Using Dependency Tree Structure for Czech – English Machine Translation

Martin Čmejrek

Doctoral Thesis

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University
Prague 2006

Supervisor Doc. RNDr. JAN HAJIČ, Dr.
Institute of Formal and Applied Linguistics MFF UK
Malostranské náměstí 25
118 00 Praha 1
Czech Republic

Opponents Doc. PhDr. KAREL PALA, CSc.
Masaryk University
Faculty of Informatics
Botanická 68a
602 00 Brno
Czech Republic

Ing. PAVEL IRCING, Ph.D.
Department of Cybernetics
University of West Bohemia
Univerzitní 8
306 14 Plzeň
Czech Republic

I certify that this doctoral thesis is all my own work, and that I used only the cited literature. The thesis is freely available for all who can use it.

Prague, April 24, 2006

Abstract

The use of statistical methods in *machine translation* in recent years has led to great improvements in these methods, and also the quantitative evaluation of results shows that they outperform rule-based systems in the field of unlimited textual domains.

Nevertheless, statistical methods often produce errors that are in contradiction to the simplest linguistic knowledge, such as missing verbs, invalid word order, the incorrect choice of functional words, or constructions that violate constraints of agreement. Though translation models that transform a string of words in one language into a string of words in another language, together with language models based on surface n-grams work well in local contexts, they are not capable of handling grammatical rules with a larger scope. On the other hand, *parsing algorithms* that give the syntactic structure of the sentences with relatively high precision exist for many languages. The aim of this work is to explore possibilities of making use of syntactical information—in our case the dependency structure used by the annotation scheme of the **Prague Dependency Treebank**—in machine translation.

We will describe two approaches to this problem. The first one is an implementation of a *Czech-English machine translation system* combining the *statistical parser* with *rule-based transfer* and *generation*, the second one is a proposal of a new *statistical* method for *tree-to-tree transductions*, that would be able to handle structural transformations in a larger context, and that could be also combined with explicit linguistic rules. We will show the appropriateness of the newly proposed method on the task of learning tree transformations by finding alignments between nodes of the corresponding trees.

The third goal of this work was to prepare the necessary data for experiments in structural machine translation. The existing algorithms for statistical machine translation require large amounts of unannotated *parallel texts*, while the parsing algorithms need syntactically annotated data, known as *treebanks*. Intuitively, the statistical methods for machine translation, that make use of syntactic information, require a *parallel treebank* to learn the transformations of the sentence structures.

As the *annotation schemes* are usually language-specific, such as in the case of Czech and English, it is necessary to find out if it is possible to have a common annotation scheme for both languages. And to find out if both existing annota-

tion schemes are compatible, so that we can automatically convert them into the common one. We observe the most important differences between the annotation schemes of the **Prague Dependency Treebank** and the **Penn Treebank**, and describe a newly created parallel treebank – the **Prague Czech-English Dependency Treebank**.

Acknowledgments

I would like to thank to Jan Hajič, who was an excellent advisor, and who introduced me to the field of mathematical linguistics 10 years ago.

Jarmila Panevová was the first reader, she gave me good advice through the length of my study at the Institute of Formal and Applied Linguistics.

Jason Eisner is the inventor of the tree-to-tree transductions and wrote the first implementation during the summer workshop at CLSP in 2002.

I would like to give thanks to Jan Cuřín, who has been my major collaborator for almost 10 years, to Jiří Havelka, for his important contribution to the experiments with the Czech-English MT, and to all colleagues at the Institute, who were a wonderful team, and upon whose results this work builds.

Finally, I would like to thank to my parents for giving me support during the work, and most importantly to my wife Kamila and our children Klára and Cyril, to whom I promised to finish this work before the snow is gone.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Machine translation is a very broad research field. It can be categorized according to many different criteria, such as the combination of source and target languages, similarity of the languages, the domain of translated texts, the amount of possible or required human interaction, or the technology used in the translation system. The aim of this work is to contribute to the research on Czech-English machine translation in the field of fully automatic translation in the free textual domain. In the following text, I will concentrate on the possibilities of combining statistical methods with linguistic rules, and on using syntactic information in machine translation.

This chapter is introductory. Firstly, we offer a short overview of the machine translation discipline from a historical perspective, then we describe the main approaches and methodologies used in the field. Finally, we spend more time introducing the most important statistical methods, and discussing their advantages and disadvantages for using them in Czech-English machine translation.

