical related questions: given the variety and complexity of natural languages, is it feasible to manually describe them by rules? Indeed the development and maintenance of hand-written grammars turned out to be a hard job that is still considered to be one of the main counterarguments against rule-based parsing.

Moreover I believe that the development of corpus linguistics has clearly shown that the equation between a language and grammar is a *rationalist dream*.[2] Does that, however, rule out rule-based parsing as such? As I explain in further text, I do not think so.

3.3 Statistical Parsing

Issues related to grammar development and the rise of statistical NLP in the past two decades led to an expectable outcome: new parsers were developed by learning a grammar from existing corpora with syntactic annotation (see e. g. [Brill, 1993, Belz, 2002]), or by skipping a formalized grammar entirely by producing statistical models that can be directly used for parsing (see e. g. [Charniak, 2000]). Statistical methods proved to be the most promising solution in many areas of NLP (tagging, machine translation etc.), they are *the* solution in speech processing, so why not use them for parsing?

Alongside of the development of statistical methods in NLP, syntactically annotated corpora (treebanks) have been prepared for a raising number of languages, serving as a source for training, development and evaluation data. Concurrently, the efforts in parsing over the past ten years have been mostly oriented to statistical parsing (as can be seen in the overview of parsing systems later).

At this point I will draw the attention to the purposes of parsing, and then get back to the question whether statistical modelling is – at least at this moment – the right solution also for parsing.

3.4 The Purpose of Parsing

Before moving to a rather critical review of state-of-the-art parsing, and related discussion about the advantages and disadvantages of rule-based and statistical parsing, I think it is first necessary to define a standing point in terms of a declaration on what I view as the purpose of parsing. Syntactic analysis is usually defined as "recovering the structure of a sentence", but

2. As my dear friend Adam Kilgarriff used to call it.

this tells about what the parser does, not what should its results be used for. I am not going to give any less vague definition, but focus on the purposes of parsing.

For many computational linguists parsing corresponds to producing some sort of a structure that fits and confirms a particular theory of syntax (or language in general). Though not saying that this might not be an appealing linguistic task, that is not the view I share at all. I see the purpose of parsing and parsers in terms of standard tools for natural language processing that do not represent a final goal as such, but should contribute to improve other applications and serve for many tasks.

This issue is closely related to how parsing results should be represented. I am confident about the fact that the most used representation, namely a tree (be it a phrasal or dependency one), is not the one that would be directly usable by many applications.

In general I have rather a skeptical attitude to how much parsing and parsers are useful nowadays. And I also think that it is this fact, that parsers are primarily not developed to serve in *multiple but particular* applications, and a general uncertainty about the representation of results, that make me conclude that parsers are hard to employ in existing applications and do not serve well for their particular needs. Of course, the obvious counterargument is that this approach might very well deteriorate into a situation when each application requires a principally different parser (and does parsing from scratch) and hence necessarily partially reinvents the wheel. I think this is a valid statement that may even come true, but before committing to such a negative result, I believe that the community has the duty to think of more reliable and useful approaches. I take this as a present challenge to elaborate on what would be a useful common representation of syntax that would be easily accessible by multiple applications.

Current state of parsing applications can be also seen reflected in proceedings and journal articles: the word *parser* is mentioned in 7,232 of 21,066 papers available within the ACL Anthology,[3] a collection of papers from the NLP domain over the past 20 years. But looking for phrases matching the regular expression *(used | using | employ | employing | exploit | exploiting) (a | the)? .* parser)* on the scope of a single sentence, one gets only 133 results. This is of course a very rough and not rigorous estimate, nevertheless the message is clear: most papers mentioning parsers talk about parsing but not about using parsers.

And even if they talk about using parsers, it is not unlikely to find papers

3. http://aclweb.org/anthology-new/

cautiously mentioning the truth:

> *"Our results do not yet indicate that parsing is beneficial to . . . "*
> [Musillo and Merlo, 2006, p. 104]

> *"There is no doubt that collocation extraction should be based on syntactic preprocessing of the source corpora (simply because collocations often have syntactic structure), but the evaluation presented in the book is not very convincing."*
> [Pecina, 2011, p. 633]

In [McCarthy and Navigli, 2007], word sense disambiguation has been described as a "task in need of an application", and I am afraid that this applies to large extent to parsing as well. Until parsing is going to become primarily application-driven, it will be subject to Anja Belz's proposition [Belz, 2009] used as a title of one of the Last Words' articles in the Computational Linguistics journal: *That's nice . . . what can you do with it?* A citation from the same source is also going to serve as my conclusion on purposes of parsing:

> *"If we don't include application purpose in task definitions then not only do we not know which applications (if indeed any) systems are good for, we also don't know whether the task definition (including output representations) is appropriate for the application purpose we have in mind."*
> [Belz, 2009, p. 113]

In the course of making parsing more applicable, I argue that both underspecification of phaenomena with low agreement and/or accuracy as well as revisiting various approaches to partial parsing (chunking) represent (maybe temporary) useful directions. Standalone research has been conducted in the direction of partial parsing, i. e. identifying either flat structures (phrases) or phrase subtrees in the given sentence (see e. g. [Abney, 1996]). Various instances of this task such as NP bracketing (see e. g. [Vadas and Curran, 2007]) and NP chunking (see e. g. [Sha and Pereira, 2003]) have been studied. The missing bits might be not to see partial parsing as a task that is that much different from full parsing, and integrate the capability to produce partial results (in various forms) into parsing systems including their evaluation data sets.

3.5 Parsing Evaluation

Proper evaluation is a crucial part of development of any NLP tool – and empirical research in general. Investigating how parsers are evaluated, it turns out that except for notable but very rare exceptions, most parsers are evaluated in a similar manner as many other NLP tools, namely compared to some gold standard parsing output handcrafted manually by human annotators. This is called *intrinsic evaluation* as opposed to *extrinsic evaluation* where the results of the tool are exploited in particular tasks (which then needs to be evaluated appropriately).

3.5.1 Intrinsic evaluation

The intrinsic evaluation certainly allows for faster development of tools without lot of understanding of the underlying language phaenomena. In terms of computer science, this is an ideal world: you are given the input, your are given the desired output, you "just" need to program the "mapping". It is very likely that this also attracted many engineers, statisticians and computer scientists to the field of natural language processing because it allowed them to work without any linguistic education and without collaboration with linguists. The unwanted consequence: people wondering at conferences whether the field has become just part of applied statistics [4] and *not* looking at the data one works with becoming frequent (mal)practice.

Adam Kilgarriff has seen the NLP community overinvesting efforts into algorithms and underinvesting into (building, annotating and formalizing) data, as he reckoned two months before his death in a short post to the corpora-list:

> *"I often think, if NLP people put as much effort into the training data as the algorithms, our systems would perform much better. There's so much more I could say on the subject (on the problem of language-specific work in Computer Science, on what's open and what's proprietary, on generalising across text types, on statistical vs rule-based systems and in-built biases in favour of statistical ones, when we compare performance figures) – but the post would quickly get much too long!"* [Kilgarriff, 2015]

While some pros and cons of both intrinsic and extrinsic evaluation are obvious (such as general suitability vs. suitability for a particular task), I am

4. See e.g. the great talk of Shuly Wintner on this topic available at: http://www.cicling.org/2010/videos/#Wintner

strongly convinced of the fact that the intrinsic evaluation, though widely accepted, has severe theoretical as well as practical drawbacks. I discuss these drawbacks below, and argue that their severity depends on the type of data, and is particularly problematic in the case of syntactic annotation.

3.5.2 Agreement among linguists

A general theoretical remark follows up on the discussion of the purposes of parsing above: aiming at practical applications, it is hard to justify that a particular syntactic theory and representation is *the* syntax and that by approximating to it one is going to build a tool that will be useful for various applications, considering how little agreement there is in the linguistic community.

This should be read as a general critic related to the fact that, facing linguists (both academic and applied) professionally over ten years now, I allege that agreement is just neither valued in linguistics, nor is the absence of agreement considered a deficiency. This includes agreement on elementary terms, concepts, principles, systems – and theories. I do not know whether this is an issue particular for linguistics, or just the same as in other disciplines of humanities.

Humanities as such were criticized by many because of that, and a very good argumentation can be found in [Wilson, 1999]. In linguistics, criticism of this kind is rare, but present. A harsh condemnation can be found in the (already quoted) interview with Gazdar:

> *"Q: So, to somebody coming into the field, offered a job in a linguistics department, what would you say?*
> *A: Learn how to use a computer and change department.*
>
> *[. . .]*
> *Q: I've heard you say that the typical linguist knows a bunch of cute facts, but that knowing cute facts isn't a sufficient goal for linguistics as a field. Why not?*
> *A: Linguists get brownie points for cute facts. [. . .] Such people are tremendously useful if, for example, you are developing a computational grammar of English. [. . .] So the ability to deal in cute facts is not something to be dismissed lightly. On the other hand, for that skill to be what makes the field tick seems to me to be unsatisfactory from an external viewpoint. Imagine what mathematics would be like as a discipline if the key to career advancement was an ability to do complex mental arithmetic quickly. If you think about funding linguistics departments in universities so that they do the work necessary to describe natural languages*

and to theorize about the nature of natural languages in a way that constitutes a contribution to scientific knowledge, then linguistics departments patently don't do that. That's not where the brownie points are. A lot of it is hermetic, it's not actually of any use to anyone else. It's a closed internal system. If I was charged with university funding, I would pull the plug." [Briscoe and Gazdar, 2001]

From a perspective of a computer scientist, this is a big problem though: when things are not well-defined, they are hard to formalize – which really is the first step of any automation. It might be also the reason why many computer scientists working in NLP, including me, in the end start revisiting the linguistic fundamentals they should be building on.

3.5.3 Agreement among annotators

Another related issue is the inter-annotator agreement achieved on human annotated data taken as the gold standard. Given a theory of (e.g.) syntax, are human annotators able to consistently apply it to real language (i.e. corpus) texts? Even though this problem has been investigated a lot, and measures have been developed to account for chance agreement, such as Cohen's and Fleiss' Kappa (see [Cohen et al., 1960, Carletta, 1996, Fleiss, 1981], a very good overview was also given in [Artstein and Poesio, 2008]), the overall importance of the inter-annotator agreement still seems to be underestimated.[5] Let me put it in very simple terms: when humans are annotating natural language data, the two absolutely most important variables of interest are:

1. size of the data,
2. achieved inter-annotator agreement.

Language phaenomena follow almost always the Zipfian distribution ([Zipf, 1949], example in Figure 3.1) – be it word forms, lemmas, part-of-speech tags or syntactic structures. When working with language, one is most of the time looking at the long tail of close-to-hapaxes. It also follows that the variance when comparing small datasets is huge: different text types, topics or genres (registers) yield substantially different tails of the distribution. Size really matters (quality untouched).

Inter-annotator agreement for syntactic annotation is, to the best author's knowledge, not known for the most used English syntactically annotated corpus (treebank), namely the Penn Treebank [Marcus et al., 1993],

5. Often I suspect this to be deliberate as it of course might invalidate achieved results.

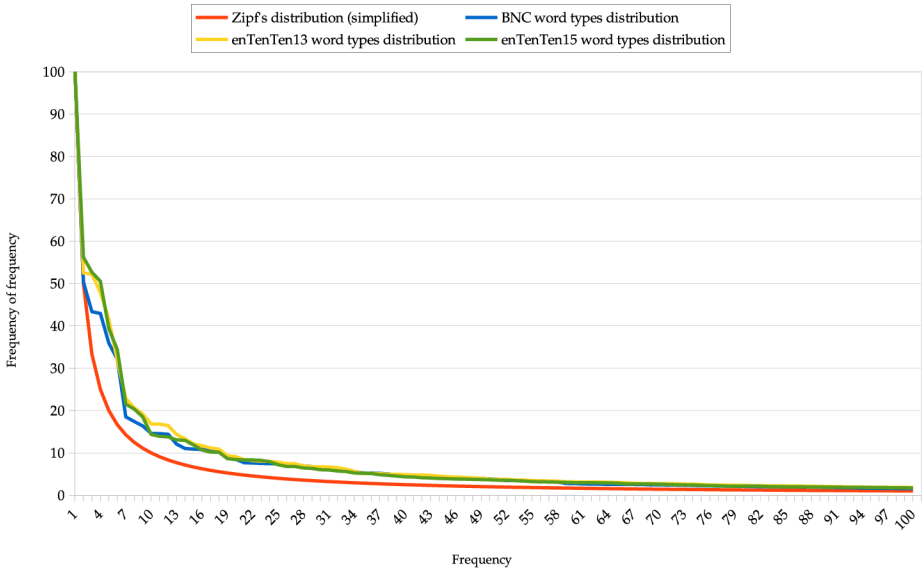


Figure 3.1: The figure shows an approximation of a Zipfian distribution given as that frequency of n^{th} most frequent word type equals frequency of the first divided by n and the distribution of lowercased word types derived from the British National Corpus and English web corpora of the TenTen series from 2013 and 2015. [Jakubíček et al., 2013]

the same applies for the already mentioned largest Czech treebank PDT. Some figures are available for the German NEGRA and TIGER treebanks [Brants, 2000, Brants and Hansen, 2002], unfortunately only in terms of an f-score that is claimed to be an appropriate measure of inter-annotator agreement, computed on syntactic nodes that have been identified by two annotators as same.

I am aware of two large studies trying to discover what inter-annotator agreement is actually achievable on syntactic annotation. They differ slightly in their approach but came to very similar conclusions. One is the work conducted by Sampson and his colleagues [Sampson and Babarczy, 2008] where they were measuring *themselves* on the task of annotating a corpus according to the (phrase structure) syntactic formalism they had been developing for years. They have achieved an average inter-annotator agreement of 95 % and, more importantly, they have shown that more precise annotation instructions do *not* help to achieve higher inter-annotator agreement. Conclusions: the human capacity is simply limited, and annotation manuals need to be simple otherwise people are not able to follow them consistently.

The second, more recent study, is the work done in [Berzak et al., 2016] measuring inter-annotator agreement on the English Penn Treebank (the

by far most widely used English corpus for parsing evaluation) using the Universal Dependencies guidelines. They worked with independent annotators, and from the description of the experiment it is obvious that they took every possible effort to train them. The annotators first took a two-month training to get familiar with the annotation scheme, then they annotated significant amount of training sentences, compared the annotations, discussed differences etc., before getting to any real annotation setup. In the actual annotation experiment, they used a setup of two annotators backed by two reviewing annotators, who might intervene by correcting mistakes. Still, the agreement on assigning dependency relations was just 92.3 % before the reviewing annotators and 94.04 % after the reviews.

On one hand, 94–95 % percent might seem good enough, on the other hand this still means that there is a disagreement in almost every single sentence. Moreover, it presents a theoretical challenge how to interpret results of parsing tools that are *above* the human agreement values. It can be only hardly argued that such evaluation is scientific.

Neither the former nor the latter study evaluated chance agreement and it is obvious that it might not be easy to calculate it properly in case of syntactic annotation. In [Skjærholt, 2014] it is suggested to use a tree edit distance measure for comparing two annotated trees:

> *"In this approach we compare tree structures directly, which is extremely parsimonious in terms of assumptions, and furthermore sidesteps the problem of probabilistically modelling annotators' behaviour entirely."* [Skjærholt, 2014, p. 936]

This is, however, entirely flawed and missing the point of correcting for chance agreement. Correcting for chance agreement is just all about modelling the annotators' behaviour to discount chance (i.e. random) agreement resulting from individual decisions made by the annotators – "sidestepping" it negates the main principle of correcting for chance agreement.

It is crucial to realize that the same annotation scheme would need a different chance agreement correction depending on the annotation (decision-making) process. The choices that are to be modelled are not necessarily those given by the annotation categories, but those that the annotators are asked to make. If annotators are asked to make a series of binary decisions that eventually lead to one of a few annotation categories (in case of syntax e.g. a dependency head or label), then corrections for chance agreement must be based on modelling this binary decision-making process, independently of how many possible annotation categories there are in the end.

Finally there is also a rather philosophical question regarding this topic

that I leave unanswered: if inter-annotator agreement should be a scientific measure in Popper's sense [Popper, 1959], under what conditions is it falsifiable?

3.5.4 Comparing trees: similarity metrics

Last counterargument to evaluation against treebanks is rather a practical one – what is a proper similarity metric for a syntactic tree (or any non-flat structure)? There are several established similarity metrics for this purpose. For phrase structures trees, it is mainly the PARSEVAL metric [Abney et al., 1991] and the Leaf-Ancestor Assessment (LAA) metric [Sampson, 2000] which has been later argued to be more appropriate than PARSEVAL in [Sampson and Babarczy, 2003]. For dependency structures, plain dependency precision (in terms of correctly identified dependencies) can be used, either with or without taking dependency labels into consideration, usually called unlabelled attachment score (UAS) and labelled attachment score (LAS).

The root of this problem lies however in the fact that any general similarity metric is necessarily going to suffer from the issue that all parts of the syntactic annotation are not equally important: whether an interjection was misattached in the tree is much less important than if it was the subject of the sentence. Therefore any such similarity metric cannot reliably answer the question how good or bad the result actually is, only how much it is structurally different. In order to avoid this, one can of course try to devise some sort of very complex weighting of different parts of the structure, but this both very hard to define and very hard to justify in front of the academic audience. As result, it would very likely lead to incomparable evaluations, and this is the most likely reason why this strategy was never adopted in the mainstream evaluation approaches.

3.5.5 Extrinsic evaluation

Promising attempts to make a comparison of parsers performance on a particular task were performed for English. In [Miyao et al., 2009] parsers were about to extract protein-protein interactions from biomedical texts and it is not surprising that their performance did not correspond at all to their f-score rank on the Wall Street Journal portion of the Penn Treebank. A similar attempt has been done when parsers contribution to machine translation has been evaluated in [Katz-Brown et al., 2011], coming to the same conclusion.

Both of these represent a serious counterargument against intrinsic eval-

uation: evaluating parsers performance on treebanks is not appropriate and does not correlate with primary end applications parsers were designed for.

It is clear that evaluating parsers on particular applications is much more difficult than just running a test suite that will compute the similarity scores against some gold data set. Moreover, one would need to evaluate against many (as much as possible) applications to avoid overfitting of the parser to one very narrowly defined particular application. Nevertheless I consider this to be the necessary step to move the parsers actually closer to their final tasks, and address this issue in the aims of my thesis as well.

3.6 Criticism of Statistical Parsing

In the following I would like to get back to the question of the appropriateness of statistical parsing. To make it clear right at the beginning, this is to sum up my arguments against statistical parsing as one of the state-of-the-art methods. At the very first, let me add that this is not to discourage from statistical methods as such. I do believe that the progress of statistical NLP that happened over the past decades was (and is going to continue to be) an extremely important contribution. Statistical methods have proven to be so useful for many NLP tasks, so why not for parsing?

To explain, let us compare parsing with one of the tasks where statistics is the core of any serious solution to the problem, namely morphological tagging. Tagging has all the properties that make it a good candidate for statistical modelling: it is a clearly defined and formalized task with well-measurable solutions and rather high inter-annotator agreement, its output is directly usable *and used* by many applications. In other words: we know what we want[6].

Parsing, on the other hand, lacks almost all of these properties: as discussed, its definition and representation of results is tightly bound to a particular theory, parsing accuracy is hard to be evaluated precisely and often relies on language resources with unknown inter-annotator agreement, parsing output is frequently not directly usable for the end application as such.

Now what does a statistical system do? It basically tries to mimic some data. But this principle crucially relies on the assumption that what we "ape" is known to be correct, to be exactly what we want. I am convinced that this is, unfortunately, not the case of parsing nowadays, where often one does *not* know what the result should look like in order to be easily usable, and

6. Though even for tagging the inter-annotator agreement turns out to be an issue, as discussed by [Manning, 2011].

the reliability of the training data is at least questionable.

What we need are parsers – flexible tools – that can be easily adjusted for various so much different tasks. This is a strong advantage of rule-based parsing – one can easily modify and accommodate the parser to new tasks. A "statistical" answer would be: well, let's annotate some new data. But this is a problem not only from a practical point of view, but, as was shown in the discussion of parsing evaluation, mainly from the theoretical one (especially in case of development of new annotation schemata).

I consider this inappropriateness of statistical parsing to be a real misfortune, because statistical parsing has on the other hand two very important advantages: an implicit probabilistic nature and lexicalization. These two make it winning the f-score competitions on the Penn Treebank, and both of them are not straightforward to implement in rule-based parsing.

Nevertheless I believe that – at least for now – our efforts should be concentrated on rule-based methods, trying to minimize their disadvantages (such as grammar development and maintenance) and incorporating advantages of statistical parsing, with the primary aim of making actual use of parsing. Only at that time we can revisit the approach of statistical parsing.

This critic should by no means be interpreted as the critic of empiricism. To be come applicable, parsing as such inevitably had to move from its commitment to *Language of possibility*, pursued by Chomsky, to *Language of probability*,[7] based on language evidence found in corpora. The problem is that this shift should not end with replicating whatever arbitrary annotation (and text types – genres and topics) is used in treebanks, but should be motivated by what the results of parsing are envisaged to be used for.

3.7 On Writing Rules

I often suspect that the condemnation of grammar writing (in favour of grammar learning/training) happened largely as result of a workflow which assumed computer scientists devising a formal grammar system, and leaving the tedious task of "just writing the rules" to the linguists.

In my experience, such setup almost never works. Unfortunately linguists mostly do not have the necessary skills for abstract formalization to carry out this task. They have great linguistic intuition, and knowledge about the languages – but these are simply different competences from those needed for writing a formal grammar. I believe it is the job of us, computer scientists, to do the labour with writing formal grammars, with the necessary

7. See [Kennedy, 2014].

assistance of linguists to provide us with what we need to know about the language.

At least all my attempts in grammar writing (both in case of Synt and word sketch grammars) clearly show that the right way to do this is not trying to explain the formalism, its theoretical background and necessary technical details to linguists and having them writing the rules, but vice versa: ask linguists for what they know best (telling about a language) but not having them doing the actual formalization.

This should not be misread as underestimating the role of linguists in this process. It is just that after years of both teaching and collaborating with linguists on various grammar writing tasks, I realized that it is both unreasonable and unrealistic to expect them to do right what computer scientists learn for years: representing infinite structures (like language) in a finite way (like a grammar), seeing the impacts of choosing various options for formalization and finding the right level of abstraction.

The role of linguists in natural language processing has been discussed by many and I touched that topic lightly in [Jakubíček et al., 2017]. Natural language processing is an interdisciplinary field that lies between computer science and linguistics. As any such field, it needs to deal with the usual issues of interdisciplinarity: the cooperation of researchers coming from different backgrounds is not straightforward at all, especially when the backgrounds cross the borders between humanities and natural sciences.

This goes hand in hand with the question how much NLP systems should exploit "deep linguistics" or rather "brute engineering". Perhaps the most famous example can be ascribed to Frederick Jelinek whose hyperbole went, "*Whenever I fire a linguist our system performance improves*" [Jelinek, 1998]. After this quote became (in)famous he later explained, reckoning the fact that many language engineers are facing: it is not at all easy to use linguistically motivated and knowledge-rich techniques in ways that actually contribute to improving performance, even though one would wish so: "*We were never reluctant to include linguistic knowledge or intuition into our systems: if we didn't succeed, it was because we didn't find an efficient way to include it*" [Jelinek, 2005, p. 25].

He concludes his paper with words that I entirely subscribe to:

"*Linguists study language phenomena as much as possible like physicists study physical phenomena. They will give us advice, but will not directly engage themselves in building systems. Just as engineers learned to take advantage of the insights of physics, it is our task to figure out how to use the insights of linguistics. Our main problem is sparseness of data, and more precisely of annotated or categorized*

data. What we should ask is that linguists help us structure systems capable of extracting knowledge under minimal human supervision." [Jelinek, 2005, p. 32]

The *minimal* in *minimal human supervision* is for me the crucial part of that paragraph. Not only because human annotations are tedious and costly to carry out, but mainly because for many tasks it is so hard to obtain reliable annotations that humans would agree on.

3.8 Parsing Systems

In this section I provide a short list of existing parsing systems and their relation to the presented syntactic formalisms. The list has no ambition to be exhaustive, since there exist numerous of them, I only mention those that have multi-language scope (or I consider them to be worth mentioned for other reasons) and those available for Czech in particular.

In recent years, there were many new neural network-based dependency parsers developed, some of which I mentioned below. An evaluation of these dependency parsers was done by [Choi et al., 2015] (in an intrinsic way on the OntoNotes corpus [Weischedel et al., 2011]). A more detailed analysis of current challenges in dependency parsing within the scope of the Universal Dependencies project is provided in [Kirilina and Versleya, 2015].

3.8.1 English and multi-language parsers

- MST parser [McDonald et al., 2006] is a graph-based statistical dependency parser. The name MST is an abbreviation of Maximum Spanning Tree since the parser first builds a dependency graph from which it later generates dependency trees. It has been trained for Czech too on the Prague Dependency Treebank as described in [McDonald et al., 2005].

- MaltParser [Nivre, 2009] is a transition-based statistical dependency parser. It comes with pre-trained models for English, French and Swedish (but has been tested with a large variety of languages).

- MATE parser [Bohnet and Nivre, 2012] is also a transition-based beam search dependency parser which allows joint part-of-speech tagging and parsing.

- RASP (Robust Accurate Statistical Parser) [Briscoe et al., 2006] is wide-coverage rule-based parser of English including a tokeniser,

tagger and lemmatiser with phrase structure trees and grammatical relations as output format.

- Stanford parser [Klein and Manning, 2003] is a statistical parser with both dependency and phrase structure output. Besides English it has been adapted for Chinese, German, Arabic, Italian, Bulgarian and Portuguese. It has been ported to a neural network backbone in [Chen and Manning, 2014].

- SyntaxNet is a neural network-based tagger and parser released by Google and built on top of the TensorFlow computing framework [Andor et al., 2016].

- Link Grammar Parser [Grinberg et al., 1995] is a rule-based parser that builds on the Link Grammar and produces the related linked structure. It is provided with grammars and lexicons for English, German, French and Lithuanian. It has been adopted by an open-source community that develops the AbiWord editor[8] and exploited as a grammar checker in the editor.

- Enju [Sagae et al., 2007] is a rule-based HPSG parser primarily developed for English, providing phrase structure and predicate argument structure as output.

- PET [Callmeier, 2000] is another HPSG parser providing a wide range of grammars including English, German, Chinese, Japanese, Spanish, Portuguese, Korean, Modern Greek, Norwegian and others, most of them originating in an grammar development environment called LKB – Lexical Knowledge Builder.

- XDG [Debusmann, 2006] is a dependency metagrammar formalism accompanied with a "solver" and sample grammars for a number of languages, including a prototype for Czech.

- C&C Parser [Clark and Curran, 2007] is a statistical parser for English exploiting a combinatory categorial grammar automatically extracted from the Penn Treebank.

- Collins' Parser [Collins, 1997] is a statistical parser developed by Michael Collins. Among other languages it has been also trained for Czech on the PDT.

8. See http://www.abisource.com/projects/link-grammar/.

- Charniak's Parser [Charniak, 2000] is a statistical parser developed by Eugene Charniak and also trained for Czech on PDT.

3.8.2 Czech parsers

3.8.2.1 Dependency parsers

A number of statistical parsing systems has been trained on the Prague Dependency Treebank, either in direct collaboration with people at the Prague Institute of Formal and Applied Linguistics at the Charles University, or within the scope of the CoNLL shared tasks in 2006, 2007 and 2009[9]. So far the CoNLL 2009 best parsing system was the one described in [Gesmundo et al., 2009], achieving labeled dependency score of 80.38 %.

A number of systems was already trained on the PDT converted to the Universal Dependencies annotation style, e.g. Google's SyntaxNet reports to have achieved labeled accuracy of 85.93 %.[10]

3.8.2.2 Phrase structure parsers

SET parser

The SET parsing system [Kovář et al., 2009] is a notable exception that goes in the opposite direction than the mainstream of syntactic processing – instead of introducing yet more complex syntactic theories and parsing systems, it tries to perform parsing using simple ranking of finite state patterns. It is still being developed in the NLP Centre at the Masaryk University and recently has been applied for noun phrase chunking [Grác et al., 2010] or punctuation correction [Kovář, 2014].

Synt parser

The Synt parsing system [Kadlec, 2007] has been developed over the past decade, and it originates also in the NLP Centre at the Masaryk University. It uses a probabilistic context-free backbone enhanced with contextual actions that are evaluated on top of the basic PCFG analysis. The parser has various output options, besides a phrasal tree it can also give an ambiguous dependency graph or flat structures. Within the work on this thesis it has been extensively developed and is described in more detail in Chapter 5.

9. See http://ufal.mff.cuni.cz/czech-parsing/ for an overview.
10. See https://github.com/tensorflow/models/blob/master/syntaxnet/g3doc/universal.md.

3.9 Parsing of morphologically-rich languages

Morphologically-rich languages (MRLs) present a number of challenges for parsing which has been first applied – like many other NLP technologies – for English. In fact, the discrepancy between the treatment of English and any other language in NLP is so big that when defining of what is a MRL one might well stick with the approach taken by the Oxford Handbook of Inflection which simply says: "We will also use the broad term morphologically-rich language (MRL) to denote a language richer than English" [Baerman, 2015]. This definition, while being so anglocentric, fits the purpose.

One thing that MRLs bring into parsing is, by definition, the rich morphology itself. The differences are obvious just by looking at the morphological tagsets of English and Czech. While the English one (as used in the Penn Treebank) contains some 60 atomic tags, the Czech one (as given by the Czech analyser Majka [Šmerk, 2009, Jakubíček et al., 2011]) has 2,873 tags,[11] composed of different grammatical categories. Correct estimation of many of these categories (in Czech mainly grammatical case, number and gender) is crucial for the success of the subsequent syntactic analysis. Indeed parsing of morphologically-rich languages starts with the morphological analysis and without any exaggeration one can claim that a successful morphological analysis is half of the parsing.

But for many MRLs, the heaviest impact is brought not by the morphological complexity itself, but by the free word-order that the morphology allows. E.g. in Czech, while there are some constituents that have a fixed sentential position (such as clitics, whose syntactic interplay was described in detail by [Hana, 2007]), the order of the constituents can be theoretically arbitrarily altered from the neutral standard (subject-verb-object) and still maintains grammatical correctness. In practice, it is used to express topic-focus articulation. Thus, the following are English-Czech equivalents:

(neutral form)
Peter gave a book to Paul.
$Petr_{Peter/Nominative}$ dal_{gave} $knihu_{book/Accusative}$ $Pavlovi._{Paul/Dative}$

(emphasis on Paul)
It was Paul who Peter gave a book to.
$Petr_{Peter/Nominative}$ dal_{gave} $knihu_{book/Accusative}$ $Pavlovi._{Paul/Dative}$

11. Without stylistic (archaic, conversational, dialectical etc.) variants. Counting them in, it yields 9,908 tags.

(emphasis on Peter)
It was Peter who gave a book to Paul.
$Knihu_{book/Accusative}$ dal_{gave} $Pavlovi_{Paul/Dative}$ $Petr._{Peter/Nominative}$
or
$Pavlovi_{Paul/Dative}$ dal_{gave} $Knihu_{book/Accusative}$ $Petr._{Peter/Nominative}$

(emphasis on book)
It was a book which was given by Peter to Paul.
$Petr_{Peter/Nominative}$ dal_{gave} $Pavlovi_{Paul/Dative}$ $knihu._{book/Accusative}$
or
$Pavlovi_{Paul/Dative}$ dal_{gave} $Petr_{Peter/Nominative}$ $knihu._{book/Accusative}$

As one can see, the emphasis is expressed just by ordering the constituents in the sentence and it is always present (first two Czech sentences are same) as Czech puts it on last constituent in the sentence. Of course, this influences the parsing of the sentence, and makes it particularly difficult to manually hand-craft rules that would capture all combinations. However, practical methods to mitigate these issues have been devised, such as the concept of a metagrammar which is described in further text.

Parsing of MRLs has been recently paid increasing interest in the parsing community as tools primarily devised for analytic languages like English continuously recorded significant performance losses after being ported to a morphologically-rich language. A good overview of the state-of-the-art can be found in [Tsarfaty et al., 2013].

3.10 Parsing large vs. parsing small

I would like to introduce two new concepts related to practical parsing that I call *parsing large* and *parsing small*. The latter one stands for the traditional approaches to parsing where the parser gets a single sentence as input and produces its parse (in most cases a tree) as the output, where the parse aims to be a full syntactic description of the sentence. Unlike this, I refer to *parsing large* where the task is to process large volumes of natural language text in order to extract some common knowledge, such as topic or genre type of the text, trends or collocations.

Clearly, tools doing parsing small can be used for all the tasks doing parsing large as well, but not vice versa. However, they are usually not because parser-based approaches to these tasks do not tend to outperform simpler approaches. The difference between parsing small and parsing large is in the focus on precision in the former case and on the recall in the latter

case. I exploit the terms precision and recall here because usually these tasks involve (or at least can be thought of as they would involve) extraction of instances of particular syntactic phaenomena (say, subjects of a verb or noun phrases or clauses), i.e. classification of phaenomena into certain groups.

In the simplest case one can think of providing dependency labels for every word in the text: when doing parsing small, the parser's recall is 100% (it assigns a dependency to each word) on the cost of the precision. If the parser would be designed in a way that it may choose not to assign any dependency label in some cases (e.g. where the rules and stats do not give high probability estimates for any of the possible labels), it may yield higher precision. But parsers are usually not designed this way, and mainly, not evaluated this way. Hence, precision is always compromised.

On the other hand, any tool that takes as input large amounts (e.g. billions of words) of text may freely choose to relax the recall, trading it off for precision, because high recall is likely to be achieved through the volume of the data processed.

In further two chapters I will present two systems I have been co-developing that deal with this two approaches to parsing. Chapter 4 discusses word sketches and Sketch Engine as an instance of the *parsing large* approach and in Chapter 5 it is the Synt system, a stochastic rule-based parser built on top of the PCFG framework. Each of these chapters has its own evaluation and both are concluded finally in Chapter 6.

Chapter 4

Parsing large: word sketches

4.1 Overview

In this chapter I present the concept of word sketches and the associated parsing machinery as a representative of the *parsing large* approach consisting of processing large amounts of texts (indexed in the form of a text corpus) with syntactic rules based on regular expressions over morphological annotation. First I introduce the concept of word sketches, the associated formalism (a word sketch grammar) and implementation. I further present extensions and improvements to the parsing system which I have been working on.

Finally I close this chapter by discussing a series of evaluations of word sketches on a collocation extraction task including a comparison with an approach based on traditional parsing systems performing *parsing small* sentence by sentence. This evaluation shows that, on contrary to common assumptions, these systems do not outperform their word sketch based counterpart which is not only (slightly) more accurate, but also simpler, faster and easier to extend to further languages.

4.2 The concept of word sketches

Word sketches, one-page summaries of a word's collocational behaviour in terms of separate syntactic relations, were first introduced in Sketch Engine [Kilgarriff et al., 2004, Kilgarriff et al., 2014a], a corpus management system devised primarily for lexicographic purposes. An example word sketch table for the English noun *resource* is given in Figure 4.1. Conceptually word sketches represent an instance of dependency syntax: the word sketch tables are computed from a set of collocation candidates in the form of triples consisting of a headword, a relation and a collocate. Each of these triples is ranked with an association score, corpus frequency and link to corpus concordances.

These triples are generated as corpus queries in the Corpus Query Language (CQL, [Jakubíček et al., 2010]), which are basically regular expressions

resource *(noun)*
English Web 2013 (enTenTen13) freq = 3,676,811 (161.76 per million)

modifiers of "resource"		
	2,176,650	59.20
natural ✚	119,441	9.48
valuable ✚	38,150	8.74
human ✚	61,931	8.46
limited ✚	32,178	8.39
financial ✚	45,532	8.04
available ✚	22,791	7.93
renewable ✚	17,545	7.86
water ✚	31,374	7.62
scarce ✚	13,316	7.61
online ✚	44,159	7.56
useful ✚	16,062	7.50
educational ✚	16,916	7.49
additional ✚	23,843	7.32
energy ✚	21,956	7.31
precious ✚	11,299	7.14
invaluable ✚	9,649	7.12
information ✚	15,602	7.08
mineral ✚	9,135	6.94
excellent ✚	15,727	6.93
community ✚	14,747	6.70
helpful ✚	7,748	6.66
great ✚	49,419	6.65
necessary ✚	7,656	6.40
free ✚	16,622	6.36
vast ✚	7,236	6.34

nouns modified by "resource"		
	716,539	19.49
box ✚	53,715	9.45
management ✚	38,039	8.44
allocation ✚	8,746	8.38
center ✚	38,027	7.64
planning ✚	8,472	7.42
conservation ✚	4,448	7.40
utilization ✚	3,383	7.11
centre ✚	10,826	7.07
constraint ✚	3,288	6.91
efficiency ✚	4,477	6.87
extraction ✚	2,863	6.86
guide ✚	7,445	6.82
depletion ✚	2,200	6.59
manager ✚	9,224	6.54
development ✚	11,525	6.49
estimate ✚	3,001	6.49
sharing ✚	2,378	6.41
sector ✚	5,461	6.33
department ✚	7,843	6.25
consumption ✚	3,066	6.25
base ✚	5,020	6.21
usage ✚	2,487	6.19
locator ✚	1,677	6.18
scarcity ✚	1,611	6.16
library ✚	3,936	6.14

verbs with "resource" as object		
	1,148,603	31.24
allocate ✚	16,066	8.63
conserve ✚	7,927	7.67
pool ✚	6,993	7.59
utilize ✚	11,083	7.43
access ✚	9,568	7.39
manage ✚	14,204	7.29
devote ✚	6,714	7.24
invest ✚	6,989	7.23
waste ✚	7,364	7.20
provide ✚	54,500	7.10
lack ✚	6,492	7.01
limit ✚	8,753	7.01
share ✚	14,066	6.90
focus ✚	6,838	6.87
exploit ✚	4,134	6.68
protect ✚	9,147	6.67
consume ✚	5,021	6.67
leverage ✚	4,100	6.66
need ✚	21,280	6.60
dedicate ✚	5,023	6.59
mobilize ✚	3,519	6.57
deplete ✚	3,359	6.49
use ✚	66,577	6.47
divert ✚	3,274	6.44
require ✚	15,627	6.34

verbs with "resource" as subject		
	316,158	8.60
include ✚	13,066	5.89
permit ✚	851	5.73
consult ✚	521	5.48
dwindle ✚	465	5.46
exist ✚	1,519	5.45
provide ✚	5,550	5.22
limit ✚	579	5.13
support ✚	1,478	4.98
plan ✚	1,086	4.96
allow ✚	2,299	4.94
relate ✚	855	4.92
enable ✚	813	4.90
pertain ✚	504	4.89
guard ✚	317	4.78
list ✚	455	4.65
back ✚	436	4.65
help ✚	2,076	4.64
deplete ✚	255	4.64
link ✚	405	4.63
need ✚	3,388	4.62
concern ✚	704	4.59
cover ✚	1,053	4.56
pool ✚	239	4.55
become ✚	3,363	4.52

Figure 4.1: Part of a word sketch table from the English corpus enTenTen 2013 [Jakubíček et al., 2013] for the noun *resource* showing its collocates clustered into the corresponding grammatical relations: verbs that *resource* is object and subject of; and nouns which are modifying or modified by *resource*.

over (morphologically annotated) corpus tokens. Each grammatical relation is generated by one or more queries, which all together are called a *sketch grammar*. The word sketch triples are further subject to scoring by a lexicographic association score (in the case of Sketch Engine, it is the logDice [Rychlý, 2008], but the system can handle other measures like T-score or MI-score). These scores are then indexed in a dedicated database which is part of Sketch Engine, together with the record of the occurrences of the particular word sketch triple in the corpus (a so called inverted, or reversed, index), which is accessible as a concordance (see Figure 4.2).

The word sketches serve as the backbone for a number of related functions:

- **distributional thesaurus** A distributional thesaurus is calculated

governments need to ***allocate*** adequate financial **resources** in national budgets development of capacities
decisions as to where to ***allocate*** your marketing **resources** . The model asks for your monthly ... read
rather than written to files. Destroy the **resources** ***allocated*** by GMT modules to hold results
which returns a pointer to the ***allocated*** **resource** . Pass family as one of GMT_IS_GRID, GMT_IS_IMAGE
which returns a pointer to the ***allocated*** **resource** . Specify which family and select mode from
awareness, enforce legislation, ***allocate*** **resources** to protect women and girls, and promote
gets full, the file system shall reuse the **resources** ***allocated*** to the oldest trace events that
event. The trace system shall reuse the **resources** ***allocated*** to any trace events already reported-see
populated with new tasks, specifically ***allocate*** **resources** to handhold potential volunteers. Allocate
this discussion, routinely do not ***allocate*** **resources** to creating attractive designs. For whatever
DOE must analyze how to ***allocate*** scarce **resources** . "If there is a need for a school we would
invisible to our users: we've ***allocated*** more **resources** to Tracker development, and increasingly
consideration in combating crime; ***allocating*** **resources** to make punishment more severe or more
Sponsored Post Strategy gamers! ***Allocate*** your **resources** toward these two titles: SimCity and Europa
and private funds. ***Allocating*** sufficient **resources** in a smart way is necessary in order to
She sees this as driving up the amount of **resources** ***allocated*** on discrimination that has already
EEOC would be better off ***allocating*** those **resources** on prevention mechanisms. Commissioner
The question for us is: will we ***allocate*** **resources** properly to help those in need? We have
few of them put in the time, ***allocate*** the **resources** , or even understand the intricacies and
balance sheet will help show you how your **resources** are ***allocated*** and also give a little more

Figure 4.2: Source concordance for the word sketch triple for the noun *resource* as object of the verb *allocate*.

by looking for words with most similar collocates in the same grammatical relations as the given searched keyword (see Figure 4.3). A similarity score is computed for all but low frequent corpus tokens using an efficient algorithm given in [Rychlý and Kilgarriff, 2007].

- **word sketch difference** Word sketch difference compares two word sketches for the given two lemmas, for two word forms of the same lemma or for one lemma compared between two subcorpora.

- **bilingual word sketches** A bilingual word sketch compares the word sketches for a translation pair, either manually given by the user or with a translation guessed using a bilingual dictionary automatically built from large parallel corpora.

- **multi word sketch** The concept of word sketches has been extended for multi words by transitively interconnecting word sketch data. This way does not need a firm definition of a multi word – instead it works for any multi word expression whose components are interconnected using a word sketch relation.

test *(noun)* Alternative PoS: verb (freq: 941,372)
enTenTen [2012] freq = 1,915,482 (147.70 per million)

Lemma	Score	Freq
testing	0.520	558,727
assessment	0.410	640,347
analysis	0.399	1,196,660
procedure	0.382	1,311,372
study	0.380	3,090,402
method	0.373	2,760,051
application	0.366	3,171,582
program	0.365	6,442,955
datum	0.362	3,165,540
evaluation	0.360	468,130
model	0.357	2,557,538
training	0.354	2,486,409
research	0.354	3,171,715
examination	0.352	375,991
requirement	0.349	1,734,482
exam	0.349	373,769
review	0.348	1,803,362

Figure 4.3: Sample thesaurus word cloud for the English noun *test* – as a table and word cloud.

All these functions are computed fully automatically on top of the word sketches without any manual intervention.[1]

In the following, I first describe the CQL and then the sketch grammar formalism which uses CQL for generating the word sketch collocation candidates.

4.3 CQL

The Corpus Query Language (CQL) is a formalism that was originally devised by [Christ and Schulze, 1994] and substantially extended during Manatee development which it is part of and which is described in detail in Section 4.6.[2] The basic structure of a query is a list of attribute-value constraints over positional and structural attributes of the corpus, such as:

```
<doc year="2000"> [word="blue"] [word="car"]
```

This query would match the phrase consisting of word forms *blue car* at the beginning of documents from the year 2000. The values can be regular expressions[3] and a single corpus position can have multiple constraints combined with Boolean operators, e.g.

1. Details on calculations of these functions are provided in https://www.sketchengine.co.uk/documentation/statistics-used-in-sketch-engine/.
2. Complete documentation on the CQL can be found at: https://www.sketchengine.co.uk/documentation/corpus-querying/
3. The regular expressions are as defined by the PCRE library (http://www.pcre.org),

```
[(word="state" & tag="noun") | lemma="country"]
```

I extended the language with a *within* and *containing* operator as described in [Jakubíček et al., 2010]. This allowed for easier formulation of queries that require free word-order, such as when looking up sentences with arbitrarily positioned constituents:

```
((<s/> containing <CONSTITUENT1>)
       containing <CONSTITUENT2>)
       containing <CONSTITUENT3>
```

These were first extensively used for looking up idioms in Czech corpora by [Bušta, 2008].

Further on, I have incorporated both the word sketch index and the (from the former derived) distributional thesaurus index in the CQL. The former was exploited for the extensions of the word sketch formalism to multi-word units of arbitrary length, called multi-word sketches [Kilgarriff et al., 2012], as shown in Figure 4.4, the latter to enhance the search expressivity by search for semantically related words shown in Figure 4.5. Both of these can be also utilized for the development and enhancements of existing sketch grammars.

4.4 Sketch grammar formalism

A sketch grammar is defined using a simple plain text format:

- lines starting with `#` are comments
- lines starting with `*` are directives influencing query processing
- lines starting with = define beginning of a grammatical relation
- all other lines are CQL queries to be evaluated; each query contains a single token labeled `1:` which stands for the headword, and a single token labeled `2:` which stands for the collocate.

A short example of a part of the English sketch grammar is provided in Figure 4.6. The simplicity of the sketch grammar formalism allows researchers to devise their own definitions, e.g. to cover other types of relations. To increase the readability and ease the maintenance, sketch grammars are sometimes defined using the `m4` macro formalism,[4] which allows automatic

optionally the system can use the ICU libraries (`http://icu-project.org`) or system-defined POSIX regular expressions.

4. See `https://www.gnu.org/software/m4/m4.html`.

days for day 3 transfer), we *do* a *blood* **test**	to find out if she is pregnant. If embryo
pregnancy test that was negative. A *blood* **test**	is *needed* because it is much more sensitive
addition, he would have to *take* a *blood* **test**	which required fasting from 10 o'clock
severity. Your doctor may *order* routine *blood* **tests**	such as B12 and thyroid hormone level to
an individual. This will *include* *blood* **tests**	, ECG and MRSA screening. These tests are
questions. You will also *need* some extra *blood* **tests**	, which you can organise through your GP
Cholesterol Education Program *recommends* a *blood* **test**	every five years, starting when people
lithium. Your GP will ask you to *have* *blood* **tests**	to make sure you're getting the right dose
hives. Your doctor may also *perform* *blood* **tests**	or skin tests to determine what may have
miscarriage had a hand in it. There *'s* no *blood* **test**	you can take, and frankly, a lot of doctors
will probably be asked to *have* a *blood* **test**	and you may be asked to provide a sample
damage. He or she may *order* routine *blood* **tests**	to check for anemia, iron or vitamin deficiency
make a diagnosis of Crohn's disease. *Blood* **tests**	are also commonly *done* to provide helpful
history and a clinical exam *including* *blood* **tests**	, can reveal whether someone has HE. It's
and organs. Common examples *are* *blood* **tests**	and skin biopsies. X-rays are used in radiography
protocol will stay at home and *need* only *blood* **tests**	, as directed by a staff designed to follow

Figure 4.4: Concordance for the multiword word sketch combining the noun *test* modified by the noun *blood* with verbs having *test* as object. The source query is `[ccoll(1,2,ws("test-n","verbs with \"%w\" as object",".*")) & ws("test-n","modifiers of \"%w\"","blood-n")]`. The `ccoll` operator merely renames the collocate from the query, normally labeled as 1, to be highlighted separately in the resulting concordance.

excellent skills and extensive experience to	**perform this procedure**	, a surgical joining of the vas deferens
produce eggs. percutaneous: To place or	**perform a procedure**	underneath the skin. No incision (cutting
diagnostic tests, prescribe pharmaceuticals and	**perform procedures**	. The College currently offers three specialty
patients over 9 years from vasectomy. I do not	**perform this procedure**	in my practice, but advise patients to
website, and that offered the lowest price to	**perform the procedure**	. Everything seemed to be too good to be
the right doctor or specialist who will	**perform a procedure**	such as a vasectomy or vasectomy reversal
of each of the wife's eggs. We routinely	**perform this procedure**	for patients who have congenital absence
neuromodulatory procedures. One goal is to	**perform studies**	that will produce evidence-based data to
up to 90% SUPPOSE a physician is asked to	**perform a procedure**	that has no recognized medical value and
you will turn off or on frequently as you	**perform test**	renderings of a scene you are developing
all that's required. A convenient time to	**perform these tests**	may be at your cat's annual vaccination
pull in cross-site scripts, in order to	**perform the testing**	. If this message does not go away, it means
defining the actors (a.k.a. users) that	**perform those use**	cases. You can (and should) use an actor
means as least 12 clinics won't be able to	**perform the procedure**	starting as soon as Friday. Some Texans
up and running quickly. Test Accounts To	**perform tests**	through any device, application, shopping
have your tracking link from Tracking202 to	**perform this test**	. Written By Cameron Stewart As VP of Global

Figure 4.5: Concordance for the input query `"perform" [tag="DT"]? ~"test-n"` exploiting thesaurus items for the noun *test*

```
*DUAL
=modifiers of "%w"/nouns modified by "%w"
2:"(JJ.*|N.*[^Z])" MODIFIER{0,3} NOUN{0,2} 1:NOUN NOT_NOUN

*DUAL
=modifiers of "%w"/adjectives modified by "%w"
2:ADVERB 1:ADJECTIVE

*DUAL
=modifiers of "%w"/verbs modified by "%w"
2:ADVERB 1:VERB
1:VERB ADVERB{0,2} 2:ADVERB [tag!="RB.*" & tag!="JJ.*"]
```

Figure 4.6: Part of the English sketch grammar based on the Penn Treebank tagset. It describes three relations covering nouns, verbs and adjectives in the modifier relation. The `*DUAL` directives instructs the parser to interchange headword and collocate and split the name of relation by the slash. The part before the slash is the name of relation for the query as given in the file, the part after is used for the interchanged query. This way a single query is evaluated covering both directions of the relation. A full description of the formalism can be found at https://www.sketchengine.co.uk/documentation/writing-sketch-grammar/.

expansion of named macros used e.g. for repeated occurrences of the same morphological tags or complex regular expressions.