In Chapter 2, we describe the process of creating the **Prague Czech-English Dependency Treebank**. We start with two different annotation schemes – of the **Prague Dependency Treebank** and the **Penn Treebank**, compare them, and discuss the possibility of defining a common annotation scheme. We also describe automatic procedures for converting the Penn Treebank annotation into the dependency style.

Chapter 3 describes the implementation of a Czech-English machine translation. The system is a combination of a statistical parser and a rule-based transfer and generation system.

Most of the commonly used statistical models for translation are based on transformation of a string of words in one language into a string of words in another language, with language models that are usually built upon surface n-grams. These methods have a major disadvantage in that they are incapable of handling grammatical rules with a larger scope. In Chapter 4, we present detailed mathematics of a new statistical model of tree-to-tree transducer, which was designed to capture the linguistic information present in the dependency tree.

Chapter 5 then describes two implementations: the first one for modeling

transformations between the two layers – analytical and tectogrammatical – of annotation of Czech, the second one for transferring Czech tectogrammatical trees into English analytical ones.

In Chapter 6 we conclude our experience in Czech-English machine translation, and propose further directions in this topic.

1.1 A Short History of Machine Translation

Although the first ideas about mechanizing the process of translation can be traced back to the seventeenth century, the real development of the research field started in 1950s after the first computers were built. In 1947, Warren Weaver proposed the use of computers for translating natural languages. In 1954, IBM in collaboration with Georgetown University gave the first public demo of a machine translation (MT) system translating from Russian to English. The system used 250 words in a vocabulary and 6 grammatical rules, nevertheless, it was considered promising and attracted massive funding of the MT research field for the following decade.

In 1950s, the activities in the MT had concentrated on translation between Russian and English, and the support and demand came especially from the military. The main motivation was information gathering and screening of large amounts of scientific texts and technical documentation for a relatively small number of experts, who could tolerate the low quality of the output. The precision of the translation was not essential, since the purpose of translating the documents was to get the idea of the content in order to preselect relevant documents for human translator.

A typical MT system was built around a translation dictionary, the entries usually had one or more possible translations. The translation program used a word-to-word replacement in the first step (without using any deeper syntactical analysis), then applied a set of rules for word order changes. As the system developed, the size of the dictionary grew, and as did the system of (usually ad hoc written) rules. Later the systems were inspired by contemporary approaches in formal linguistics (cf. Noam Chomsky published his Syntactic Structures in 1957), these were mostly generative-transformational grammars.

In spite of all the efforts and high expectations, there was no major breakthrough reached, and the patience of the US funding agencies was at its end. In 1966, the Automatic Language Processing Advisory Committee (ALPAC), an organization set up to evaluate the prospects of MT, considered MT as slower, less accurate and twice as expensive as human translation and concluded that “there is no immediate or predictable prospect of useful machine translation”. Instead of supporting further experiments with MT systems, the ALPAC recommended funding of basic research in the field of natural language technologies.

The research in the field of MT in the USA stopped for about one decade, but it did, however, continue in Canada and Europe. The demands for MT systems came from the industrial sector and also there were more languages of interest as trade became multinational. The first commercially successful systems appeared. For example, the Meteo system for translating weather reports from English to French was developed at Montreal University at 1976, and the Systran, the most successful MT system so far, was installed in 1976 to translate documents of the Commission of the European Communities. A new market for cheaper MT systems appeared after microcomputers became available for personal use and after the using of word processor became the dominant way of writing texts.

The prevailing techniques at that time were rule-based. The translation process used morphological, syntactic, and semantic analysis into an interlingua-like representation, and a vice-versa sequence of tasks for generation of the target text. The theoretical description of natural language had advanced since 1950 and the designer of a research system could choose from a number of competing formalisms.

At the beginning of the 1990s, the increasing performance of computers allowed for the use of statistical methods and large corpora. Candide, a fully statistical system for MT was developed in IBM at the end of 1980s, and the results were published in 1991 [Berger et al., 1994]. The system was purely statistical, without using any linguistic knowledge, the statistical translation models were trained on a corpus of more that one million parallel sentences containing English and French transcriptions of speeches from the Canadian parliament. Another method invented at that time was an example-based approach using a large parallel corpus of previously translated sentences [Nagao, 1984]. Apart from research systems, many practical systems were developed to assist professional translators, such as electronic dictionaries, or translation memory.