4.5 Handling rich-morphology

The CQL-based sketch grammar formalism can handle arbitrary morphological annotation. In Sketch Engine it has been applied to diverse tagsets for dozens of languages. It allows regular expression queries to be formulated over arbitrary annotation in the corpus (word forms, lemmas, part-of-speech tags etc.). The Manatee library also includes the capability of deriving one corpus attribute from another. This is regularly used for deriving lowercase word forms or lowercase lemmas automatically, but it can also be used for decomposing a complex morphological tag into individual grammatical categories.

In the Czech tagset (as given by the Majka analyzer) for instance, substantives are usually assigned at least 4 categories: the part-of-speech (kind, 1 in case of substantives), case (values 1–7), number (singular `S` or plural `P`) and gender (feminine `F`, neuter `N`, masculine animate `M` and inanimate `I`). Thus a tag like `k1c2nSgF` would indicate any substantive in genitive case, singular number and feminine gender. To account for agreement in the CQL queries, one would normally need to explicitly list all agreement com-

binations which is tedious and very inefficient as it multiplies the number of executed queries. Just to handle a case agreement in a prepositional phrase consisting of a preposition (`k7`) followed by a substantive, there would need to be 7 queries:

```
[tag="k7c1"] [tag="k1c1.*"]
[tag="k7c2"] [tag="k1c2.*"]
...
[tag="k7c7"] [tag="k1c7.*"]
```

With a designated attribute for each category (which can be dynamically derived from the complex tag), CQL allows appending global conditions across the whole query. Thus a single query is then an equivalent of the 7 queries above:

```
1:[k="k7"] 2:[k="k1"] & 1.c = 2.c
```

4.6 Implementation

4.6.1 Manatee

Word sketches, i.e. the parsing machinery and dedicated database management with indexing, are implemented within the Manatee corpus management system [Rychlý, 2000, Rychlý, 2007] which I was co-developing from 2009. Manatee represents the backend of Sketch Engine and provides facilities for efficient indexing (compiling) and searching billion-word-sized corpora [Pomikálek et al., 2012]. It is a C++ library with bindings for Python (and other languages which are however not used within Sketch Engine) and command-line tools for writing and reading the indices. Manatee works in any POSIX-compatible operating system, normally some Linux distribution.

From a formal perspective a corpus in Manatee consists of *text data* (called tokens or positions, each of which may be associated with a number of attributes such as word, tag or lemma, further referred to as positional attributes) and *text metadata* (called structures, each of which is denoting a span in the corpus such as a document, paragraph or sentence, and may be associated with arbitrary number of structure attributes, denoting e.g. the author of a document, date of creation etc.).

Every positional or structural attribute possesses following basic index structures:

- **attribute lexicon** providing efficient string $\leftrightarrow$ ID mapping. Each unique attribute string value is assigned a unique numeric ID which is further used in all indices and for processing CQL queries,

- **attribute inverted (reversed) index** providing sequential access to a sorted list of occurrences (corpus positions) of a given attribute ID,
- **attribute text** storing the actual text of this corpus attribute, i.e. a sequence of attribute IDs in the order of occurrence in the corpus.

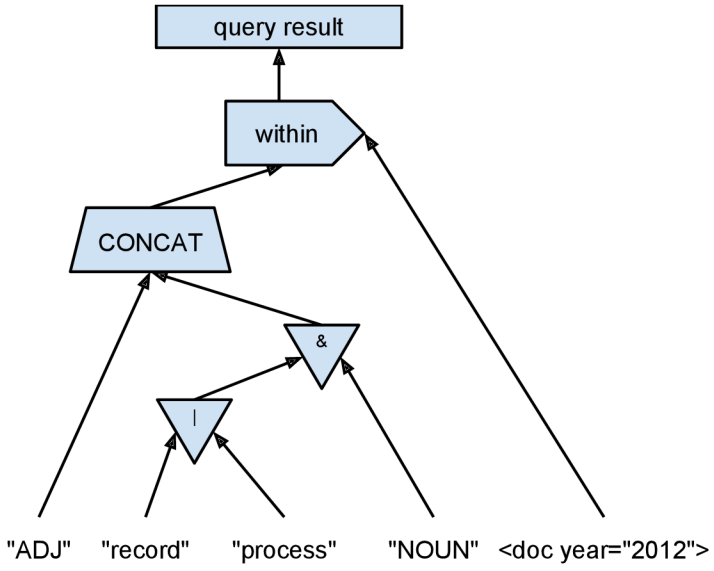


Figure 4.7: Evaluation of a sample CQL query
`[tag="ADJ"] [(word="record" | word="process") & tag="NOUN"] within <doc year="2012"/>`
in terms of an abstract syntax tree.

On a very abstract level evaluating a CQL query consists of mapping attribute strings given in the query to their IDs (using the lexicon index), retrieving the relevant positions from the inverted index, combining them according to the given CQL operators and displaying the results (e.g. in the form of a concordance). In the simplest case, considering a query looking for all occurrences of a single word in corpus like `[word="someword"]`, one yields the ID n of `someword` from the lexicon and then retrieves the sorted list of positions for n from the inverted index. Further CQL operations always rely on processing sorted streams (of corpus positions or attribute

IDs). Finally the resulting stream of positions is extended by its context through lookup in the text index and retrieved text index IDs are converted to strings to be displayed to the user using the lexicon.

The CQL language is based on streaming queries: all CQL expressions operate on sorted corpus positions (called `FastStream` and primarily retrieved from the inverted index) or sorted pairs of corpus positions (called `RangeStream` and primarily retrieved from corpus metadata indices or combination of `FastStream`s). This evaluation process is illustrated for a sample query in Figure 4.7 in the form of an abstract syntax tree of the CQL expression and in Figure 4.8 for the same query with tree nodes translated to particular `FastStream` and `RangeStream` classes.

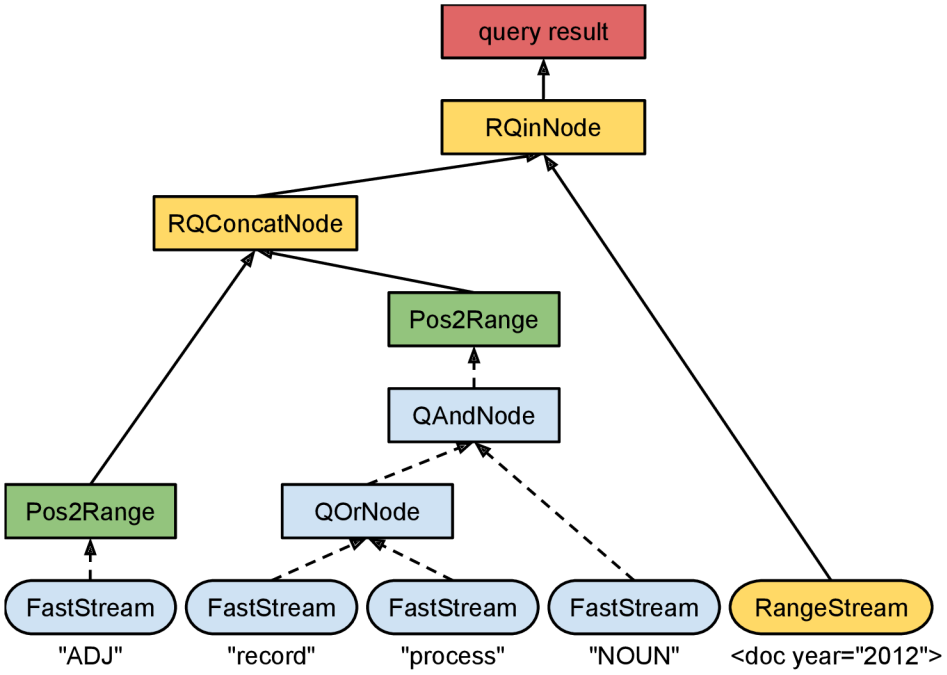


Figure 4.8: Evaluation of the same query as in Figure 4.7 in terms of an abstract syntax tree that exploits particular classes of streaming queries. First four leaf nodes are `FastStream`s obtained from the inverted index for the respective attribute (tag and word). Last leaf node is obtained from the metadata index for the document structure and its year attribute. Blue nodes operate on `FastStream`s, orange nodes operate on `RangeStream`s and green `Pos2Range` node represent an adapter class that converts a `FastStream` to a `RangeStream` by merely creating identity pairs: for a position i, it emits a pair (i, i). The key property of all streams is that they are sorted which enables very efficient implementation of the operators.

4.6.2 Generating Word Sketches

The word sketch generation, in particular, is handled by three command-line tools. The first one is doing the parsing and generating of (unscored) word sketch triples (hence is called `genws`): it processes the sketch grammar, evaluates all queries and prints found word sketch triples together with the position of both headword and collocate in binary (hence, a 5-tuple of 64bit binary integers) on its standard output.

The second tool reads the binary output from `genws`, merges the generated word sketch triples, sorts them and writes a designated it into a designated index called a wordmap. Therefore the tool is called `mkwmap`.

Finally, a third tool called `mkwmrank` computes the scores for every triple ("ranks" the wordmap) and stores them into the index which has a hierarchical structure and allows fast retrieval of all headwords, all grammatical relations for a given headword, all word sketch triples for a given headword-relation pair and eventually all corpus occurrences for a given word sketch triple.

The overall workflow in the form of a POSIX pipe looks as follows:

```
genws <CORPUS> <ATTR> <DATAPATH> <SKETCHGRAMMAR>
| mkwmap <DATAPATH> ; mkwmrank <DATAPATH>
```

where `CORPUS` is the configuration file of the indexed corpus, `ATTR` is the corpus positional attribute to be used for the evaluation (usually lemma or a combination of lemma and part-of-speech; but technically any attribute, including a raw word form, works), `DATAPATH` is the path where the indices are written to and `SKETCHGRAMAR` is the sketch grammar definition file in the format explained in Section 4.4.

The described workflow, besides taking advantage of the Unix-like elegance of the "one-tool has one-task" principle, also means that all collocation candidates, normally generated by `genws` using the given sketch grammar, can be also generated by any other tool that can output the simple binary 5-tuple format. Thus, the word sketches can be also compiled from the output of any other parser.

First experiments with a standard parser used for word sketch calculations were conducted in [Horák and Rychlý, 2009] for Czech and in [Ambati et al., 2012] for Turkish.

In 2012 I developed a tool[5] for generating the word sketch index from a corpus possessing dependency annotation in the so called CoNLL format (see [Buchholz and Marsi, 2006]) which we later used for comparing the

5. Full documentation available at https://www.sketchengine.co.uk/documentation/building-sketches-from-parsed-corpora/.

quality of parser-based word sketches with sketch grammar-based word sketches, as described in Section 4.10, and also for building word sketches from 3-billion-word corpus of Portuguese [Kilgarriff et al., 2014b] parsed by the Palavras parser [Bick, 2000].

4.7 Scalability: Parallelization

The speed of parsing word sketches depends on the size of the corpus and on the number and complexity of the grammatical relations. The input for the sketch grammar parsing is an already indexed text corpus and the parsing exploits preexisting indices, mainly the inverted index. Therefore, when evaluating the speed of parsing, the time of compiling the corpus indices needs to be counted in. Any parallelization efforts were therefore devoted both to corpus compilation as well as word sketch computation as described in further two subsections.

4.7.1 Corpus compilation

4.7.1.1 Corpus virtualization

The corpus compilation is a complex indexing process where mutually dependent indices (lexicon, inverted index and text; as described before) are created, therefore trivial parallelization is not possible. Since 2013 I collaborated on an indexing strategy called *corpus virtualization* [Jakubíček et al., 2014] which allows combining arbitrary corpora into a virtual corpus that can be used like any regular corpus within the Manatee system.

The virtualization of corpora allows significant space- and time-savings and also addresses current operational corpus management problems with combining large datasets. Current situation in corpus linguistics is to some extent similar to that in IT before hardware virtualization: for many languages there are large text collections available (see e.g. [Jakubíček et al., 2013, Callan et al., 2009, Pomikálek et al., 2012]) and one has to decide how these will be organized into corpora at the technical level, i.e. as independent database units resulting from a (possibly costly, both in terms of runtime and final space occupation) indexing procedure.

While we presume that the corpus axiom is: the bigger the better, clearly having just a single huge corpus per language is not always desirable for obvious practical reasons – smaller data is always faster to process, one corpus implies one annotation scheme which would then be very limited, and finally one might just find himself in a situation where the subject of

corpus	number of tokens (billions)	database size (gigabytes)
esAmTenTen11	8.7	217
esEuTenTen11	2.4	35
esTenTen11	11.1	252

Table 4.1: Overview of the esTenTen corpus and its parts.

	virtual	regular
space occupied	13 GB	252 GB
compilation time	3.4 hrs	30.6 hrs

Table 4.2: Comparison of the esTenTen being compiled as a virtual and regular corpus.

studies would be a portion of the language (possibly defined using complex constraints).

The obvious solution to this problem lying in creating separate and independent corpora for any combination of needs becomes less and less feasible for very large datasets. This is being addressed by the virtual corpus features in Manatee, a method allowing flexible management of corpora into logical units.

A virtual corpus is defined as a set of segments from one or more corpora. A virtual corpus therefore might be used just for a subcorpus as well if the segments originate from a single corpus – but in most cases this will not be the case and a virtual corpus will rather be a *supercorpus* in this respect. Compilation of a virtual corpus consists mainly of providing a new lexicon and mappings to all existing lexicons of the source corpora. Apart of that only the preexisting indices of those corpora are used for query evaluation resulting in large storage savings while having negligible influence on the query evaluation performance.

Experiments we have conducted show the advantages of virtual corpora as opposed to the regular ones on the example of the Spanish esTenTen corpus [Kilgarriff and Renau, 2013] consisting of two substantial parts, the esEuTenTen (European Spanish) and esAmTenTen (American Spanish) as given in Table 4.1. In Table 4.2 a comparison of its compilation in both variants is provided: as a virtual corpus consisting of two regular corpora and as a single regular corpus. As can be seen, the virtual corpus approach

achieves space savings by factor of more than 20 and time saving by factor of more than 10.

4.7.1.2 Parallelization through virtualization

I exploited the corpus virtualization to design an effective workflow for parallelization of the whole corpus compilation, where a single corpus is:

1. compiled as N independent parts
2. a virtual corpus is built out of these N parts
3. finally the virtual corpus is devirtualized

In the first phase, the source text for the corpus is being read and on the fly split into the N compilation processes in a round-robin fashion. In the second phase the virtualization takes care of creating compatible index mappings from all the N parts into a virtual corpus. Finally, these index mappings are used to create a regular (non-virtual) corpus.[6]

I have carried out experiments with two differently sized corpora: with the British National Corpus [Leech, 1992], which has – by current measures – a rather modest size of about 110M tokens but has very rich textual metadata, and a large web corpus enTenTen12 [Jakubíček et al., 2013], which has almost 13 billion tokens. In Figures 4.9 and 4.10 I give an overview of the speedup achieved by varying the number of parallel jobs being run. All measurements were run on commodity hardware servers equipped with two Intel Xeon E5-2630 v4 CPUs (40 cores total), 128 GB RAM and rotational hard drives. In the case of BNC, the maximum speedup is about 4 with the 10 parallel processes. As the corpus is quite small, adding further parallelization clearly does not help because the individual parts are then too small.

In the case of the enTenTen12 corpus the maximum speedup is slightly less than 3 (in case of using 20 parallel jobs). This is partially due to the fact that lot of compilation time is taken by computing attributes that are derived automatically from those present source text. These include dynamic attributes, regular expression optimization attributes (both described in more detail in Section 4.8 and normalization attributes which allow to map from one attribute to another (and hereby are used e.g. to normalize word forms to lemmas). These automatically derived attributes are usually not very many

6. The devirtualization procedure consists of a single shell script running many other helper tools for manipulating the indices. It makes use of GNU parallel [Tange, 2011].

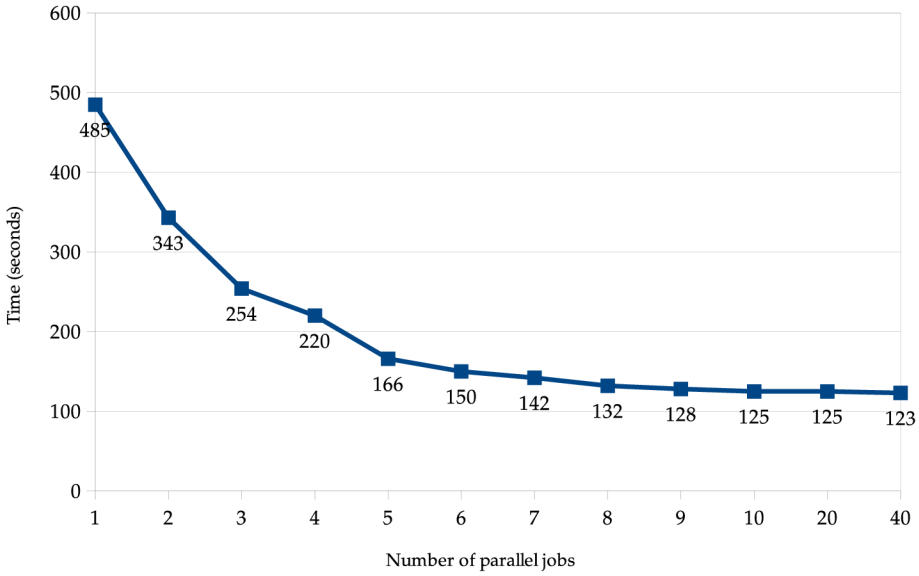


Figure 4.9: Evaluation of corpus compilation parallelization for the British National Corpus, which has about 113 million tokens.

(usually less than 10) and are compiled on top of the existing corpus, hence represent a bottleneck for parallelization. Discounting them yields a speedup of more than 4.5, which confirms the identification of this bottleneck.

4.7.2 Word sketch compilation

The most expensive part of compilation of word sketches is of course the actual parsing, i.e. evaluating of corpus queries by the `genws` tool. Subsequent indexing with `mkwmap` usually takes less than 10 % of the whole time, and scoring of the collocations with `mkwmrank` less than 2 %. On the British National Corpus, the whole word sketch compilation took 19.5 minutes, out of which `genws` took slightly over 17 minutes, `mkwmap` about 2 minutes and `mkwmrank` less than half a minute.

Therefore any aims at speeding up the compilation target the corpus parsing. Fortunately in most cases individual relations and queries are independent so the process can be trivially parallelized by executing them concurrently, and merging the results. Relations making use of specific fea-

7. In a sequential setup the blue plot corresponds to running the `encodevert` command and the red one to `encodevert -r -n -d`, i.e. options that disable compilation of dynamic attributes, normalization attributes and regular expression attributes.

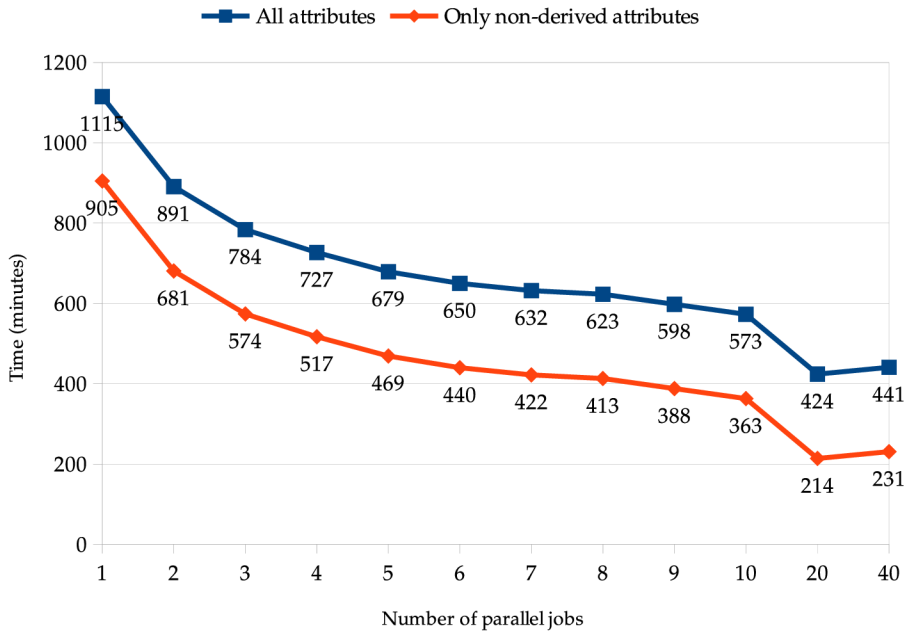


Figure 4.10: Evaluation of corpus compilation parallelization for the enTenTen12 corpus, which has about 13 billion tokens. The blue plot corresponds to a full compilation, the red plot to compilation where attributes that are derived automatically from those present in corpus source text are omitted from the compilation.[7]

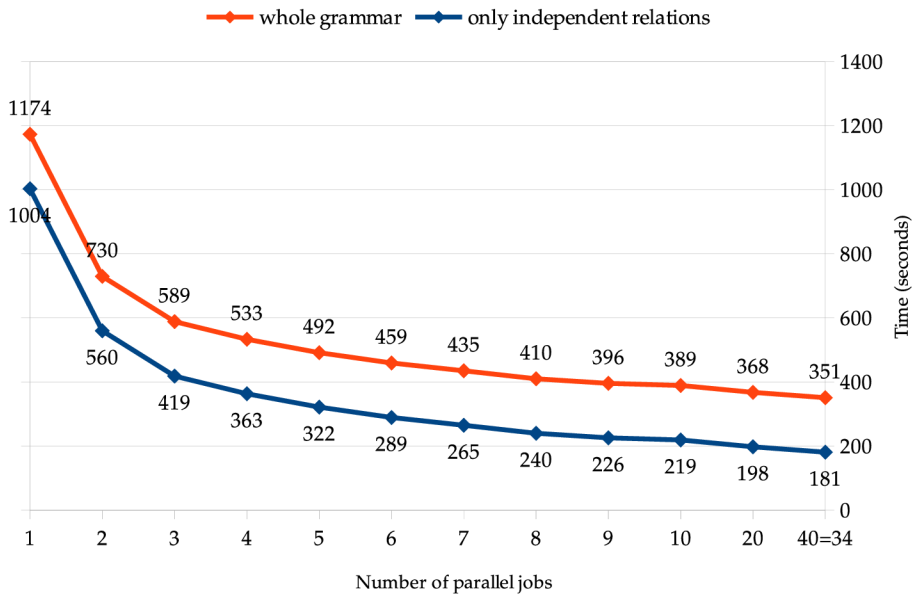


Figure 4.11: Evaluation of word sketch compilation parallelization for the British National Corpus. The sketch grammar used in that corpus has 34 independent queries, hence parallelizing in more than 34 jobs does not provide any added speedup. The evaluation also clearly shows that already with 8 parallel jobs the most of the remaining time is then spent with subsequent serial compilation of three remaining dependent queries that use the `TRINARY` directive.

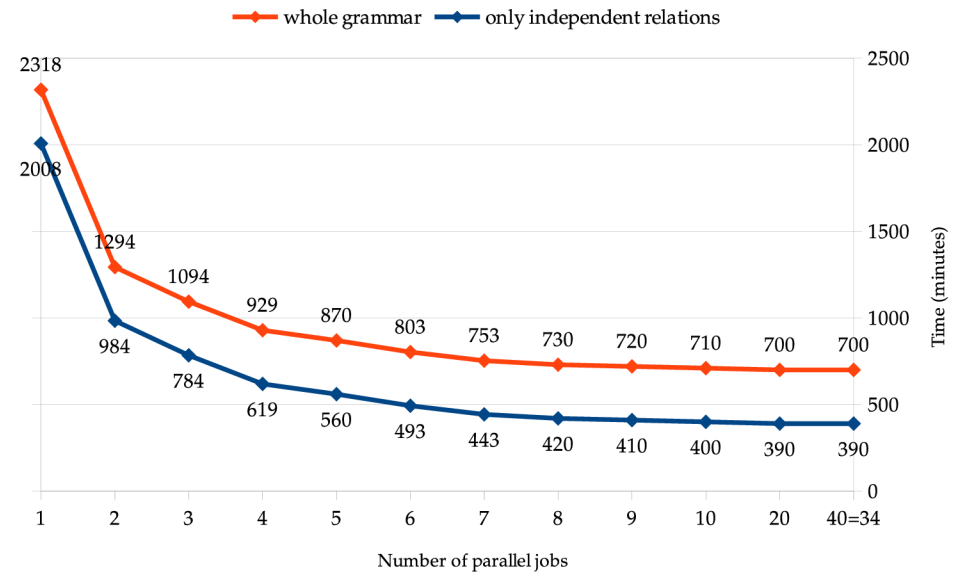


Figure 4.12: Evaluation of word sketch compilation parallelization for the enTenTen12 corpus. Again, the limit for parallelization of the used sketch grammar was at 34 concurrent jobs as explained in Figure 4.11.

tures that are mutually dependent [8] can be simply run in a single last batch after any others.

In Figure 4.11 I provide an overview of the speedup of word sketch compilation achieved on the British National Corpus and in Figure 4.12 for the enTenTen12 corpus, which shows a speedup of about 4 in both cases.

4.8 Scalability: Optimization of regular expressions

4.8.1 Basic strategies

In the case of CQL queries which contain attribute constraints given as regular expressions, the straightforward string-to-number mapping using the lexicon is not possible. First one needs to find out which attribute values are matching the given regular expression (by scanning the whole lexicon), then map each of the values to the respective ID and merge all the corresponding position streams retrieved from the inverted index. Since all the streams and results must be sorted by design, before the matching has finished no

8. Namely those using the `COLLOC` and `TRINARY` directives which modify shared lexicons of relations and created collocations, as explained in the documentation.

results are available to the user. This exhibits a serious issue for large (i.e. billion-word-sized corpora) corpora where the lexicon size is often reaching tens of millions of values and hence the time taken to evaluate the regular expressions across the whole lexicon represents a significant slow down of query evaluation, and especially of retrieving first n results.

Manatee used to have two basic optimization strategies to to tackle this problem:

- **any string optimization**: a regular expression matching `^(\.\*)+$` was omitted as it obviously must match the whole lexicon,
- **prefix optimization**: since the lexicon provides an index of attributes ID sorted alphabetically by the corresponding string values (representing one of the possible implementations of string↔ID mapping based on a simple binary search, see [Jakubíček et al., 2013] for comparison), all regular expressions containing a fixed-string prefix like `re.*` or `mis.*ing` make it possible to shrink the set of possible matching strings by selecting the range of IDs with the given prefix (here `re` and `mis`, respectively).

4.8.2 n-gram prefetching

In 2014 I have significantly extended the optimization of regular expressions, described in more detail in [Jakubíček and Rychlý, 2014]. I called the new strategy *n-gram prefetching*. It is based on the assumption that most user queries exploiting regular expressions still contain some fixed-string portions (because they are linguistically motivated). While queries like `^.{0,3}$` are of course possible, they are very rare. To verify this hypothesis we have inspected a set of 128,406 queries which the users of Sketch Engine have issued to that system over the period from June to September 2014. Only 12 of them did not contain any fixed-string portions, moreover from these 12 another 6 were `^.*$` and got optimized as well.

The idea of n-gram prefetching consists of two steps:

1. on compile time, all character uni-, bi- and trigrams of every lexicon string are indexed together with mapping to the strings where these character n-grams occur,
2. on run time (during query execution), all these character n-grams are extracted from the regular expression, looked up in the index and through the mapping a set of candidate strings is obtained,

which is usually just a fraction of the whole lexicon. Thus, the evaluation of the regular expression is much faster.

4.8.3 n-gram indexing

For indexing of the character n-grams we leverage the idea of *dynamic attributes* in Manatee. A positional or structural attribute in Manatee can be a so called *dynamic attribute* in which case such an attribute is automatically derived from another existing (regular or dynamic) attribute. Each dynamic attribute is assigned a dynamic function which takes the source attribute string as input (and optionally other parameters as well) and returns the new (dynamic) attribute value. Manatee contains a predefined set of dynamic functions which mostly focus on simple string manipulations [9] (such as getting a prefix or suffix of a string or its lowercase variant) and users can supply their own dynamic functions as well (in the form of Linux plugins – dynamically linked C/C++ shared libraries). The main benefits of a dynamic attribute are:

- **space savings in source data:** no need to make it part of the source text
- **space savings in indexing:** a dynamic attribute has only the lexicon and inverted index, but no text index. The inverted index stores for each dynamic attribute ID a sorted list of source attribute IDs which map to this dynamic attribute ID (instead of storing corpus positions).
- **very limited runtime overhead:** depending on the type of operation, the overhead (slowdown) of using a dynamic attribute instead of a regular one is very small. Optionally an index providing mappings of source attribute ID to dynamic attribute ID is compiled as well, in which case the dynamic functions do not need to be executed at runtime at all.

Each lexicon item (string) is processed generating all occurring uni-, bi- and trigrams as shown in Figure 4.13 and storing the n-gram values as a dynamic attribute of the source attribute.

The resulting dynamic attribute contains a lexicon with all the n-grams found in the source attribute and provides fast access to all source attribute

9. See https://www.sketchengine.co.uk/documentation/wiki/SkE/Config/DynamicAttributes.

```
classical  ->  c|l|a|s|s|i|c|a|l
               ^c|cl|la|as|ss|si|ic|ca|al|l$
               ^cl|cla|las|ass|ssi|sic|ica|cal|al$
```

Figure 4.13: Uni-, bi- and trigram string generation.

Table 4.3: Comparison of original and n-gram lexicon sizes

corpus	language	size $\times 10^9$	attribute	lexicon size	n-gram lexicon size
czTenTen12	Czech	5.126	word	18,978,703	522,745
			lemma	14,151,454	506,580
			tag	12,061	1,702
enTenTen12	English	12.968	word	27,894,538	1,880,911
			lemma	26,426,200	1,880,808
			tag	60	260
enClueWeb09		82.581	word	115,820,931	2,350,697
			lemma	110,606,268	2,296,072
			tag	60	260
jpTenTen11	Japanese	10.322	word	13,844,200	6,353,186
			lemma	13,303,479	3,766,160
			tag	53	297

IDs containing a given n-gram. We prepend the caret sign (`^`) and append the dollar sign (`$`) to each string so that we generate specific n-grams occurring at beginnings and ends of words.

On runtime we parse the given regular expression and extract all occurring fixed-string uni-, bi- and trigrams (preferring longer n-gram where available and combining strings longer than 3 characters into a set of trigrams by the logical `AND` operator in CQL). Since regular expression as such can be quite complicated they are analyzed using a bison-generated grammar that captures all fixed-word strings, handles logical disjunction and Kleene plus (which is expanded into bi- and tri-grams) as well as correct parsing of other metaconstructs in a regular expression.

Overall the regular expression parsing grammar is based on the following principles:

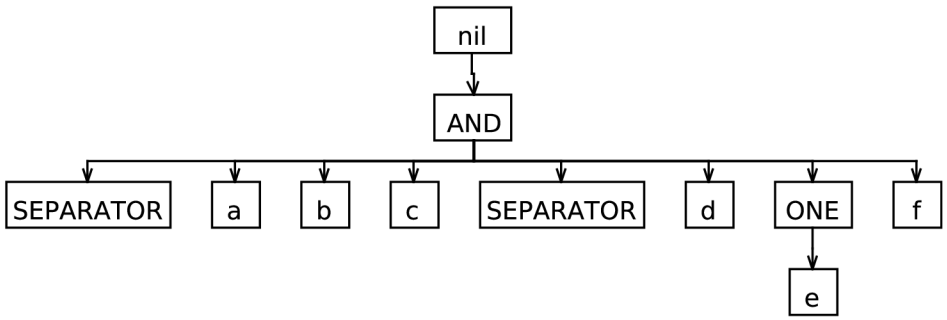


Figure 4.14: Sample abstract syntax tree for the input query `.abc.*de+f`

- it recognizes separately regular characters and metacharacters (except for ^and $ which are intentionally indexed as part of the n-grams as explained above), because metacharacters must not be included into the n-grams search and hence represent a separator between n-grams,
- it recognizes character classes (enclosed in `[` and `]` brackets) which must be kept as a single token as they represent a single character position,
- it recognizes escaped sequences so that they can be handled correctly (possibly de-escaped),
- it recognizes repetitions which must be either entirely omitted (in case zero number of repetitions is allowed and hence the respective string portion is entirely optional) or if at least one occurrence is obligatory, the operand character is duplicated and forms a suffix of previous n-grams and prefix of next n-gram. E.g. `abc+de` gets expanded into two n-grams `abc` and `cde` since every matching string must contain these.

Two sample abstract syntax trees are provided in Figures 4.14 and 4.15. The tree walking parser looks up the found n-grams (in this sample `abc`, `de` and `ef` and combines the related source attribute IDs using the logical `AND` operator into a single stream of IDs that represent possibly matchings strings. Only these IDs are then subject to evaluation of the regular expression instead of the full lexicon. An overview of the whole dataflow is presented in Figure 4.16.

The lexicon lookup for particular n-grams is either a direct mapping of an n-gram to ID using the dynamic attribute's lexicon, or in case of n-

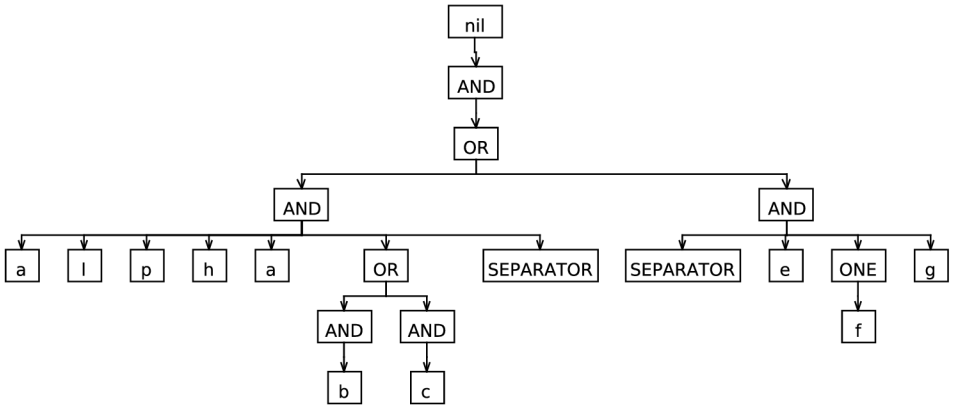


Figure 4.15: Sample abstract syntax tree for the input query `(alpha(b|c).*|ef+g)`

grams containing regular expression character class again an evaluation of a regular expression – however on a much smaller lexicon. In Table 4.3 we provide a comparison of original and n-gram lexicon sizes of various corpus and attribute combinations showing that the n-gram lexicon is usually by orders of magnitude smaller. It is obvious that for attributes with very small lexicons (such as tags based on atomic tagsets), the optimization is not worth doing and may even slow down the processing (as in the case of English and Japanese tags), therefore in the current implementation the n-gram indices are being compiled only for attributes with lexicons exceeding 10,000 items.

The optimization has been evaluated on a number of regular expressions executed against various corpora under the following conditions:

- a single evaluation thread running on an Intel Xeon CPU E5506 @2.13GHz,
- hot cache – the best time of three consecutive runs was counted,
- we have measured the time to retrieve first 20 hits – this includes the regular expression evaluation plus a number of other operations on the resulting positions stream but it is more representative from the user perspective (the processing within Sketch Engine is asynchronous and as soon as first 20 hits are available they are displayed to users),
- we provide the number of regular expression evaluations (calls to the PCRE matching function) with and without the optimization.

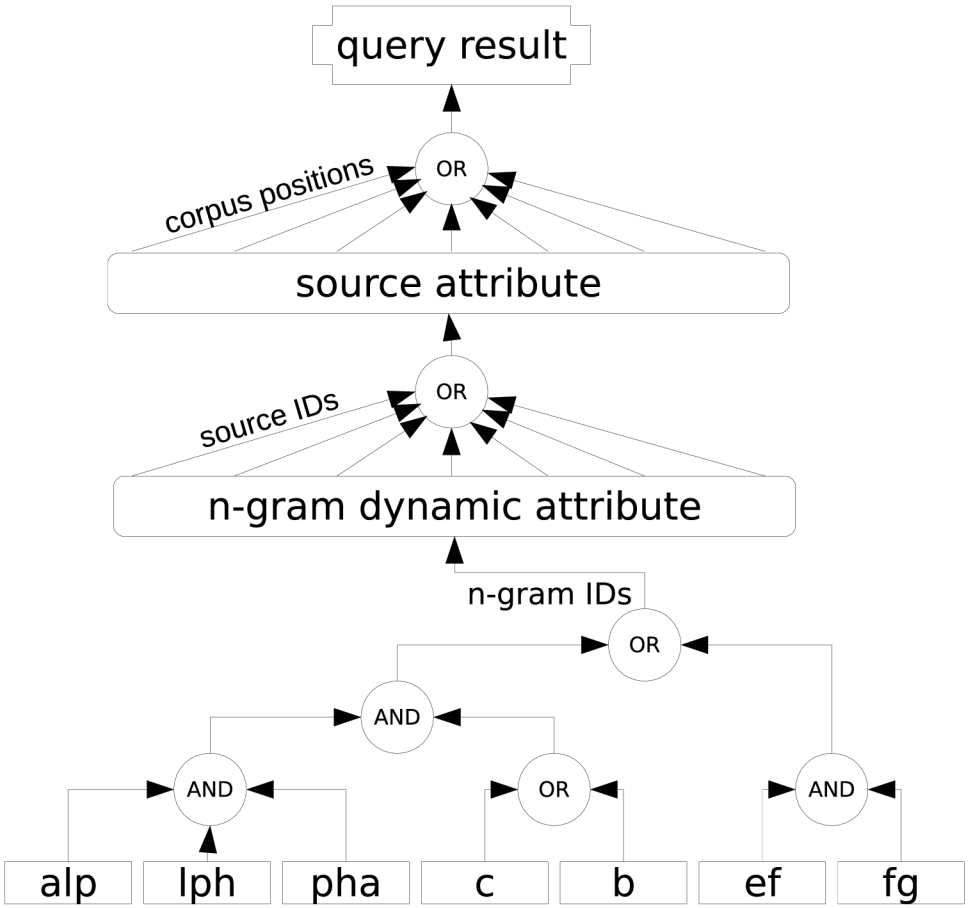


Figure 4.16: Overview of the evaluation workflow using the n-gram prefetching optimization for input query `(alpha(b|c).*|d*e+f)`

From the evaluation provided in Table 4.4 it follows that the speedup ranges from rather negligible 1.12 to enormous 100-times and the speedup ratio depends on a number of circumstances:

- obviously the larger the lexicon size of the source attribute, the more speedup can be achieved, and the evaluation shows that the additional indexing pays off only in cases where the lexicon size exceeds about 10,000 items
- for very small lexicons (e.g. in the case of the tag attribute in English corpora following the Penn Treebank tagset with only 60 atomic tags), the n-gram prefetching is not very beneficial as it enlarges the lexicon size,

Table 4.4: Evaluation of n-gram prefetching optimization. #RE denotes the number of regular expression matching function executions without and with n-gram prefetching, time is the evaluation time to acquire first 20 hits in seconds, S denotes the achieved speedup.

corpus	query	#RE w/o	#RE w/	time w/o	time w/	S
czTenTen12	`[word=".*ější"]`	18,978,703	41,426	10.03	0.34	29.4
	`[lemma=".*strč.*"]`	14,151,454	888	6.60	0.07	100.0
	`[tag="k1.*c4.*"]`	1,357	251	0.06	0.05	1.2
	`[word="[sz]p.*"]`	18,978,703	115,347	10.02	0.70	14.4
enTenTen12	`[word=".*ing"]`	27,894,538	913,004	21.93	2.77	7.9
	`[lemma=".*ten.*"]`	26,426,200	195,758	22.16	1.22	18.2
	`[word="pre.*ed"]`	80,054	7,553	0.29	0.18	1.7
	`[word="pr[oe].*"]`	251,924	198,297	1.33	0.92	1.4
	`[word=".*[dt]"]`	27,894,538	3,466,379	41.10	8.54	4.8
	`[tag="N.*"]`	60	5	0.06	0.05	1.2
jpTenTen11	`[word=".*ち.*"]`	13,844,200	30,160	8.18	0.36	22.4
	`[lemma=".*ア.*ス"]`	13,303,479	69,228	8.39	0.45	18.6
	`[word="ンテ.*"]`	17,078	17,077	0.2	0.18	1.12

- the n-gram prefetching is beneficial even in cases where the prefix optimization has already been previously in operation which is important since these two optimizations cannot be combined,[10]
- not surprisingly the n-gram prefetching is most beneficial for regular expressions containing rare n-grams (e.g. `strč` in Czech) but even for frequent n-grams (like `ten` in English) the speedup is usually around 20,
- even where the optimization itself involves regular expression evaluation (character class matching as in `[sz]p.*`), the speedup is significant and present even in comparison with prefix optimization (as in `pr[oe].*`).

4.9 Scalability: Multilinguality

Word sketches have been successfully exploited in linguistic, lexicographic and natural language processing research in numerous academic and commercial projects. These concerned a wide range of languages including

10. The prefix optimization results – by its nature – in a list of IDs sorted alphabetically, not numerically, and hence cannot be as such subject to any AND/OR stream operations.

Arabic, Chinese, Czech, Dutch, English, Estonian, German, Irish, Italian, Japanese, Slovenian and others; all together there have been sketch grammars developed for 34 languages so far.[11]

The number of languages rather high thanks to relative simplicity of the word sketch formalism and no assumptions about the morphological annotation: any corpus which is part-of-speech tagged and lemmatized can be immediately used for writing a sketch grammar.[12]

From this perspective, word sketches represent one of the most multilingual approach to syntactic processing. Individual word sketch grammars were mostly built by native speaker linguists interested in having them, with technical support of some of the Sketch Engine engineers (including myself). After ten years of developing the sketch grammars, we realized that even though the formalism is rather simple, the adoption barrier for linguists is often still too high.

In particular, we found out that the lack of formal and abstract competences often prevented linguistic experts to exploit the full potential of their linguistic introspection. This had impact both on the quality of the word sketch grammar as well as quantitative measures such as the speed of the development. After carrying out an experiment with two sketch grammars developed internally for languages where we had enough linguistic expertise (but which we were not native speakers of), namely English[13] and German[14] we largely reaffirmed ourselves that the grammar writing task is a task to be carried out at best by an informed computer scientist with a linguistic assistance of a native speaker, rather than by a native speaker linguist with a technical assistance of a computer scientist.[15]

Another related aspect of sketch grammar writing is making sketch grammars for different languages mutually compatible in terms of names of the relations, lingustic phaenomena that are covered and (from the maintenance point of view) overall structure of the sketch grammar. This is, again, difficult to establish among a large pool of grammar writers coming from different

11. Full list and references for individual languages can be found at https://www.sketchengine.co.uk/bibliography-of-sketch-engine.
12. Though in our experience reasonable outputs can be obtained from corpora only over 50 million words.
13. Written by my colleague Vojtěch Kovář and applied to the enTenTen 2013 corpus, as shown for the English noun *resource* at http://ske.li/english_resource.
14. Written by myself and applied to the deTenTen 2013 corpus, as shown for the German noun *Mittel* at http://ske.li/german_mittel.
15. This is also inline with Jelinek's *minimal supervision* principle that I discussed in Section 3.7 of the previous chapter.

backgrounds[16] and having different perspectives on the task.

For the development of bilingual functions on top of word sketches, it is however almost necessary to devise such a rather unified framework. Therefore in 2014 we started mapping individual relations across languages onto each other and, where it was necessary, rewriting the sketch grammars so that this would be possible. As a side effect we eventually hope to get a syntactic core for different language families which would speed up the sketch grammar development within the language family as well as represent a theoretic contribution to the topic of the universality of syntax across languages.

4.10 Word Sketch Accuracy Evaluation

4.10.1 2010: precision-only

First experiments with assessing the performance of word sketches were conducted in 2010 [Kilgarriff et al., 2010]. At that time the focus was only on the precision of the collocation extraction task for Dutch, English, Japanese and Slovene. The annotators were checking how many of the collocate candidates are true collocates. The evaluation dataset consisted of stratified sampled data for high frequency, mid frequency and low frequency words and the annotators were given three options to decide on a given collocation: *good*, *maybe* or *no*. In the evaluation, only collocations where all annotators (two or three depending on the language) agreed were used, and the categories *maybe* was merged into *no* to put emphasis on finding strong collocates.

The resulting precision varied from 66 (Dutch) to 87 (English) percent which, given the difficulty of the task for human annotators, was a clearly positive outcome as it shown practical usability. The missing bit was the unknown recall: how many good collocations did the system miss?

4.10.2 2012: precision and recall

Therefore, in 2012 we carried out a second evaluation [Kilgarriff et al., 2014c] targeting both precision and recall, for Czech and English. The aim was to combine whatever available resources – collocations automatically extracted corpora and those available in edited dictionaries – to create a gold standard dataset for 500 headwords (nouns, adjectives and verbs, stratifically sampled again from different frequency bands). The ultimate goal was to show that full recall was achieved as providing further collocations candidates to

16. and parts of the world

tool	sort	precision	recall	f_1	f_5
Czes2	frq	11.00	73.44	19.13	60.28
Czes2-Synt	dice	15.54	59.89	24.68	53.96
Czes2-SET	frq	9.67	71.16	17.02	57.17
Czes2-Malt	dice	10.50	70.23	18.27	57.62
Czes2-MST	frq	10.51	70.54	18.29	57.83

Table 4.5: Results of the evaluation of the collocation extraction task for Czech comparing different tools, all which were extracting from the same corpus (Czes2). The first row stands for using the standard Czech sketch grammar, further rows use the Synt, SET, MST and Malt parser, respectively. We varied different parameters for filtering (taking top n items, filtering by minimum frequency) and sorting (by frequency or by the association score) of the collocations, and for each tool we are giving the best forming combination here. Since the overall focus of the task was more on recall than precision, the main evalution criterion was the f_5 score instead of f_1. Each row shows first the tool used and the sorting of the candidates used (frq stands for frequency, dice for the logDice association measure), followed by precision, recall and the f-scores. We used minimum frequency of 3 hits for filtering the collocation candidates and taking maximum of 1,000 collocates into the evaluation for each headword. Details of the evaluation are discussed in [Kilgarriff et al., 2014c].

annotators will not add any new. But that was not achieved and in terms of that, this was a negative result.

I believe the reasons for that were two-fold: first, assessing collocability is a task with notoriously low inter-annotator agreement, second, annotators by nature expect some distribution of the annotation data, so the conclusion is that whatever further data we gave them, they ticked a few words as collocations. When doing post-annotation consulting with the annotators, we also realized that part of the disagreement among them was simply caused by the fact that they did not know the meaning of some of the collocations that were given to them.[17]

In this evaluation, we also compared word sketches built using sketch grammars with those built using various parsers for Czech. An important outcome of this comparison was that while the parsers were on par with sketch grammar, they did not outperform them – in contrary to common assumption that parsers would be ideal tools for tasks like collocation extraction.

We repeated this kind of evaluation a number of items (with improvements to the parsers as well as to the sketch grammars), but the parsers never outperformed the sketch grammar. Latest results for the Czech collocation dataset are provided in Table 4.5.

17. In particular, we realized that we cannot presume that a 22-years-old girl will know what an *inline-four engine* is or that what makes *Thai box* different from, say, *South-Italian box*.

On one hand the fact that the parsers did not outperform the sketch grammar was a strong counterargument against their future use for this task (especially given the overall complexity and slow speed of parsing compared to the sketch grammar). On the other hand, the task may be – surprisingly – seen as one of the future challenges of parsing.

4.10.3 2016: precision-only

In 2016 I revisited the evaluation and decided that from practical point of view it is better to sacrifice the evaluation of recall but create a solid evaluation dataset that could be used for sketch grammar development, for regression testing of the parsing system as well as for comparing various corpus annotation tools like part-of-speech taggers, lemmatizers or tokenizers that preprocess the corpus before the word sketches are compiled. In other words, to conduct similar kind of evaluation like the one in 2010, but instead of merging *maybe* with *no* rather merging it with *yes*.

For the above given purposes, there was simply much stronger need to be able to identify clear mistakes resulting from incorrect part-of-speech tagging, errors in the sketch grammar queries, potential errors in the parser etc. I have devised a new annotation manual giving concise and specific instructions to the annotators as for what counts (from our perspective) as a good collocate, whose schema is given in Figure 4.17. It classifies the collocations into four classes: strong collocates, weak collocates, correct word combinations and errors.

The underlying idea is that any of the first three categories are in principle correct to be shown for a headword (with strong collocates being the most important one), but the last one is not and for sketch grammar development it is important that it is minimized. Early evaluation conducted internally within a team of five annotators on a small sample of 250 collocations with a given adjective headword, 500 collocations with a given verb headword and 500 collocations with a given noun headword show promisingly high inter-annotator agreement (of all five annotators) on error vs. non-error categories of 87.6 %, 82.6 % and 90.4 % for the adjective, verb and noun collocation group respectively.

In the future I aim at further improving the annotation guidelines and building new datasets for collocation evaluation on top of them.

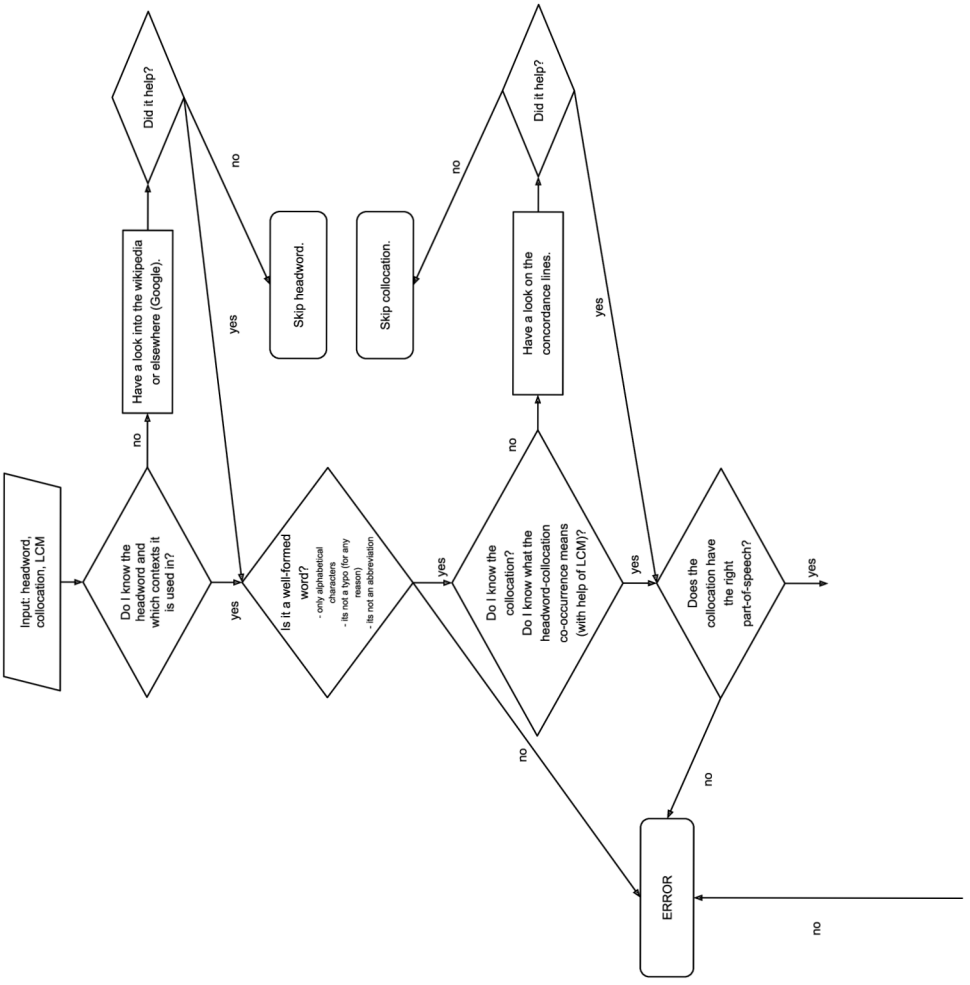
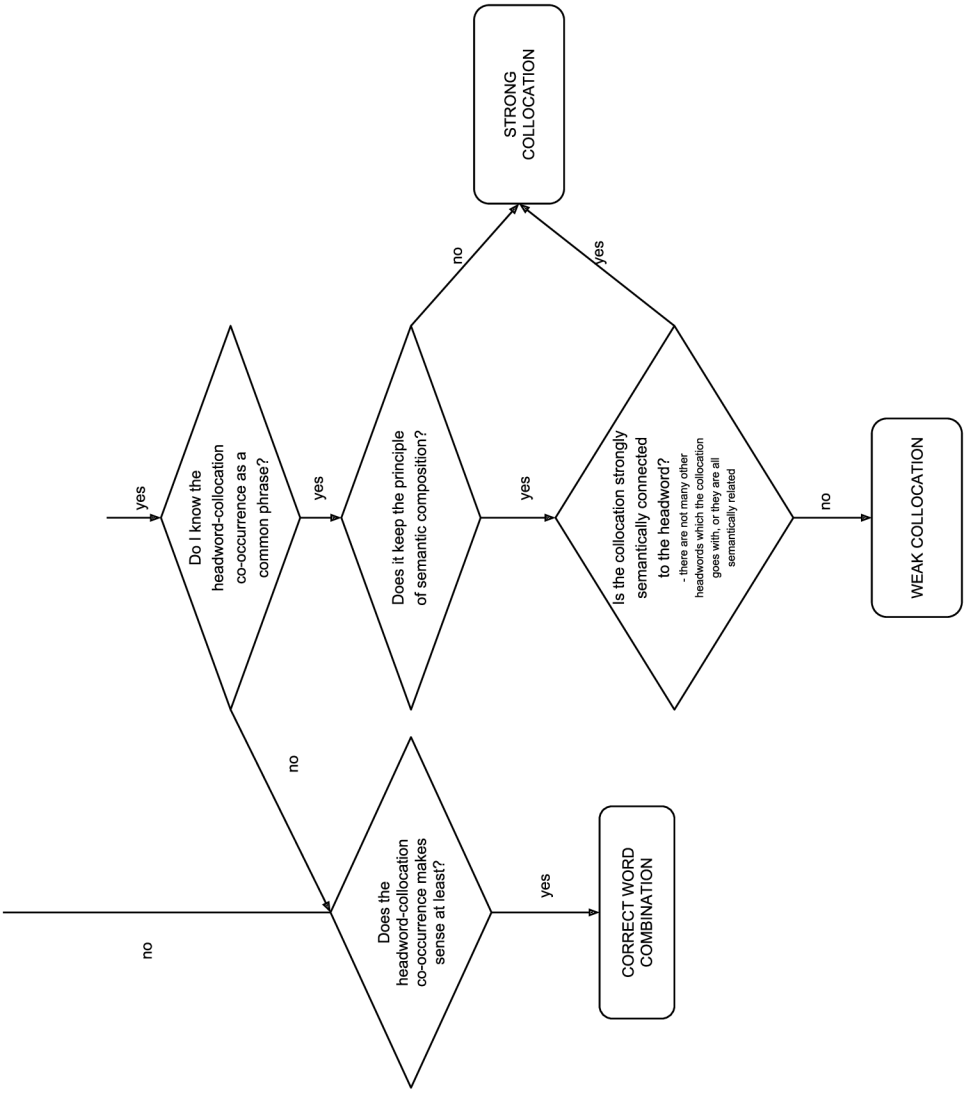


Figure 4.17: Current annotation key for evaluating automatic extraction of collocations (continues on next page).


Do I know the headword-collocation co-occurrence as a common phrase?
yes
Does it keep the principle of semantic composition?
yes
Is the collocation strongly semantically connected to the headword?
- there are not many other headwords which the collocation goes with, or they are all semantically related
yes
STRONG COLLOCATION
no
WEAK COLLOCATION
no
Does the headword-collocation co-occurrence makes sense at least?
yes
CORRECT WORD COMBINATION
no
no

Chapter 5

Parsing small: Synt

5.1 Overview

In this chapter I focus on *parsing small* by means of parsing individual sentences. I focus only on full parsing where the task is to automatically derive a syntactic representation of the whole sentence.

I present an approach that has a rule-based parser in its heart (Synt) and show how even the ambiguous output of the parser can be used for practical tasks such as morphological disambiguation. This is discussed in closing of the chapter together with detailed evaluation which shows that the parser can improve state-of-the-art automatic tagging of Czech tags when deployed in front of a statistical tagging system.

5.2 The Synt parser

Synt is a rule-based parsing system with probabilistic context-free backbone which was initially developed by [Kadlec, 2007] and which I worked on since 2008.

Synt performs a stochastic agenda-based head-driven chart analysis using the provided grammar for Czech, employs several disambiguating techniques and offers both a phrase-structure as well as dependency and chunk output.

An overview of the parsing workflow of Synt is given in Figure 5.1. First, the input sentence is morphologically analyzed (using the Majka morphological analyzer) if the annotation is not already provided. Synt is rather unique among parsing systems in that it can deal with ambiguous morphology on input which I exploited for partial morphological disambiguation, as is explained in further text.

As the next step, a basic head-driven CFG analysis is performed. All parsing results are collected in the so called *chart* structure which is built up during the analysis. Upon the basic CFG analysis, we build a new graph structure called *forest of values*. This corresponds to what is usually called

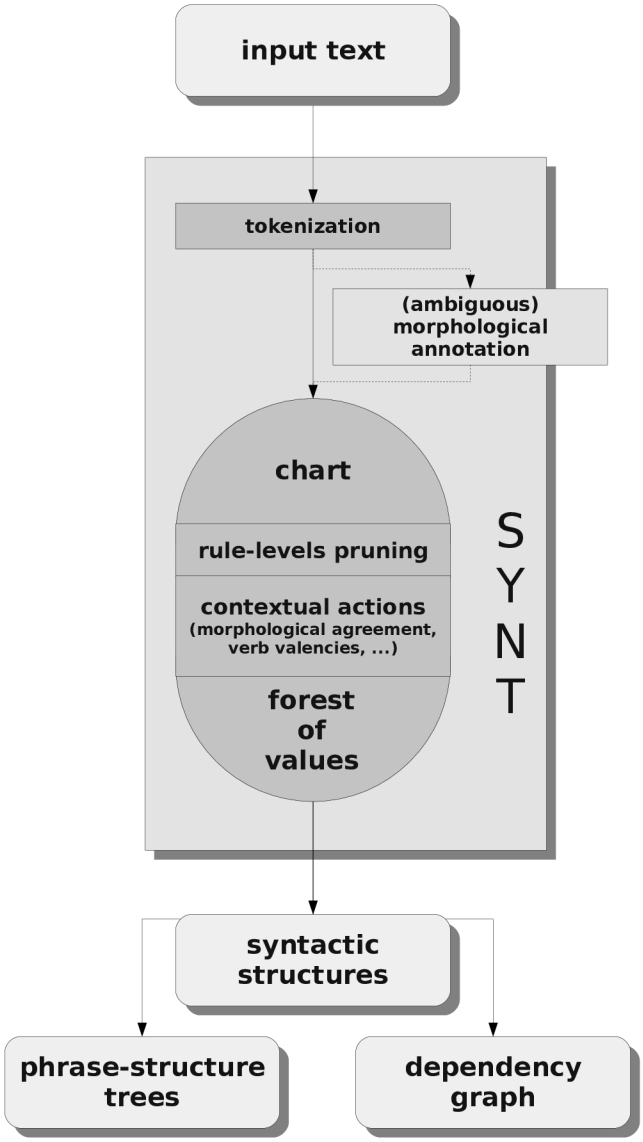


Figure 5.1: Parsing workflow of the Synt parser

a *parse forest* except that it breaks down individual parse trees according to the morphological values that the trees are holding. Thus a forest of values is a refinement of a parse forest where two trees can have identical structure but differ in the morphological values kept in the tree nodes.

The forest of values originates from the application of contextual actions (especially morphologically agreements and tests) that are defined for individual rules in the parser's grammar. These contextual actions are evaluated on the resulting chart.

To address the very serious problem of grammar maintenance and development, the grammar is edited in the form of a metagrammar that contains only about 250 rules. From this metagrammar, initially developed by [Horák and Kadlec, 2005], a full grammar is automatically generated by applying simple transformations that are associated with each rule and mostly regard the fact that Czech is a free-word-order language (a sample transformation creates a permutation of all right-hand side non-terminals in a rule or generates clause rules from a clause pattern with corresponding derivation type). A full grammar contains almost 4,000 rules.

5.3 Grammar and Metagrammar

The concept of a metagrammar facilitating a simple but expressive metalanguage for rule production is absolutely crucial for a rule-based parsing system. For a rule-based system to be competitive with statistical methods, the number of rules necessarily must be very large. Without a robust mechanism for the maintenance of the grammar, its development becomes soon unmanageable.

It is important to emphasize that the sole purpose of a metagrammar is to ease the grammar development, enhance readability and comprehensiveness. The metagrammar formalism does not need to adhere to any syntactic theory, nor be linguistically plausible in any way. It is a purely technical aspect of the system development.

The core of the metagrammar lies in the description of the verbal structure of Czech clauses. A clause is described by rules where predicative parts are interleaved with arbitrary intersegments (both arguments and adjuncts) which may be positionally constrained. These intersegments are labeled as the `intr` non-terminal and are heavily used in the metagrammar generation concept in that they may be automatically inserted during the expansion of the full grammar from the metagrammar depending on the rule production type.

The metagrammar in Synt contains different rule production types denoted by different arrows in the rule:

-> standard rule production

--> expansion of `intr` non-terminal between all right-hand non-terminals of the rules, but not before the first and after the last

---> expansion of `intr` non-terminal between all right-hand non-terminals of the rules, and also before the first and after the last

----> like **--->** but also adds non-terminals for conjunction to the beginning of the right-hand side

Each of the above four variants can be also altered by using an equal sing instead of dash (e.g. **==>**), in which case the rule generation system also checks for correct positioning of non-terminals of clitics on the right-hand side of the rule. Clitics represent one of rare Czech syntactic phaenomena that contravene the free-word-order principle because they have a fixed position being always the second constituent in a clause.

In addition, the metagrammar allows following three functions to be applied at the right-hand side of any rule:

order generates all possible permutations of right-hand side non-terminals (while pertaining to the constraints imposed by the arrow type)

first constraints the given non-terminal argument to be placed as the first constituent on the right-hand side

rhs replaces the given non-terminal argument with all its right-hand side derivation. This basically allows using "silent" rules, whose left-hand side are not present in the final grammar.

During my work on Synt I have significantly extended the metagrammar towards two directions:

- decreasing its overgeneration by applying the grammar stratification principle explained in Subsection 5.9.2,
- increasing its coverage by expanding the grammar core describing verbal structure of Czech clauses.

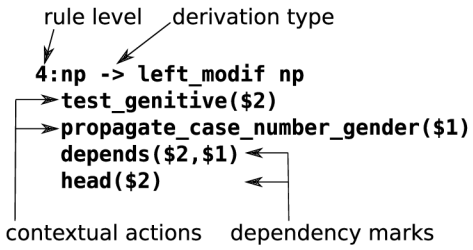


Figure 5.2: A sample grammar rule illustrating key rule features.

In the latter point, I devoted most grammar development work to Czech infinitive constructions which are frequent and exhibit rather complex syntactic behaviour. As result, the current grammar achieves coverage of 84 % on the Czech morphologically disambiguated corpus Desam (which has ca 1 million words). It must be noted though that the parser produces results in all (even those extra-grammatical) cases by falling back to strategy where partial parses of the sentence obtained so far are combined together. This strategy, called maximum optimal coverage, was initially developed by [Ailomaa et al., 2005].

5.4 Phrase structure grammar with dependency marking

The metagrammar rules in Synt may also contain information about dependency relations among constituents used at the right-hand side of the rule production (see Figure 5.2). These are used to build an ambiguous dependency graph, alongside with the tree forest, in the process of the analysis.

While the primary output of the parser is phrasal, the dependency graph can be used where ambiguous dependency annotation is sufficient, and was e.g. used for the experiments with building word sketches from the parser output as described in Section 4.6.2 of the previous chapter.

Further on, obtaining dependency trees from the ambiguous dependency graph is possible through algorithms for finding mimimum spanning trees in the graph.[1]

5.5 Chart and Forest of Values

The analysis done by Synt proceeds in two main steps: first, a probabilistic CFG analysis is performed, using a head-corner chart parser, a variant of the general chart parsing as described by [Sikkel, 1997] where the matching

1. As was demonstrated in a bachelor thesis I was supervising, see [Duráni, 2012].

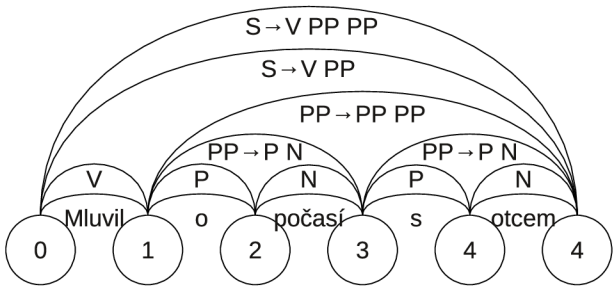


Figure 5.3: A sample chart structure representing the result of CFG parsing of the sentence: *Mluvil/(he) talked/V o/about/P počasí/weather/N s/with/P otcem/father/N*.

of each rule starts at a designated non-terminal (a head). At this stage, the internal chart structure (a directed acyclic multigraph) is built and serves as the core result for next steps.

To capture contextual phenomena such as grammatical agreement, the resulting chart is subject to multiple contextual actions that mostly handle morphological agreement and other lexical specifics. This allows to separate a clean and simple CFG analysis from more complex techniques that, while mostly expressible within a CFG formalism, would significantly complicate the grammar. In Synt these actions are in-programmed and as such theoretically Turing-complete. However their practical complexity is usually quite low as they mostly focus on local context.

During the evaluation of these actions, a new data structure is built and represents main result of the whole analysis – a so called *forest of values*, a packed representation of all syntactic trees where each node (*a value*) contains related morphological information. The relation between both of the structures is demonstrated in Figures 5.3 and 5.4.

From this forest all other outputs are retrieved, including n best syntactic trees or unambiguous syntactic structures [Jakubíček et al., 2009]. A forest of values can be viewed as a graph of value nodes where each node holds multiple lists of children, with each list representing one possible analysis. The packing is achieved by sharing same children (list items) among all the lists. To get all possible derivation trees, one would recursively traverse the graph and pick a single list with children in each node.

5.6 Handling rich-morphology

Unlike most parsing systems, Synt can handle morphologically ambiguous input, i.e. a sentence with words associated with their analysis (usually

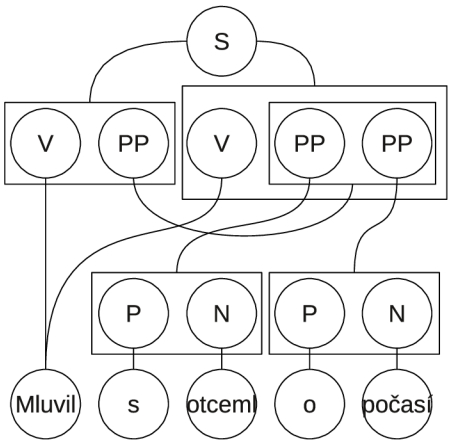


Figure 5.4: A sample forest of values built from the chart given in Figure 5.3.

according to a morphologically database), but without disambiguation performed by a part-of-speech tagger. For Czech and Slovak the parser contains bindings to the Majka morphological analyzer so that it can handle plain text input and pass it internally to Majka for analysis.

Obviously syntactic analysis on input text which is not morphologically disambiguated is much more ambiguous. Due to the free-word order of Czech, it is not uncommon to find sentences that (with ambiguous morphology) have billions of possible derivation trees after the primary CFG analysis, and millions after evaluating all contextual actions. Thanks to the packed representation of chart and forest of values, the system can usually handle these very efficiently.

Moreover, the parser can help for partial disambiguation of the morphological annotation of its input as I describe in detail in Section 5.10.2.

5.7 Implementation

The parser is implemented in a mixed C/C++ code, with additional helper scripts in Python and Bash. During my work on the Synt system, I have significantly refactored the old codebase to make it smaller and more compact but mainly to speed up the parser and make it easier to port to further languages (as explained in the next section).

The parser as such is a single binary that is built from multiple C/C++ modules. These modules are categorized according to their purpose into separate directories corresponding to the modules function:

1. `morphlex`

modules for handling input text and morphological annotation

2. `chart`
   modules for performing chart-based PCFG analysis
3. `actions`
   modules for executing contextual actions on top of the chart
4. `deps`
   modules for handling dependency graph analysis
5. `data`
   data directory for grammars and other language resources
6. `scripts`
   helper scripts for manipulating language resources
7. `doc`
   documentation
8. `utils`
   utility modules used across the system

In terms of speed, the main engineering strategy that I exploited to improve the parser's speed is careful handling of memory allocation. Graph-based algorithms like the ones used by Synt are extremely susceptible for high memory fragmentation and performing lot of memory allocations and deallocations which always involve context-switching by the operating system and therefore are very costly.

Thus I have implemented an own memory allocation pool which is used by the most memory intensive data structures in Synt. The underlying principle is simple: instead of allocating and deallocating memory for every sentence, data structures use a preallocated pool. This pool only deallocates memory when it uses more than double of the average memory needed so far. Because no reallocation is needed, the implementation can be rather simple: a 150-lines C module which redefines the standard `malloc` and `free` syscalls to macros pointing to its own implementation.

A current speed evaluation conducted on parsing of the Czech corpus Desam comprising 49,163 sentences shows that Synt parses about 24 sentences per second.[2] Synt, like any sentence-by-sentence processing tools, can be trivially parallelized by splitting the data into n jobs and running them concurrently.

2. Using a single core of an Intel Xeon CPU X5355 running at 2.66 GHz.

5.8 Scalability: Multilinguality

One of the key aspects I was focusing during the development of Synt was it's multilinguality. Clearly, one of the biggest advantages of statistical approaches (not only) to parsing was easy portability to further languages of the tools, whereas rule-based systems need special care.

When I started working on Synt, it was hardly portable at all and many of its features were hardcoded into the tool. These were eliminated one by one, so that at the moment there are three critical parts that need to be amended when porting Synt to a new language:

1. **metagrammar**
   This is the obvious. The metagrammar exists in the form of a single plain text file. An open research question is how much of the grammar is transferrable across languages. Early experiments show that lots of it is transferrable across language families, i.e. that the grammar can easily be ported to Slavic languages (see below the case for Slovak).

2. **mapping from morphology to syntax**
   Individual words need to be assigned so called pre-terminals, i.e. non-terminals which are immediately dominating them. This mapping used to be in-programmed, but is now externalized into a separate lexer where it is defined in a plain text file in which each line denotes the desired non-terminal mapping from a word-lemma-tag input triple. For more details see [Jakubíček and Medved', 2013].

3. **contextual actions**
   These represent the hard core of the Synt business. At the moment they are still in-programmed for efficiency reasons, however many of the current actions can be reused for further languages: they cover grammatical agreement that is not specific for Czech, but parametrized for particular grammatical categories such as case, gender or number. Nevertheless, this is by all means the biggest obstacle for porting to a new language that still needs to be worked on.

All these enhancements to Synt were exemplified on the case of Slovak: the closest language to Czech, which has some (irregular) lexical differences, some (rather regular) morphological differences and almost identical syntax. This was all in collaboration with Marek Medved', a native speaker of Slovak

who I supervised for his bachelor and master thesis and who was keen on implementing many of the ideas that I had to ease the three porting steps mentioned above.

As result, Synt has now got full support for Slovak and is one of the very few parsing systems available for Slovak, which, unlike Czech, has substantially less language resources and tools. Details of porting Synt to Slovak are described in [Medved' et al., 2012].

5.9 Facing ambiguity in parsing

Two main strategies are used to mitigate the ubiquitous parsing ambiguity, one probabilistic and one rule-based. The probabilistic one lies in using a probabilistic CFG grammar, with probabilities learnt from a small phrasal treebank used for the parser development; the rule-based consisting of assigning *levels* to individual rules as a precedence mechanism.

5.9.1 Rule probabilities: Learning Probabilities From Treebanks

Having each rule assigned a weight (or probability) allows obtaining n-best trees from the resulting parse forest.

The obvious issue with learning probabilities from a treebank for rule-based system is that when the grammar changes, the underlying treebank should be amended so as to account for the grammar changes and hence provide the necessary probabilities for new or changed rules. For this task, the treebank exists in the form of *tree limits*: phrase structure constraints that limit the output of the parser to a single tree. On grammar change, the treebank compliant with the new grammar can be created by reprocessing treebank sentences using these limits and – if needed – amending the limits so that they constrain the analysis sufficiently to obtain a single tree at its end.

The other option for obtaining probabilities is to used an inside-outside-like algorithm to obtain probability estimates from a large unannotated corpus. A very good overview of this method can be found in [Eisner, 2016].

5.9.2 Rule levels: Grammar Stratification Exploiting Competing PCFG Rules

Each grammar rule may be also given a rule level indicated by an integer – the higher number is assigned to the rule, the lower is the rule priority.

Currently, the whole grammar is structured into 7 levels, 0 being the default and most prioritized level.

5.9.2.1 Chart-based Local Pruning

First implementation of the metagrammar design and parsing algorithms in Synt used a naive algorithm for pruning of the chart based on rule levels at the local scope of a single chart edge: if a chart edge had multiple parent edges with different rule levels, only the parents with lowest level have been preserved, all others have been removed. This was however, only sufficient when the difference in rule levels was present within the locality of the given edge, as is illustrated in Figure 5.5. Non-local differences were not propagated upwards in the chart and hence situations like the one demonstrated in Figure 5.6 remained uncovered by this approach.

Another important disadvantage of this method was that it actually harmed the coverage of the parser to some extent due to the fact that the pruning was performed on the chart before computing contextual actions and building the forest of values – in rare (but possible) cases it happened that a pruned analysis would have passed the contextual actions while all others that were preserved in the chart after the pruning have failed in the contextual actions.

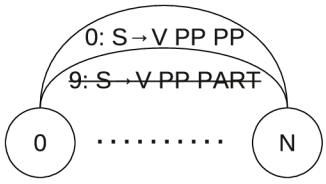


Figure 5.5: A positive example of local pruning on the chart (rules are prefixed by their levels). Here the S → V PP PART edge would get removed.

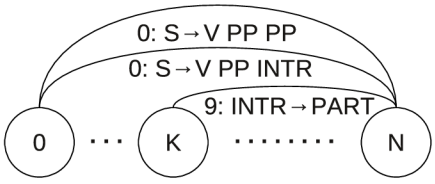


Figure 5.6: A negative example of local pruning on the chart (rules are prefixed by their levels). No pruning would be performed because the rule level does not distinguish in the local scope.

5.9.2.2 Forest-based Non-local Pruning

The new algorithm [Jakubíček, 2011] that I designed and implemented addresses both of the issues mentioned above – it performs non-local pruning so that it effectively filters all but best-leveled analyses and it operates on the forest of values instead of the chart so that it cannot cause the parsing process to fail even though some analysis was possible.

Moreover, the pruning process happens on-the-fly when the forest of values is being built up from the chart. Henceforth the algorithm does not slow down the analysis, but on contrary speeds it up because the resulting forest of values is significantly smaller.

Algorithm 1 Forest-based non-local pruning algorithm pseudocode.

Require: V – value node, CL – list of children to be considered
  $newChildLevel = 0$;
  for all $C \in CL$ **do**
    $L = \text{MAX}(C_e, C_c)$
    if $L \geq newChildLevel$ **then**
      $newChildLevel = L$
    end if
  end for
  if $V_c \geq newChildLevel$ **then**
    removeAllChildrenLists(V)
    addChildrenList(V, CL)
    $V_c = newChildLevel$
  else if $V_c \leq newChildLevel$ **then**
    return
  else
    addChildrenList(V, CL)
    $V_c = newChildLevel$
  end if

The whole pruning algorithm is described in pseudo-code below and illustrated in Figure 5.7 and proceeds as follows: each value node N in the forest of values contains the rule level of the corresponding edge in the chart N_e and maximum rule level of all its children N_c. Before adding a new possible analysis (i. e. another list of children) it retrieves the maximum C_{max} of the rule levels of all the new children, where a level of child M is defined as $max(M_e, M_c)$. If this new level C_{max} is greater than the current children level N_c, this analysis is discarded. If C_{max} is smaller than N_c, all of

the current analyses are substituted by this new one and C_{max} is updated to N_c. Otherwise (when $C_{max} = N_c$) this analysis is just added to the current list of children.

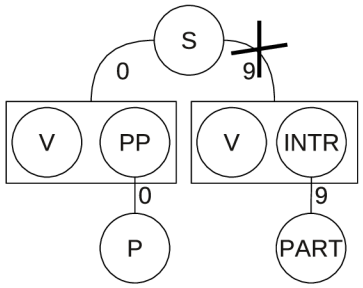


Figure 5.7: Forest-based non-local pruning: rule levels get propagated (see Figure 5.8) and prune the structure.

5.10 Evaluation

I present two types of evaluation of the parser's performance. The first is based on a small treebank of phrasal trees, referred to as Brno Phrasal Treebank, that was originally developed in 2008 [Kovář and Jakubíček, 2008] and that I later expanded and updated alongside the grammar development in Synt. The second is based on a particular task of using the parser for improving the accuracy of part-of-speech tagging of Czech. While the former is used for parser development to evaluate the contribution of various techniques to the parser's performance, the latter combines the parser with state-of-the-art taggers for Czech and shows that the parser can improve the tagging accuracy of purely statistical taggers and that such a combination

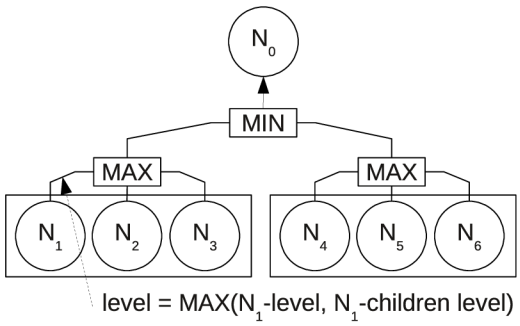


Figure 5.8: Forest-based non-local pruning: after the propagation, each value node contains only lists of children with minimal rule level.

represents best state-of-the-art tagging for Czech.

value	before	after
# of sentences	5,599	
# of not accepted sent.	202	116
median trees count	**80**	**16**
average trees count	**2,175,562**	**41,388**
LAA Best	0.8973	0.9016
position of LAA Best	26	17
LAA First	0.8578	0.8649
Time elapsed	0:28:23	0:21:54
Time per sentence	0.28 s	0.21 s

Table 5.1: Evaluation of the forest-based non-local pruning on the Brno Phrasal Treebank. LAA (Leaf-Ancestor Assessment) best value refers to the best achieved LAA among the first 100 trees, LAA First is the LAA of the first tree. Note that the indicated times entail analysing each sentence twice, computing LAA and all other statistics. Raw time of analysing each sentence once is ca. 25 % of this one.

5.10.1 Brno Phrasal Treebank

The Brno Phrasal Treebank [Kovář and Jakubíček, 2008] consists of 5,599 sentences, out of which 3,779 have unambiguous and 1,820 have ambiguous morphological annotation (plain text annotated by the Majka morphological analyzer). Each sentence is accompanied with a set of limits (described in Section 5.9.1) and a constituent tree. In Table 5.1 an overall comparison is given for the whole set of sentences showing the difference in parser performance before and after implementing the forest-based pruning method. A significant drop-off in the ambiguity can be seen as the average number of syntactic trees has been reduced by two orders of magnitude and the median dropped to quarter of its former value – all of this without harming parsers accuracy that has been actually even slightly improved. A noticeable speedup of the analysis has been achieved as well, conforming to the reduced size of the parse forest.

In Table 5.2 the results are provided separately for sentences with ambiguous and unambiguous morphological annotation. We can see that in the unambiguous case, the newly implemented methods lead to pruning up to dozens of trees. This is extremely important since it potentially enables to perform an exhaustive search of the remaining trees and to use deep and

	unambiguous		ambiguous	
value	**before**	**after**	**before**	**after**
# of sentences	3,779		1,820	
# of not accepted sent.	114	73	88	43
median trees count	**24**	**6**	**6144**	**480**
average trees count	**12,601.5**	**144.6**	**6,768,409.0**	**127,525.9**
LAA Best	0.9128	0.9131	0.8643	0.8776
position of LAA Best	19	8	43	36
LAA First	0.8784	0.8833	0.8141	0.8263

Table 5.2: Evaluation of the forest-based non-local pruning separated for sentences with ambiguous and unambiguous morphological annotation.

possibly time-consuming methods to rerank such a small set of trees.

The forest-based pruning method effectively reduces the number of resulting syntactic trees to the extent where we can afford using tree-based reranking techniques to continue improving the precision of the parser. It also allows further development of the metagrammar used by Synt in the form of competing CFG rules and extend the coverage of the parser without harming its accuracy by increasing the overgeneration of the grammar.

The measured results also confirm the fact that the ambiguity of the morphological annotation of the input still plays a key role in the ambiguity of the parsing itself – while the above mentioned results are satisfiable for unambiguous input, the ambiguous case still remains to be an issue.

5.10.2 Part-of-speech tagging

Thanks to the fact that the parser is capable of handling morphologically ambiguous input, it can be exploited for partial morphological disambiguation. In further text I explain the mechanism of such a partial disambiguation and present the evaluation of the contribution that such a partial disambiguation brings to the accuracy of part-of-speech tagging.

5.10.2.1 Partial morphological disambiguation

As explained in Section 5.5, the analysis conducted by Synt proceeds in two steps: after the initial CFG tagging, the chart of all parses is subject to contextual actions. These actions in most cases focus on checking mor-

phological agreement among constituents in the candidate parse trees. The contextual actions are evaluated recursively in a bottom-up manner where results are propagated up to the root of each parse tree. The additional step (not needed normally for parsing) is top-down propagation of the results through the parse forest back to individual surface words, in terms of pruning morphological ambiguities.

In [Jakubíček et al., 2009] I have conducted initial experiments with this approach when I evaluated this strategy on a morphologically disambiguated corpus (DESAM [Pala et al., 1997]) by reprocessing the text with the (ambiguous) analyzer and checking in how many cases the parser would predict removal of a correct tag. The results of that evaluation are given in Table 5.3 and since they were quite promising, as the off-the-shelf approach yielded an error rate of less than 1.5 %,[3] I have been continuously working on improvements to the parser and grammar to improve the results. The overall strategy still remains the same: it targets precision, not recall. The parser removes a morphological tag if and only if such a tag is not allowed under any syntactic interpretation of the sentence that is present in the parse forest. Clearly, this technique as such disambiguates only a little – but as is shown in the evaluation, it disambiguates cases where a statistical tagger otherwise may make a mistake such as long distance dependencies or subject-verb agreement.

In Figure 5.9 I give a simplified example of the process of partial morphological disambiguation on a single tree. In practice this procedure runs on the parse forest by traversing the directed acyclic graph in topological order.

value	**before**	**after**
average unambiguous words	20.73 %	46.12 %
average pruned word tags	38.37 %	
error rate [4]	< 1.46 %	
number of sentences	29,604	

Table 5.3: Partial morphological disambiguation evaluated on the DESAM corpus. Averages are computed per sentence: after the partial disambiguation a sentence had ~46 % of fully disambiguated words on average.

3. And in fact, the parser discovered many errors in the human annotation of the corpus at that time.

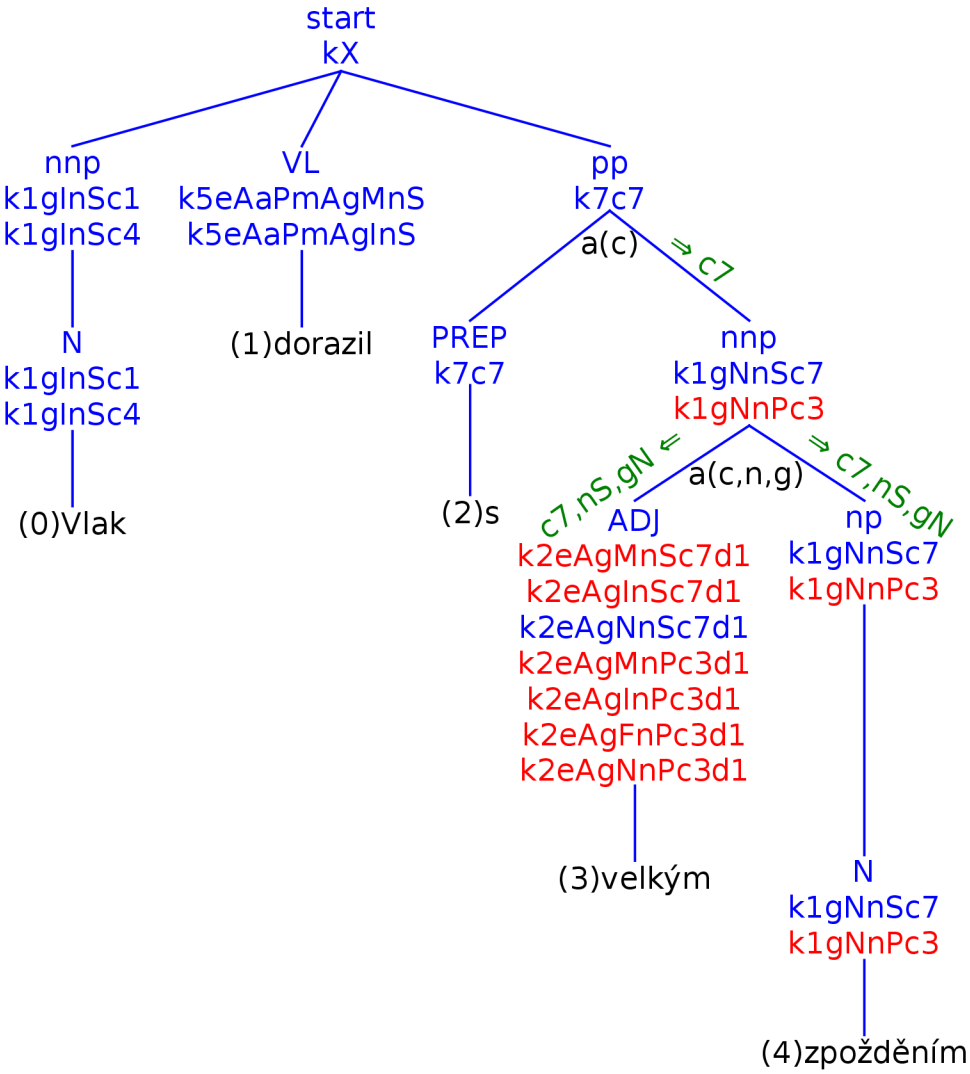


Figure 5.9: . Partial morphological disambiguation illustrated on a sample tree of the input sentence *Vlak/Train/N dorazil/arrived/V s/with/PREP velkým/big/ADJ zpožděním/delay/N*. Each contextual action prescribes grammatical categories that are propagated down the tree. In this example the agreement in case – `a(c)` – and case, number and gender – `a(c,n,g)` – on the level of a noun and prepositional phrases is in action.

5.10.2.2 Comparison with part-of-speech taggers

The mechanism of partial morphological disambiguation has been evaluated against or in combination with various part-of-speech taggers for Czech. These tools come from divergent sources and use divergent annotation tagsets, which makes them hard to compare. When developing a single tool, the standard evaluation scheme that one usually uses is some sort of cross-validation. However, as soon as different tools are to be compared, there is a large number of factors that are related to differences in design or annotation standard used by these tools, and performing a fair comparison is extremely hard, especially when state-of-the-art results on standard datasets are improved only by fraction of a percent (hence every single word counts).

This already starts with the very first, and by far simplest, step of sentence segmentation and, mainly, tokenization. While tokenization is a relatively easy task for most languages using Latin alphabet where words are space-separated, it can influence the evaluation a lot: a tagger trained on a corpus where all numbers followed by a unit (such as *3 cm*) are being handled as one token will not perform well if it is fed with an input text where these are tokenized as two tokens.[5] In every language there is plenty of such peculiarities in tokenization here and there: enough to disqualify a tool from the game.[6] Moreover, it is often the case that the expected tokenization is actually prescribed only by the tokenization of the training corpus and is hard to be replicated on new texts (and, especially, words or phrases not present in the training corpus).

Therefore, I have instead evaluated only the *differences* in the results of the tools that have been tested, and all of that after careful preprocessing for accounting of the major tagset differences. In the remainder of this section I first describe the taggers used in the comparison, the annotation setup of the evaluation task, and finally discuss the results in detail.

In the comparison I have used four statistical taggers, two based on Hidden Markov Models and two on neural networks. Two out of these four were designed to work also on morphologically analyzed (but not disambiguated) text and these were used in conjunction with Synt to evaluate its contribution through partial morphological disambiguation. In this case the input sentence was first parsed and partially disambiguated by Synt and then forwarded for final disambiguation by the tagger. One of the taggers

5. A detailed study on the effects of (also) tokenization on taggers comparison was conducted for Polish in [Radziszewski and Acedański, 2012].

6. Even in English there are different strategies for tokenizing words like *don't*, *cannot*, I'd etc.

was already used in combination with a syntactic chunker, either with or without a verb valency resolver, hence there were all together 8 processing chains that were compared:

1. Google SyntaxNet [Andor et al., 2016]

   Google has released a part-of-speech tagger and parser built on top of what they call TensorFlow, a framework for numerical computations. SyntaxNet uses a feed-forward neural network to perform the training and comes with pre-trained models for 40 languages[7]. I used the model for Czech, trained on the Prague Dependency Treebank, in the comparison.

2. MorphoDiTa [Straková et al., 2014]

   MorphoDiTa is an averaged perceptron based tagger developed at the Institute of Formal and Applied Linguistics at Charles University in Prague.[8] It also comes with a model for Czech trained on the Prague Dependency Treebank. The tagger makes it possible to use morphologically analyzed input, but this needs to be specified at the time of training. Hence, using this option would need retraining the model. As the authors do not make the training parameters publicly available, I better used their model as is in the comparison (but was not able to combine it with the preprocessing by Synt).

3. RFTagger [Schmid and Laws, 2008]

   RFTagger is a variant of the TreeTagger [Schmid, 1994] devised for morphologically-rich languages. It differs from TreeTagger in that it does not treat the tags as atomic units but decomposes them into the respective grammatical categories which are modelled and conditionalized separately. The tagger normally accepts only plain text input but thanks to the fact that it comes with the source code[9] I have developed a slightly modified version which accepts morphologically analyzed input and uses it to prune hypotheses of the trained model so that it was possible to preprocess the input by Synt. In the comparison I use a model that I have trained myself on the DESAM corpus alongside with the morphological lexicon obtained from the Majka database. I have compared both the standalone tagger and its variant with input preprocessed by Synt.

7. Available at https://github.com/tensorflow/models/blob/master/syntaxnet/g3doc/universal.md
8. Available at http://hdl.handle.net/11858/00-097C-0000-0023-43CD-0
9. Available at http://www.cis.uni-muenchen.de/~schmid/tools/RFTagger/

4. desamb [Šmerk, 2004]

   Desamb is a simple HMM-based tagger developed by my colleagues in the Natural Language Processing Centre at Masaryk University in Brno. It analyzes the input sentence with the Majka morphological analyzer first, and then performs disambiguation separately. Internally it may use two systems for syntactic chunking (Dis) and valency detection (Vadis) [Žáčková, 2002]. In the comparison I used four variants: the tagger with no extensions, the tagger with Dis enabled, the tagger with both Dis and Vadis enabled and finally the tagger with preprocessing by Synt.

In further text, the processing chains described above are referred to as:

- `syntaxnet`
- `morphodita`
- `rftagger`
- `rftagger+synt`
- `desamb`
- `desamb+dis`
- `desamb+disvadis`
- `desamb+synt`

For the annotation, I used a small but newly created corpus of Czech newspaper texts[10] consisting of 1,841 sentences, out of which a vast majority (1,715) contained at least one token where the tools produced different annotation and hence was subject to inspection by the annotators. However, the annotators only annotated the differing tokens which was just a fraction – 7,885 – out of the total number of 33,845 tokens. For every token with a difference, they were given the task to choose the right morphological tag out of the set provided by the compared tools or select that none of them is correct. The annotators did not know which tag is coming from which tool. They were using a web-based annotation interface that I developed specifically for this task where they were randomly given a set of sentences for annotation. A screenshot from the annotation interface is given in Figure 5.10.

10. Accessible alongside with other materials related to this thesis at http://nlp.fi.muni.cz/~xjakub/phd/

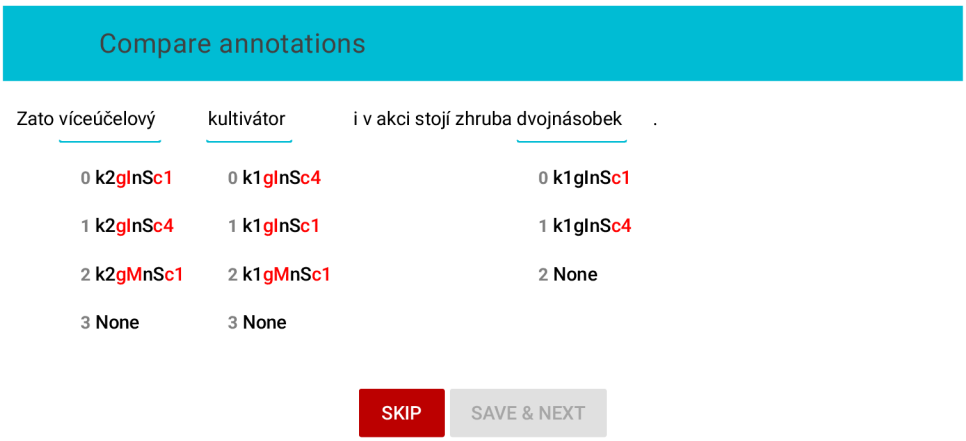


Figure 5.10: Annotation interface for comparing the tagging results of the compared toolchains. The interface can be fully operated by both mouse (selection by tapping or clicking) and keyboard (navigation by arrows and selection through pressing enter or using numeric keys) and comes with a mobile-friendly version.

The annotators also had the option to skip a sentence with three predefined justifications:

- when they were unable to decide about the correct tagging for any of the words in the given sentence
- when the decision about correct tagging required extra-sentential context
- any other reason

The main difficulty in the comparison indeed turned out to be the inherent differences in the annotation schemes used by the processing chains. In order to perform a fair evaluation which would focus on actual discriminative capabilities I have first manually inspected the most common differences and performed significant preprocessing of the annotation as follows:

- The most differences occurred in the annotation of function words: adverbs, prepositions, conjunctions, particles and interjections. There is indeed very low agreement among Czech linguists on these categories, which is also exhibited in the codification as given by standard Czech dictionaries. Especially ambiguities between adverbs, conjunctions and particles are extremely common.

  To give an example, one can take the word *ovšem* (which could be translated into English as *of course*, *naturally* or *though*). In the

two most recent Czech normative dictionaries (Dictionary of Written Czech and Dictionary of Literary Czech [Filipec et al., 1995, Havránek et al., 1971]), this word is classified as a particle in the former dictionary and as an adverb or conjunction in the latter dictionary. This is not an exception, but rather a pattern that can be observed very frequently. The classification of these words is tricky: they often behave in very unique ways and their handling needs to be lexicalized (i.e. defined per individual word) in subsequent processing such as parsing.

For the purposes of the evaluation, I have merged all these four part-of-speech categories into a single one. It would not make any sense to let the annotators choose from different categories without giving them clear guidance on how to disambiguate, whereas the only reasonable and consistent disambiguation I could think of would be to assign (almost) always a single part-of-speech. From the perspective of the comparison, this would not bring any added value.

- The only grammatical categories to be evaluated were the part-of-speech (grammatical kind), case, number and gender (for nouns, adjectives, pronouns and numerals) and person for verbs. Not all words falling into these five categories were always assigned all the attributes by the tools (e.g. some of the tools do not specify gender for some pronouns which do not inflect for gender), and in these cases I always evaluated only on those categories assigned by all the tools to discount for this fact.

- The annotators were asked not to evaluate foreign words and non-alphabetic symbols which represent another challenge in terms of standardisation and are handled in very different ways by the individual tools.

- In the same way I discounted the differences in the annotation of abbreviations. Some of the tools assign a specific category for abbreviations, others assign the category the word would have if not abbreviated. I asked the annotators to prefer the latter (more specific) annotation, but discounted for these differences in the final evaluation.

- Finally, the SyntaxNet and MorphoDiTa systems sometimes assign ambiguous gender categories (such as “feminine or neuter”). In

phaenomenon	#
proper nouns	303
numerals vs. adverbs[11]	15
nouns vs. prepositions[12]	13
foreign words	10
units (such as *km/h*)	9
abbreviations	6
adjectives vs. adverbs[13]	5
words varying between gender[14]	4
extra-sentential context	3
spelling errors[15]	3
nouns vs. adjectives[16]	3
adjectives vs. past participles of a verb[17]	2

Table 5.4: Analysis of remaining discrepancies. Counts are word counts from the 5.57 % of the annotation corpus with discrepancies after review.

most cases the ambiguity can be however resolved on the sentence level and it can be resolved almost always with addition extra-sentential context, therefore I considered these to be simply wrong. However, such cases were still subject to annotation so as to get a fair view on cases where other tools would predict a wrong gender anyway (e.g. a word in feminine gender which SyntaxNet or MorphoDiTa predict to be feminine or neuter, but other tools predicted it to be a masculine form).

To sum up, the overall focus was to get a fair comparison of the discriminative performance of the various tools where the clear cut cases are not lost in a cloud of noise coming from different paradigms, annotation schemes or traditions behind the individual tools.

Given all this preprocessing, the annotators were given only very light guidance since I wanted to evaluate whether relaxing the annotation task would make it sufficiently easy to rely on their linguistic intuition to disambiguate. Therefore, they were given just a one-page summary of annotation

toolchain	**accuracy**$_{\Delta}$	**accuracy**$_{sup}$
baseline	–	76.70 %
SyntaxNet	41.33 %	82.33 %
morphoDiTa	49.98 %	83.50 %
desamb+dis	60.86 %	84.98 %
desamb	62.77 %	85.24 %
desamb+disvadis	65.75 %	85.65 %
desamb+Synt	65.86 %	85.66 %
RFtagger	73.89 %	86.75 %
RFtagger+Synt	75.56 %	86.98 %

Table 5.5: Pre-review results of the comparison of tagging toolchains: accuracy$_{\Delta}$ stands for the percentage of correct tags among those annotated without a discrepancy (i.e. 82.67 % of those with a difference in tagging by any of the tools). Accuracy$_{sup}$ is a hypothetical upper bound accuracy assuming that the 76.70 % of words with no differences (denoted as baseline) in tagging by any of the tools were actually correct.

guidelines. All annotators were students of linguistic maior and every sentence was initially annotated by two annotators, after which I examined the results.

Initial inter-annotator agreement (IAA) was rather low – 82.67 %. A quick review immediately identified the most significant discrepancies that I consulted with the annotators. Most notably, some of them consistently marked the reflexive pronouns *se* and *si* as incorrectly tagged by all tools because they expected them to be tagged as verbs when part of a verb which always requires them (a verb traditionally called *reflexivum tantum*). Correcting these occurrences raised the IAA to 85.92 %.

Afterwards, a third review was conducted by me on all the remaining discrepancies and it revealed that most of them were merely mistakes done by one of the two annotators. Correcting these highered the agreement to 97.62 %. I have manually inspected all the remaining discrepancies and it turned out that the vast majority (over 80 %) is coming from (mostly foreign) proper nouns. These represent a major issue in tagging and native speakers take various approaches when naturalizing these words, especially what concerns their inflection. A detailed analysis of all the remaining discrepancies is provided in Table 5.4.

These discrepancies were not taken into account on further evaluation, hence all toolchains were compared only on the 97.62 % portion that I deem to be consistent after review. Nevertheless I reckon that the review conducted by me represents a methodological problem that is very hard to address generally, namely how to detect plain mistakes made by annotators and how to distinguish them from real disagreement that indicates issues in the annotation scheme and guidelines. Therefore I provide an evaluation on the raw dataset as obtained from the annotators (with IAA of 82.67 % and evaluating just on this portion) as well as on the post-review dataset.

Since the evaluations perfectly correlate in ordering of the toolchains, I am rather confident that the (random) mistakes corrected by me simply result from students trying to finish the task (that was part of their fulfillments for the term exams) as soon as possible. However, generally this is a deep methodological problem that needs to be accounted for on case-by-case basis.

The overall results of the comparison are provided in Tables 5.5 (pre-review) and 5.6 (post-review). It follows that Synt improved the tagging accuracy of both Desamb and RFTagger, and the combination with RFTagger was the best processing toolchain.

To my best knowledge, these are the first results showing that full parsing can improve (statistical) state-of-the-art taggers. Besides, the tagging task represents a useful way how the parser can be developed and evaluated. It also fosters a tight integration between the morphological and syntactic annotation which is not easy to maintain. Discrepancies in the interface between morphology and syntax can be seen manifested in results like [Vinyals et al., 2014] where the performance of a neural network based parser *improved* after removing part-of-speech tagging from the training data.

toolchain	**accuracy**$_{\Delta}$	**accuracy**$_{sup}$
baseline	–	76.70 %
SyntaxNet	42.22 %	83.45 %
morphoDiTa	50.49 %	84.77 %
desamb+dis	58.15 %	85.99 %
desamb	60.00 %	86.29 %
desamb+disvadis	63.10 %	86.78 %
desamb+Synt	63.21 %	86.80 %
RFtagger	70.37 %	87.95 %
RFtagger+Synt	72.30 %	88.25 %

Table 5.6: Post-review results of the comparison of tagging toolchains: accuracy$_{\Delta}$ stands for the percentage of correct tags among those annotated without a discrepancy (i.e. 97.62 % of those with a difference in tagging by any of the tools). Accuracy$_{sup}$ is a hypothetical upper bound accuracy assuming that the 76.70 % of words with no differences (denoted as baseline) in tagging by any of the tools were actually correct.

Chapter 6

Conclusions

In this thesis I discussed the topic of parsing of morphologically-rich languages.[1] In the first two chapters I have reviewed the state-of-the-art in syntactic theories and parsing systems. I argued that low agreement on a single syntactic representation together with predominantly intrinsic evaluation is a big argument against using purely statistical approaches in the form of machine learning systems that merely try to learn from the training data. I claimed that this is one of the main reasons why parsers are underused for many tasks for which they in theory could and should be used and why extrinsic evaluations exhibit very low correlation with intrinsic evaluations in parsing.

Further on, I presented two parsing solutions that I was co-designing and co-developing and I argued that each of them is suitable for different kinds of tasks and can be seen as mutually complementary. The first group of tasks targets gathering salient information from large amounts of texts which allow using simpler recognition procedures that focus on recall. The second group, which can be seen as a representative of traditional parsing, targets processing individual sentences aiming at maximizing the precision.

I explained how both of the presented approaches can handle language morphology that is rich and ambiguous. The second approach has been specifically designed for handling well high morphological ambiguity and is from that perspective rather unique among systems performing full parsing. Both of these approaches have a ruled-based backbone with additional statistical (probabilistic) measures. I showed that despite both of them primarily produce ambiguous syntactic representations, they can be successfully used for a wide range of multilingual tasks.

In case of word sketches, the collocation extraction task was used for numerous academic and commercial projects for many languages and despite the relative simplicity of the word sketch syntactic framework, word

1. Though many of the observations are applicable to languages with not-so-rich morphology as well.

sketches outperform full parsing systems.

In case of the Synt parser, I presented a state-of-the-art performance on the task of morphological tagging of Czech – which I believe is the very first case where full parsing improves a statistical tagging system.

All data and tools that are freely available are accessible from http://nlp.fi.muni.cz/~xjakub/phd/. This website contains electronic version of this thesis, links to the open source Synt parser and datasets used for the evaluation.

Appendices

Appendix A

List of author's publications

A.1 Journal papers

[1] 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.
[author's contribution: 80% – system design and implementation, evaluation and text writing].

[2] Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. The Sketch Engine: Ten Years On. *Lexicography*, 1(1):7–36, 2014.
[author's contribution: 20% – contribution to system design and development as well as text writing].

[3] Vojtěch Kovář, Vít Baisa, and Miloš Jakubíček. Sketch Engine for Bilingual Lexicography. *International Journal of Lexicography*, 29(3):339–352, 2016.
[author's contribution: 25% – contribution to system design and development as well as text writing].

A.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. Tribun EU s.r.o., Brno, 1. edition, 2009.
[author's contribution: 33% – contribution to corpus compilation and text writing].

[2] Adam Kilgarriff, Miloš Jakubíček, Jan Pomikálek, Tony Berber Sardinha, and Pete Whitelock. PtTenTen: A corpus for Portuguese lexicography. In *Working with Portuguese Corpora*, pages 280–287. Bloomsbury Publishing, London, 1 edition, 2014.

[author's contribution: 25% – contribution to corpus compilation and text writing].

[3] Miloš Jakubíček, Vít Baisa, Jan Bušta, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. *Walking the tightrope between linguistics and language engineering*. Springer-Verlag, 2017 (to appear).
[author's contribution: 90% – major part of design and text writing].

A.3 Edited proceedings

[1] Iztok Kosem, Miloš Jakubíček, Jelena Kallas, and Simon Krek. Electronic lexicography in the 21^{st} century: linking lexical data in the digital age. Proceedings of the eLex 2015 conference, 2015.
[author's contribution: 25% – co-editor].

[2] Iztok Kosem, Miloš Jakubíček, Jelena Kallas, Simon Krek, and Carole Tiberius. Electronic lexicography in the 21^{st} century: lexicography from scratch. Proceedings of the eLex 2017 conference, 2017 (to appear).
[author's contribution: 25% – co-editor].

A.4 Conference papers (peer-reviewed)

[1] 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: 33% – system implementation and text writing].

[2] Jan Bušta and Miloš Jakubíček. Building of Corpus Based E-learning Materials for Czech. In *SCO 2009 : Sharable Content Objects : 6. ročník konference o elektronické podpoře výuky*, pages 144–149, Brno, 2009. Masaryk University.
[author's contribution: 50% – system implementation and text writing].

[3] Miloš Jakubíček, Aleš Horák, and Vojtěch Kovář. Mining Phrases from Syntactic Analysis. In *Proceedings of the 12^{th} International Conference on Text, Speech and Dialogue, TSD 2009*, pages 124–130, Berlin, Heidelberg, 2009. Springer-Verlag.
[author's contribution: 75% – system design, implementation and text writing].

[4] Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. Syntactic analysis as pattern matching: The set parsing system. In *Proceedings of 4th Language & Technology Conference*, pages 100–104, Poznań (Poland), 2009. Wydawnictwo Poznańskie.
[author's contribution: 25% – system implementation and text writing].

[5] 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, Tokyo, 2010. Waseda University.
[author's contribution: 75% – system design, implementation and text writing].

[6] Miloš Jakubíček, Vojtěch Kovář, and Marek Grác. 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, Tokyo, 2010. Waseda University.
[author's contribution: 25% – system design and text writing].

[7] 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*, pages 161–171, Berlin/Heidelberg, 2011. Springer-Verlag.
[author's contribution: 25% – system implementation and text writing].

[8] Miloš Jakubíček. Effective Parsing Using Competing CFG Rules. In *Proceedings of the 14th International Conference on Text, Speech and Dialogue, TSD 2011*, pages 115–122, Berlin, Heidelberg, 2011. Springer-Verlag.

[9] Jan Pomikálek, Pavel Rychlý, and Miloš Jakubíček. Building a 70 billion word corpus of English from ClueWeb. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 502–506, Istanbul, Turkey, 2012. European Language Resources Association (ELRA).
[author's contribution: 33% – corpus compilation and text writing].

[10] Adam Kilgarriff, Jan Pomikálek, Miloš Jakubíček, and Pete Whitelock. Setting up for corpus lexicography. In Ruth Vatvedt Fjeld and

Julie Matilde Torjusen, editors, *Proceedings of the 15th EURALEX international Congress*, pages 606–612, Oslo, 2012. University of Oslo.
[author's contribution: 25% – corpus compilation and text writing].

[11] Miloš Jakubíček and Vojtěch Kovář. Enhancing Czech Parsing with Verb Valency Frames. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing 2013*, pages 282–293, Greece, 2013. Springer-Verlag.
[author's contribution: 75% – system design, implementation and text writing].

[12] Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlý, and Vít Suchomel. The TenTen corpus family. In *Proceedings of the 7th International Corpus Linguistics Conference, CL 2013*, pages 125–127, 2013.
[author's contribution: 20% – corpus compilation and text writing].

[13] Marek Medveď, Miloš Jakubíček, and Vojtěch Kovář. Towards taggers and parsers for Slovak. In Zygmunt Vetulani and Hans Uszkoreit, editors, *Human Language Technologies as a Challenge for Computer Science and Linguistics. Proceedings of the 6th Language and Technology Conference*, pages 527–530, Poznań, Poland, 2013. Fundacja Uniwersytetu im. A. Mickiewicza.
[author's contribution: 33% – system design and text writing].

[14] Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlý, and Vít Suchomel. Finding Terms in Corpora for Many Languages with the Sketch Engine. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 53–56, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.
[author's contribution: 20% – system design, implementation and text writing].

[15] Adam Kilgarriff, Pavel Rychlý, Miloš Jakubíček, Vojtěch Kovář, Vít Baisa, and Lucia Kocincová. Extrinsic Corpus Evaluation with a Collocation Dictionary Task. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language

Resources Association (ELRA).
[author's contribution: 20% – system evaluation and text writing].

[16] Miloš Jakubíček, Adam Kilgarriff, and Pavel Rychlý. Effective corpus virtualization. In *Proceedings of the Second Workshop on the Challenges in the Management of Large Corpora (CMLC-2)*, pages 7–9. European Language Resources Association – ELRA, 2014.
[author's contribution: 50% – system implementation and text writing].

[17] Lucia Kocincová, Miloš Jakubíček, Vojtěch Kovář, and Vít Baisa. Interactive visualizations of corpus data in sketch engine. In Gintaré Grigonyté, Simon Clematide, Andrius Utka, and Martin Volk, editors, *Proceedings of the Workshop on Innovative Corpus Query and Visualization Tools at NODALIDA 2015*, pages 17–22, Vilnius, Lithuania, 2015. Linköping University Electronic Press, Linköpings universitet.
[author's contribution: 25% – system design and text writing].

[18] Diana McCarthy, Adam Kilgarriff, Miloš Jakubíček, and Siva Reddy. Semantic word sketches. In *Proceedings of the 8th International Corpus Linguistics Conference, CL 2015*, pages 231–233, 2015.
[author's contribution: 10% – system implementation].

[19] Adam Kilgarriff, Vít Baisa, Miloš Jakubíček, and Pavel Rychlý. Longest-commonest match. In I. Kosem, M. Jakubíček, J. Kallas, and S. Krek, editors, *Electronic lexicography in the 21st century: linking lexical data in the digital age. Proceedings of the eLex 2015 conference*, pages 397–404, Ljubljana, 2015. Trojina, Institute for Applied Slovene Studies.
[author's contribution: 25% – system design and text writing].

[20] Vít Baisa, Vít Suchomel, Adam Kilgarriff, and Miloš Jakubíček. Sketch Engine for English Language Learning. In *Proceedings of the 8th International Corpus Linguistics Conference, CL 2015*, pages 33–36, 2015.
[author's contribution: 25% – system design and text writing].

[21] Adam Kilgarriff, Ondřej Herman, Jan Bušta, Vojtěch Kovář, and Miloš Jakubíček. DIACRAN: a framework for diachronic analysis. In *Proceedings of the 8th International Corpus Linguistics Conference, CL 2015*, pages 195–197, 2015.
[author's contribution: 25% – system evaluation and text writing].

[22] Vít Baisa, Adam Kilgarriff, Pavel Rychlý, and Miloš Jakubíček. Longest-commonest match. In *Proceedings of the 8th International Corpus*

Linguistics Conference, CL 2015, pages 36–39, 2015.
[author's contribution: 25% – system design and text writing].

[23] Vojtěch Kovář, Miloš Jakubíček, and Aleš Horák. On evaluation of natural language processing tasks: Is gold standard evaluation methodology a good solution? In Jaap van den Herik and Joaquim Filipe, editors, *Proceedings of the 8th International Conference on Agents and Artificial Intelligence*, pages 540–545, Rome, 2016. SCITEPRESS.
[author's contribution: 20% – text writing].

[24] Vít Baisa, Jan Michelfeit, Marek Medveď, and Miloš Jakubíček. European Union Language Resources in Sketch Engine. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2799–2803, Portorož, Slovenia, 2016. European Language Resources Association (ELRA).
[author's contribution: 25% – corpus compilation and text writing].

[25] Marek Medveď, Miloš Jakubíček, and Vojtěch Kovář. English-French Document Alignment Based on Keywords and Statistical Translation. In *Proceedings of the First Conference on Machine Translation*, pages 728–732, Berlin, Germany, August 2016. Association for Computational Linguistics.
[author's contribution: 25% – system design].

A.5 Conference papers (non-peer-reviewed)

[1] Vojtěch Kovář and Miloš Jakubíček. Test Suite for the Czech Parser Synt. In Petr Sojka and Aleš Horák, editors, *Proceedings of the Second Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2008*, pages 63–70, Brno, 2008. Masaryk University.
[author's contribution: 50% – system design, implementation and text writing].

[2] Miloš Jakubíček. Extraction of Syntactic Structures Based on the Czech Parser Synt. In Petr Sojka and Aleš Horák, editors, *Proceedings of the Second Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2008*, pages 56–62, Brno, 2008. Masaryk University.

[3] Miloš Jakubíček. Extrakce strukturních informací z běžného textu na základě syntaktického analyzátoru. In *VARIA XVIII*, pages 78–80, Prešov, 2009. Slovenská jazykovedná spoločnosť pri SAV.

[4] Miloš Jakubíček, Vojtěch Kovář, and Aleš Horák. Measuring Coverage of a Valency Lexicon using Full Syntactic Analysis. In Petr Sojka and Aleš Horák, editors, *Proceedings of the Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009*, pages 75–79, Brno, 2009. Masaryk University.
[author's contribution: 50% – system design, implementation and text writing].

[5] Vojtěch Kovář and Miloš Jakubíček. Prague Dependency Treebank Annotation Errors: A Preliminary Analysis. In Petr Sojka and Aleš Horák, editors, *Proceedings of the Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009*, pages 101–108, Brno, 2009. Masaryk University.
[author's contribution: 50% – evaluation and text writing].

[6] Miloš Jakubíček, Jan Bušta, Dana Hlaváčková, and Karel Pala. Classification of Errors in Text. In Petr Sojka and Aleš Horák, editors, *Proceedings of the Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009*, pages 109–119, Brno, 2009. Masaryk University.
[author's contribution: 25% – implementation and text writing].

[7] Vojtěch Kovář, Aleš Horák, and Miloš Jakubíček. How to Analyze Natural Language with Transparent Intensional Logic? In Petr Sojka and Aleš Horák, editors, *Proceedings of the Fourth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2010*, pages 69–76, Brno, 2010. Masaryk University.
[author's contribution: 10% – implementation].

[8] Miloš Jakubíček and Vojtěch Kovář. CzechParl: Corpus of Stenographic Protocols from Czech Parliament. In Petr Sojka and Aleš Horák, editors, *Proceedings of the Fourth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2010*, pages 41–46, Brno, 2010. Masaryk University.
[author's contribution: 80% – corpus compilation and text writing].

[9] Aleš Horák, Miloš Jakubíček, and Vojtěch Kovář. Analyzing Time-Related Clauses in Transparent Intensional Logic. In A. Horák and

P. Rychlý, editors, *Proceedings of the Fifth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2011*, pages 3–9, Brno, 2011. Tribun EU.
[author's contribution: 10% – corpus compilation and text writing].

[10] Miloš Jakubíček, Vojtěch Kovář, and Pavel Šmerk. Czech Morphological Tagset Revisited. In Aleš Horák and Pavel Rychlý, editors, *Proceedings of the Fifth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2011*, pages 29–42, Brno, 2011. Tribun EU.
[author's contribution: 33% – design of the methodology and text writing].

[11] Marek Medveď, Miloš Jakubíček, Vojtěch Kovář, and Václav Němčík. Adaptation of Czech Parsers for Slovak. In Aleš Horák and Pavel Rychlý, editors, *Proceedings of the Sixth Recent Advances in Slavonic Natural Language Processing, RASLAN 2012*, pages 23–30, Brno, 2012. Tribun EU.
[author's contribution: 40% – system design and text writing].

[12] Aleš Horák, Miloš Jakubíček, and Vojtěch Kovář. Linguistic Logical Analysis of Direct Speech. In Aleš Horák and Pavel Rychlý, editors, *Proceedings of the Sixth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2012*, pages 51–59, Brno, 2012. Tribun EU.
[author's contribution: 10% – corpus compilation and text writing].

[13] Miloš Jakubíček, Pavel Šmerk, and Pavel Rychlý. Fast Construction of a Word-Number Index for Large Data. In A. Horák and P. Rychlý, editors, *Proceedings of the Seventh Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2013*, pages 63–67, Brno, 2013. Tribun EU.
[author's contribution: 33% – system implementation and text writing].

[14] Miloš Jakubíček and Marek Medveď. Portable Lexical Analysis for Parsing of Morphologically-Rich Languages. In A. Horák and P. Rychlý, editors, *Proceedings of the Seventh Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2013*, pages 21–26, Brno, 2013. Tribun EU.
[author's contribution: 50% – system design and text writing].

[15] Miloš Jakubíček and Pavel Rychlý. Optimization of Regular Expression Evaluation within the Manatee Corpus Management System. In

A. Horák and P. Rychlý, editors, *Proceedings of the Eighth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2014*, pages 37–48, Brno, 2014. Tribun EU.
[author's contribution: 75% – system design, implementation and text writing].

[16] Vojtěch Kovář, Pavel Rychlý, and Miloš Jakubíček. Low Inter-Annotator Agreement = An Ill-Defined Problem? In A. Horák and P. Rychlý, editors, *Proceedings of the Eighth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2014*, pages 57–62, Brno, 2014. Tribun EU.
[author's contribution: 33% – methodology and text writing].

[17] Vít Baisa, Ondřej Herman, and Miloš Jakubíček. Towards Automatic Finding of Word Sense Changes in Time. In Aleš Horák, Pavel Rychlý, and Adam Rambousek, editors, *Proceedings of the Ninth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2015*, pages 33–41, Brno, 2015. Tribun EU.
[author's contribution: 33% – system design, evaluation and text writing].

[18] Darja Fišer, Vít Suchomel, and Miloš Jakubíček. Terminology Extraction for Academic Slovene Using Sketch Engine. In Aleš Horák, Pavel Rychlý, and Adam Rambousek, editors, *Proceedings of the Tenth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2016*, pages 135–141, Brno, 2016. Tribun EU.
[author's contribution: 25% – system implementation].

[19] Miloš Jakubíček and Pavel Šmerk. Large Scale Keyword Extraction using a Finite State Backend. In Aleš Horák, Pavel Rychlý, and Adam Rambousek, editors, *Proceedings of the Tenth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2016*, pages 143–146, Brno, 2016. Tribun EU.
[author's contribution: 75% – system design, implementation and text writing].

A.6 Other publications

[1] Miloš Jakubíček. Extrakce strukturních informací z běžného textu na základě syntaktického analyzátoru (bachelor thesis), 2008.

[2] Miloš Jakubíček. Enhancing czech parsing with complex valency frames (master thesis), 2010.

[3] Miloš Jakubíček. Rule-based parsing of morphologically rich languages (dissertation thesis proposal), 2012.

Bibliography

[Abney, 1995] Abney, S. (1995). Dependency grammars and context-free grammars. Manuscript. Presented at meeting of Linguistic Society of America.

[Abney, 1996] Abney, S. (1996). Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(4):337–344.

[Abney et al., 1991] Abney, S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., et al. (1991). Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the workshop on Speech and Natural Language*, pages 306–311. Association for Computational Linguistics.

[Ailomaa et al., 2005] Ailomaa, M., Kadlec, V., Rajman, M., and Chappelier, J.-C. (2005). Robust stochastic parsing: Comparing and combining two approaches for processing extra-grammatical sentences. In Werner, S., editor, *Proceedings of the 15th NODALIDA Conference, Joensuu 2005*, pages 1–7, Joensuu. Ling@JoY.

[Akmajian and Heny, 1980] Akmajian, A. and Heny, F. (1980). *An introduction to the principles of transformational syntax*. MIT Press.

[Ambati et al., 2012] Ambati, B. R., Reddy, S., and Kilgarriff, A. (2012). Word Sketches for Turkish. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

[Andor et al., 2016] Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

[Artstein and Poesio, 2008] Artstein, R. and Poesio, M. (2008). Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4):555–596.

[Bach, 1974] Bach, E. (1974). *Syntactic Theory*. Holt, Rinehart and Winston.

[Baerman, 2015] Baerman, M. (2015). *The Oxford Handbook of Inflection*. Oxford Handbooks. OUP Oxford.

[Belz, 2002] Belz, A. (2002). Learning Grammars for Different Parsing Tasks by Partition Search. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

[Belz, 2009] Belz, A. (2009). That's Nice... What Can You Do With It? *Computational Linguistics*, 35(1):111–118.

[Berzak et al., 2016] Berzak, Y., Huang, Y., Barbu, A., Korhonen, A., and Katz, B. (2016). Anchoring and agreement in syntactic annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2215–2224, Austin, Texas. Association for Computational Linguistics.

[Bick, 2000] Bick, E. (2000). The parsing system palavras. *Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*.

[Bohnet and Nivre, 2012] Bohnet, B. and Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea. Association for Computational Linguistics.

[Brants and Hansen, 2002] Brants, S. and Hansen, S. (2002). Developments in the TIGER Annotation Scheme and Their Realization in the Corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1643–1649.

[Brants, 2000] Brants, T. (2000). Inter-Annotator Agreement for a German Newspaper Corpus. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, pages 69–76.

[Bresnan, 1977] Bresnan, J. (1977). *A realistic transformational grammar*. University of Texas at Austin.

[Bresnan, 2001] Bresnan, J. (2001). *Lexical-Functional Syntax*. Wiley-Blackwell.

[Brill, 1993] Brill, E. (1993). Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. In *Proceedings of the workshop on Human Language Technology*, pages 237–242. Association for Computational Linguistics.

[Briscoe et al., 2006] Briscoe, T., Carroll, J., and Watson, R. (2006). The Second Release of the RASP System. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80. Association for Computational Linguistics.

[Briscoe and Gazdar, 2001] Briscoe, T. and Gazdar, G. (2001). Ted Briscoe interviews Gerald Gazdar. [online] at https://nlp.fi.muni.cz/~xjakub/briscoe-gazdar.

[Buchholz and Marsi, 2006] Buchholz, S. and Marsi, E. (2006). 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.

[Bušta, 2008] Bušta, J. (2008). Computing Idioms Frequency in Text Corpora. In *Proceedings of Second Workshop On Recent Advances in Slavonic Natural Language Processing, RASLAN 2008*, pages 71–74, Brno. Masaryk University.

[Callan et al., 2009] Callan, J., Hoy, M., Yoo, C., and Zhao, L. (2009). Clueweb09 data set. Online presentation available at: http://boston.lti.cs.cmu.edu/classes/11-742/S10-TREC/TREC-Nov19-09.pdf.

[Callmeier, 2000] Callmeier, U. (2000). PET – a Platform for Experimentation with Efficient HPSG Processing Techniques. *Natural Language Engineering*, 6(1):99–107.

[Carletta, 1996] Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22(2):249–254.

[Charniak, 2000] Charniak, E. (2000). 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.

[Chen and Manning, 2014] Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

[Choi et al., 2015] Choi, J. D., Tetreault, J., and Stent, A. (2015). It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 387–396, Beijing, China. Association for Computational Linguistics.

[Chomsky, 1957] Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague/Paris.

[Chomsky, 1995] Chomsky, N. (1995). *The Minimalist Program*. Current studies in linguistics series. The MIT Press.

[Christ and Schulze, 1994] Christ, O. and Schulze, B. M. (1994). *The IMS Corpus Workbench: Corpus Query Processor (CQP) User's Manual*. University of Stuttgart, Germany, http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench.

[Clark and Curran, 2007] Clark, S. and Curran, J. (2007). Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

[Cohen et al., 1960] Cohen, J. et al. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and psychological measurement*, 20(1):37–46.

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

[Culicover, 1976] Culicover, P. (1976). *Syntax*. Academic Press, New York.

[de Marneffe et al., 2014] de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal Stanford dependencies: A cross-linguistic typology. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk,

J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

[Debusmann, 2006] Debusmann, R. (2006). *Extensible Dependency Grammar: A Modular Grammar Formalism Based On Multigraph Description*. PhD thesis, Saarland University.

[Duráni, 2012] Duráni, J. (2012). *Generování stromů ze závislostního grafu syntaktické analýzy*. Bachelor thesis, Masaryk University.

[Earley, 1970] Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

[Eisner, 2016] Eisner, J. (2016). Inside-outside and forward-backward algorithms are just backprop. In *Proceedings of the EMNLP 16 Workshop on Structured Prediction for NLP, Austin*.

[Filipec et al., 1995] Filipec, J. et al. (1995). *Slovník spisovné češtiny (Dictionary of Literary Czech, SSČ)*. Academia, Praha, 1^{st} edition. electronic version, LEDA, Praha.

[Fleiss, 1981] Fleiss, J. (1981). The Measurement of Interrater Agreement. *Statistical Methods for Rates and Proportions*, 2:212–236.

[Gazdar et al., 1985] Gazdar, G., Klein, E., Pullum, G., and Sag, I. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Massachusetts.

[Gesmundo et al., 2009] Gesmundo, A., Henderson, J., Merlo, P., and Titov, I. (2009). A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 37–42, Boulder, Colorado. Association for Computational Linguistics.

[Grinberg et al., 1995] Grinberg, D., Lafferty, J., and Sleator, D. (1995). A Robust Parsing Algorithm for Link Grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*.

[Grinder and Elgin, 1973] Grinder, J. and Elgin, S. (1973). *Guide to transformational grammar: history, theory, practice*. Holt, Rinehart and Winston, New York.

[Grác et al., 2010] Grác, M., Jakubíček, M., and Kovář, V. (2010). Through low-cost annotation to reliable parsing evaluation. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 555–562, Tohoku University, Sendai, Japan. Institute of Digital Enhancement of Cognitive Processing, Waseda University.

[Hajič, 1998] Hajič, J. (1998). Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 106–132, Prague. Karolinum.

[Hajič et al., 1998] Hajič, J., Brill, E., Collins, M., Hladká, B., Jones, D., Kuo, C., Ramshaw, L., Schwartz, O., Tillmann, C., and Zeman, D. (1998). Core Natural Language Processing Technology Applicable to Multiple Languages. The Workshop 98 Final Report. Technical report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, Maryland. [online] http://www.clsp.jhu.edu/ws98/projects/nlp/report/.

[Hajičová et al., 2004] Hajičová, E., Havelka, J., Sgall, P., Veselá, K., and Zeman, D. (2004). Issues of Projectivity in the Prague Dependency Treebank. *Prague Bulletin of Mathematical Linguistics*, 81:5–22.

[Hall and Novák, 2005] Hall, K. and Novák, V. (2005). Corrective Modeling for Non-Projective Dependency Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 42–52. Association for Computational Linguistics.

[Hana, 2007] Hana, J. (2007). *Czech Clitics in Higher Order Grammar*. PhD thesis, The Ohio State University.

[Havelka, 2007] Havelka, J. (2007). Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615, Prague, Czech Republic. Association for Computational Linguistics.

[Havránek et al., 1971] Havránek, B. et al. (1960-1971). *Slovník spisovného jazyka českého (Dictionary of Written Czech, SSJČ)*. Academia, Praha, 1^{st} edition. electronic version, created in the Institute of Czech Language, Czech Academy of Sciences Prague in cooperation with Faculty of Informatics, Masaryk University Brno.

[Hill, 1961] Hill, A. A. (1961). Grammaticality. *Word*, 17(1):1–10.

[Horák and Kadlec, 2005] Horák, A. and Kadlec, V. (2005). New Meta-grammar Constructs in Czech Language Parser Synt. In Matoušek, V., Mautner, P., and Pavelka, T., editors, *Proceedings of the 8th International Conference on Text, Speech and Dialogue, TSD 2005*, pages 85–92, Berlin, Heidelberg. Springer-Verlag.

[Horák and Rychlý, 2009] Horák, A. and Rychlý, P. (2009). Discovering Grammatical Relations in Czech Sentences. In *Proceedings of the Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009*, pages 81–88.

[Hudson, 1984] Hudson, R. (1984). *Word Grammar*. Blackwell Oxford.

[Jakubíček et al., 2013] Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., and Suchomel, V. (2013). The TenTen corpus family. In *Proceedings of the 7th International Corpus Linguistics Conference, CL 2013*, pages 125–127.

[Jakubíček et al., 2014] Jakubíček, M., Kilgarriff, A., and Rychlý, P. (2014). Effective corpus virtualization. In *Proceedings of the Second Workshop on the Challenges in the Management of Large Corpora (CMLC-2)*, pages 7–9. European Language Resources Association – ELRA.

[Jakubíček et al., 2009] Jakubíček, M., Horák, A., and Kovář, V. (2009). Mining Phrases from Syntactic Analysis. In *Proceedings of the 12th International Conference on Text, Speech and Dialogue, TSD 2009*, pages 124–130, Berlin, Heidelberg. Springer-Verlag.

[Jakubíček et al., 2010] Jakubíček, M., Kilgarriff, A., McCarthy, D., and Rychlý, P. (2010). Fast Syntactic Searching in Very Large Corpora for Many Languages. In *PACLIC*, volume 24, pages 741–747.

[Jakubíček, 2011] Jakubíček, M. (2011). Effective Parsing Using Competing CFG Rules. In *Proceedings of the 14th International Conference on Text, Speech and Dialogue, TSD 2011*, pages 115–122, Berlin, Heidelberg. Springer-Verlag.

[Jakubíček et al., 2017] Jakubíček, M., Baisa, V., Bušta, J., Kovář, V., Michelfeit, J., Rychlý, P., and Suchomel, V. (to appear 2017). *Walking the Tightrope between Linguistics and Language Engineering*. Springer-Verlag.

[Jakubíček et al., 2011] Jakubíček, M., Kovář, V., and Šmerk, P. (2011). Czech Morphological Tagset Revisited. In Horák, A. and Rychlý, P., editors, *Pro-*

ceedings of the Fifth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2011, pages 29–42, Brno. Tribun EU.

[Jakubíček and Medveď, 2013] Jakubíček, M. and Medveď, M. (2013). Portable Lexical Analysis for Parsing of Morphologically-Rich Languages. In Horák, A. and Rychlý, P., editors, *Proceedings of the Seventh Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2013*, pages 21–26, Brno. Tribun EU.
[author's contribution: 50% – system design and text writing].

[Jakubíček and Rychlý, 2014] Jakubíček, M. and Rychlý, P. (2014). Optimization of Regular Expression Evaluation within the Manatee Corpus Management System. In *Proceedings of the Eighth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2014*, pages 37–48. Tribun EU.

[Jakubíček et al., 2013] Jakubíček, M., Šmerk, P., and Rychlý, P. (2013). Fast construction of a word-number index for large data. In A. Horák, P. R., editor, *Proceedings of the Seventh Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2013*, pages 63–67, Brno. Tribun EU.

[Jelinek, 1998] Jelinek, F. (1998). Applying information theoretic methods: Evaluation of grammar quality. In *Workshop on Evaluation of NLP Systems*.

[Jelinek, 2005] Jelinek, F. (2005). Some of my best friends are linguists. *Language resources and evaluation*, 39(1):25–34.

[Johnson and Lappin, 1997] Johnson, D. and Lappin, S. (1997). A Critique of the Minimalist Program. *Linguistics and Philosophy*, 20(3):273–333.

[Joshi and Schabes, 1997] Joshi, A. K. and Schabes, Y. (1997). Tree-adjoining grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, pages 69–123. Springer-Verlag, New York.

[Kadlec, 2007] Kadlec, V. (2007). *Syntactic Analysis of Natural Languages Based on Context-Free Grammar Backbone*. PhD thesis, Masaryk University, Brno.

[Katz-Brown et al., 2011] Katz-Brown, J., Petrov, S., McDonald, R., Och, F., Talbot, D., Ichikawa, H., Seno, M., and Kazawa, H. (2011). Training a parser for machine translation reordering. In *Proceedings of the 2011*

Conference on Empirical Methods in Natural Language Processing, pages 183–192, Edinburgh, Scotland, UK. Association for Computational Linguistics.

[Kay, 1967] Kay, M. (1967). Experiments with a powerful parser. In *Proceedings of the 1967 Conference on Computational Linguistics*, COLING '67, pages 1–20, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Kay, 1986] Kay, M. (1986). *Algorithm Schemata and Data Structures in Syntactic Processing*. Morgan Kaufmann Publishers Inc.

[Kennedy, 2014] Kennedy, G. (2014). *An Introduction to Corpus Linguistics*. Studies in Language and Linguistics. Taylor & Francis.

[Kilgarriff, 2015] Kilgarriff, A. (2015). [Corpora-list] User generated content corpora. http://mailman.uib.no/public/corpora/2015-February/021917.html.

[Kilgarriff et al., 2014a] Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., and Suchomel, V. (2014a). The Sketch Engine: Ten Years On. *Lexicography*, 1(1):7–36.

[Kilgarriff et al., 2014b] Kilgarriff, A., Jakubíček, M., Pomikálek, J., Sardinha, T. B., and Whitelock, P. (2014b). Pttenten: A corpus for portuguese lexicography. In *Working with Portuguese Corpora*, pages 280–287. Bloomsbury Publishing, London, 1 edition.

[Kilgarriff et al., 2010] Kilgarriff, A., Kovář, V., Krek, S., Srdanović, I., and Tiberius, C. (2010). A quantitative evaluation of word sketches. In *Proceedings of the XIV Euralex International Congress*, pages 372–79, Leeuwarden. Fryske Akademy.

[Kilgarriff and Renau, 2013] Kilgarriff, A. and Renau, I. (2013). esTenTen, a Vast Web Corpus of Peninsular and American Spanish . *Procedia - Social and Behavioral Sciences* , 95(0):12 – 19. Corpus Resources for Descriptive and Applied Studies. Current Challenges and Future Directions: Selected Papers from the 5th International Conference on Corpus Linguistics (CILC2013) .

[Kilgarriff et al., 2014c] Kilgarriff, A., Rychlý, P., Jakubíček, M., Kovář, V., Baisa, V., and Kocincová, L. (2014c). Extrinsic Corpus Evaluation with a Collocation Dictionary Task. In Calzolari, N., Choukri, K., Declerck,

T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

[Kilgarriff et al., 2012] Kilgarriff, A., Rychlý, P., Kovář, V., and Baisa, V. (2012). Finding multiwords of more than two words. Oslo. University of Oslo.

[Kilgarriff et al., 2004] Kilgarriff, A., Rychlý, P., Smrž, P., and Tugwell, D. (2004). The Sketch Engine. *Information Technology*, 105:116.

[Kirilina and Versleya, 2015] Kirilina, A. and Versleya, Y. (2015). What is hard in universal dependency parsing. In *6th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2015)*, pages 31–38.

[Klein and Manning, 2003] Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

[Knuth et al., 2003] Knuth, D. E. et al. (2003). *Selected papers on computer languages*. CSLI Publications.

[Kovář, 2014] Kovář, V. (2014). *Automatic syntactic analysis for real-world applications*. PhD thesis, Masaryk University.

[Kovář et al., 2009] Kovář, V., Horák, A., and Jakubíček, M. (2009). Syntactic Analysis as Pattern Matching: The SET Parsing System. In *Proceedings of the 4^{th} Language & Technology Conference*, pages 100–104, Poznaň, Poland.

[Kovář and Jakubíček, 2008] Kovář, V. and Jakubíček, M. (2008). Test Suite for the Czech Parser Synt. In Sojka, P. and Horák, A., editors, *Proceedings of the Second Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2008*, pages 63–70, Brno.

[Leech, 1992] Leech, G. (1992). 100 million words of English: the British National Corpus (BNC). *Language Research*, 28(1):1–13.

[Manning, 2011] Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In Gelbukh, A. F., editor, *Proceedings*

of the 12th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing 2011, pages 171–189, Berlin, Heidelberg. Springer-Verlag.

[Marcus et al., 1993] Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

[McCarthy and Navigli, 2007] McCarthy, D. and Navigli, R. (2007). SemEval-2007 Task 10: English Lexical Substitution Task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53.

[McDonald et al., 2006] McDonald, R., Lerman, K., and Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City. Association for Computational Linguistics.

[McDonald et al., 2005] McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

[Medveď et al., 2012] Medveď, M., Jakubíček, M., Kovář, V., and Němčík, V. (2012). Adaptation of Czech Parsers for Slovak. In Horák, A. and Rychlý, P., editors, *Proceedings of the Sixth Recent Advances in Slavonic Natural Language Processing, RASLAN 2012*, pages 23–30, Brno. Tribun EU. [author's contribution: 40% – system design and text writing].

[Mel'čuk, 1988] Mel'čuk, I. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press.

[Merlo and Ferrer, 2006] Merlo, P. and Ferrer, E. E. (2006). The notion of argument in prepositional phrase attachment. *Computational Linguistics*, 32(3):341–378.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on*

Neural Information Processing Systems, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

[Miyao et al., 2009] Miyao, Y., Sagae, K., Saetre, R., Matsuzaki, T., and Tsujii, J. (2009). Evaluating Contributions of Natural Language Parsers to Protein-Protein Interaction Extraction. *Bioinformatics*, 25(3):394–400.

[Musillo and Merlo, 2006] Musillo, G. and Merlo, P. (2006). Accurate parsing of the proposition bank. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 101–104, New York City, USA. Association for Computational Linguistics.

[Nivre, 2009] Nivre, J. (2009). 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, Suntec, Singapore. Association for Computational Linguistics.

[Nivre et al., 2016] Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal Dependencies v1: A Multilingual Treebank Collection. In Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

[Nivre and Nilsson, 2005] Nivre, J. and Nilsson, J. (2005). Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan. Association for Computational Linguistics.

[Pala et al., 1997] Pala, K., Rychlý, P., and Smrž, P. (1997). DESAM — Annotated Corpus for Czech. In *Proceedings of SOFSEM'97*, pages 523–530. Springer-Verlag.

[Pecina, 2011] Pecina, P. (2011). Review: Syntax-Based Collocation Extraction Violeta Seretan (University of Geneva) Berlin: Springer-Verlag (Text, speech and language technology series, volume 44), 2011; ISBN 978-94-007-0133-5. *Computational Linguistics*, pages 1–3.

[Pollard and Sag, 1994] Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

[Pomikálek et al., 2012] Pomikálek, J., Jakubíček, M., and Rychlý, P. (2012). Building a 70 billion word corpus of English from ClueWeb. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

[Popel et al., 2013] Popel, M., Mareček, D., Štěpánek, J., Zeman, D., and Žabokrtský, Z. (2013). Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 517–527, Sofia, Bulgaria. Association for Computational Linguistics.

[Popper, 1959] Popper, K. (1959). *The Logic of Scientific Discovery*. Basic Books, Inc., New York.

[Pullum and Gazdar, 1982] Pullum, G. and Gazdar, G. (1982). Natural Languages and Context-Free Languages. *Linguistics and Philosophy*, 4(4):471–504.

[Radziszewski and Acedański, 2012] Radziszewski, A. and Acedański, S. (2012). Taggers Gonna Tag: An Argument against Evaluating Disambiguation Capacities of Morphosyntactic Taggers. In Sojka, P., Horák, A., Kopeček, I., and Pala, K., editors, *Proceedings of the 15th International Conference on Text, Speech and Dialogue, TSD 2012*, pages 81–87, Berlin, Heidelberg. Springer-Verlag.

[Rychlý, 2007] Rychlý, P. (2007). Manatee/Bonito – a Modular Corpus Manager. In *Proceedings of the First Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2007*, pages 65–70, Brno. Masaryk University.

[Rychlý and Kilgarriff, 2007] Rychlý, P. and Kilgarriff, A. (2007). An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 41–44, Prague, Czech Republic. Association for Computational Linguistics.

[Rychlý, 2000] Rychlý, P. (2000). *Korpusové manažery a jejich efektivní implementace*. PhD thesis, Masaryk University, Brno.

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

[Sagae et al., 2007] Sagae, K., Miyao, Y., and Tsujii, J. (2007). HPSG Parsing with Shallow Dependency Constraints. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 624–631, Prague, Czech Republic. Association for Computational Linguistics.

[Sampson, 2000] Sampson, G. (2000). A Proposal for Improving the Measurement of Parse Accuracy. *International Journal of Corpus Linguistics*, 5(01):53–68.

[Sampson, 2001] Sampson, G. (2001). *Empirical Linguistics*. Open Linguistics Series. Continuum.

[Sampson, 2005] Sampson, G. (2005). *The 'Language Instinct' Debate*. Open Linguistics Series. Continuum.

[Sampson and Babarczy, 2003] Sampson, G. and Babarczy, A. (2003). A Test of the Leaf-Ancestor Metric for Parse Accuracy. *Natural Language Engineering*, 9(04):365–380.

[Sampson and Babarczy, 2008] Sampson, G. and Babarczy, A. (2008). Definitional and Human Constraints on Structural Annotation of English. *Natural Language Engineering*, 14(4):471–494.

[Satta, 1994] Satta, G. (1994). Tree-Adjoining Grammar Parsing and Boolean Matrix Multiplication. *Computational Linguistics*, 20:173–191.

[Schmid, 1994] Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Machester.

[Schmid and Laws, 2008] Schmid, H. and Laws, F. (2008). Estimation of Conditional Probabilities with Decision Trees and an Application to Fine-grained POS Tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics – Volume 1*, COLING '08, pages 777–784, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Sgall et al., 1986] Sgall, P., Hajičová, E., Panevová, J., and Mey, J. (1986). *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reidel.

[Sha and Pereira, 2003] Sha, F. and Pereira, F. (2003). Shallow Parsing with Conditional Random Fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

[Shieber, 1985] Shieber, S. (1985). Evidence Against the Context-Freeness of Natural Language. *Linguistics and Philosophy*, 8(3):333–343.

[Sikkel, 1997] Sikkel, K. (1997). *Parsing Schemata – A Framework for Specification and Analysis of Parsing Algorithms*. Springer-Verlag.

[Skjærholt, 2014] Skjærholt, A. (2014). A chance-corrected measure of inter-annotator agreement for syntax. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 934–944, Baltimore, Maryland. Association for Computational Linguistics.

[Sleator and Temperley, 1993] Sleator, D. and Temperley, D. (1993). Parsing English with a Link Grammar. In *Third International Workshop on Parsing Technologies*.

[Šmerk, 2004] Šmerk, P. (2004). Unsupervised Learning of Rules for Morphological Disambiguation. In *Proceedings of the 7^{th} International Conference on Text, Speech and Dialogue, TSD 2004*, pages 211–216, Berlin. Springer-Verlag.

[Šmerk, 2009] Šmerk, P. (2009). Fast morphological analysis of Czech. In *Proceedings of the Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009*, pages 13–16, Brno. Masaryk University.

[Šmilauer, 1969] Šmilauer, V. (1969). *Novočeská skladba: Vysokoškolská příručka*. Edice Vysokoškolské přiručky. SPN, Český Těšín.

[Stageberg, 1968] Stageberg, N. C. (1968). Structural ambiguity in the noun phrase. *TESOL Quarterly*, 2(4):232–239.

[Steedman, 2000] Steedman, M. (2000). *The Syntactic Process*. MIT Press.

[Straková et al., 2014] Straková, J., Straka, M., and Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland. Association for Computational Linguistics.

[Tange, 2011] Tange, O. (2011). GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine*, 36(1):42–47.

[Tesnière, 1959] Tesnière, L. (1959). *Élements de syntaxe structurale: Préf. de Jean Fourquet*. C. Klincksieck.

[Trautwein, 1995] Trautwein, M. H. (1995). The complexity of structure sharing in unification-based grammars. In *Proceedings of the Proceedings CLIN V Conference*, pages 165–180.

[Traxler et al., 2002] Traxler, M. J., Morris, R. K., and Seely, R. E. (2002). Processing subject and object relative clauses: Evidence from eye movements. *Journal of Memory and Language*, 47(1):69 – 90.

[Tsarfaty et al., 2013] Tsarfaty, R., Seddah, D., Kübler, S., and Nivre, J. (2013). Parsing morphologically rich languages: Introduction to the special issue. *Computational Linguistics*, 39(1):15–22.

[Vadas and Curran, 2007] Vadas, D. and Curran, J. (2007). Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, Czech Republic. Association for Computational Linguistics.

[Vijay-Shanker and Weir, 1994] Vijay-Shanker, K. and Weir, D. J. (1994). The Equivalence of Four Extensions of Context-Free Grammars. *Theory of Computing Systems*, 27:511–546. 10.1007/BF01191624.

[Vinyals et al., 2014] Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. E. (2014). Grammar as a foreign language. *CoRR*, abs/1412.7449.

[Vinyals et al., 2015] Vinyals, O., Kaiser, L. u., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015). Grammar as a foreign language. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

[Weischedel et al., 2011] Weischedel, R., Hovy, E., Marcus, M., Palmer, M., Belvin, R., Pradhan, S., Ramshaw, L., and Xue, N. (2011). Ontonotes: A large training corpus for enhanced processing. *Handbook of Natural Language Processing and Machine Translation. Springer-Verlag*.

[Wilson, 1999] Wilson, E. O. (1999). *Consilience: The Unity of Knowledge*. ISSR library. Vintage Books.

[Younger, 1967] Younger, D. (1967). Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control*, 10(2):189–208.

[Žáčková, 2002] Žáčková, E. (2002). *Partial syntactic analysis of Czech*. PhD thesis, Masaryk University.

[Zipf, 1949] Zipf, G. K. (1949). *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Reading.