1.2 Classification of Translation Systems

The ideal machine translation system would produce translations of a high quality for any sort of text, and without any human interaction. Nevertheless, this goal is not achievable in practice. As there are many different translation tasks, there are also various types of MT systems that suit them. An MT system can be categorized according to several characteristics, such as the domain of the translated texts, the amount of user-interaction required for producing output, or the quality of translations. There are also several types of approaches used for constructing the MT system. In the following, we will try to describe the general paradigms.

1.2.1 Domain of Translated Text

One of the most important factors is the domain of the translated texts. We can consider translation of web pages as an example of a task with an *unconstrained domain*. The difficulties we have to face are obvious. First of all, the dictionary. Since the vocabulary is unlimited and the language is permanently evolving, there will always be unknown – "out-of-vocabulary" words or meanings that will not be translated by the system, and since the words may have more meanings, for some words it will not be possible to disambiguate the correct one. Another problem is the impossibility to cover all (un)grammatical structures that may occur in the unlimited domain. Last but not least, the meaning of the sentence may depend on an outer context so that real world knowledge may be necessary to properly understand and translate the sentence.

The task simplifies as the domain narrows. For example, when translating scientific texts or technical documentation from one domain, the above mentioned problems become easier. Since most of the authors of the texts tend to use a limited set of common words and terms specific to a domain, the vocabulary can be covered more successfully by a common translation dictionary and a dictionary specific to the domain. Another advantage is that terminology is mostly constructed to be unambiguous within the domain. Also the sentences are always grammatical, and the number of used grammatical constructions is smaller, for example, sentences in 2^{nd} person almost never occur in scientific texts. For example, the English-Czech system APAČ (Automatický překladač angličtina – čeština) was designed to translate abstracts of scientific articles from the field of metallurgy [Hajič, 1987].

1.2.2 User Interaction Based on Translation Purpose

As mentioned above, MT systems can produce translations of a good quality only in highly constrained situations. Otherwise, a varying amount of user's interaction is necessary – based on the reason for using the MT system and the desired level of quality.

We can recognize two basic strategies of interaction. *Human-aided machine translation* (HAMT), where the translation process is performed by the machine, and the human interaction consists in pre-edition of the input, post-edition of the output, or interaction in the middle of the translation process. *Machine-aided human translation* (MAHT), where the human translates with help of tools running on the machine, such as translation memory (a database of previous translations), automatic revisions of terminology, etc.

The choice of the method of interaction always depends on the purpose of the translation. If the reason is to *screen* documents, the quality of the output is not

so important in this case, the main goal of the MT system is to provide translation to the reader (or a screening algorithm), who probably does not know the source language. The screening can be done with no costs. Relevant documents are then sent to a professional human translator, who will probably not make any use of the automatic translation result. The need for screening is still present, indeed there are several working systems: SYSTRAN (for the main language-pairs), ATLAS II (Japanese – English), or CAT (for accessing Japanese Databases in English).

MAHT systems are suitable if a professional translator wants to use an MT system for producing a *draft version* of the translation and postedit it into the final version. In order to reduce the translation costs (the time that is spent on the translation), the MT system has to produce a high-quality draft, so that the costs of corrections are lower than writing the translation from scratch. A tool commonly used for this task is called *translation memory*. It is a database of source sentences and their translations. When a new sentence has to be translated, the system tries to find the most similar example in the database. If successful, the tool offers the previously used translation of the example. There are several factors that increase the efficiency of using a translation memory: If the translated text is a modification of a previously translated document, such as in case of documentations for subsequent versions of the same product, if the translation memory is large, or if it is shared among several translators. The translation memory is usually capable of handling small mismatches, such as numbers of versions or changes of proper names, but it cannot handle changes in grammatical structure. Another feature offered by translation memory is terminology assistance. The system uses a terminological lexicon and checks that terms are translated consistently. Some systems are capable of finding new candidates for terminology and offer them to the translator.

Translation companies usually divide the work among several translators in order to finish the contract as soon as possible. Translation tools have to support parallel processing of the translation: versioning of the document, merging concurrent modifications, sharing the translation memory and terminological lexicon among translators, who may work either on-line or off-line, and also tracking the progress of the parallelized translation job.

1.2.3 Main Methodologies

MT systems can be either designed as bilingual (for a fixed pair of languages), or multilingual (for more language-pairs). Bilingual systems can be either unidirectional (translating only in one direction), or bidirectional.

Figure 1.1 (often referred to as Vauquois' triangle [Vauquois, 1975]) shows the three main processes taking part in translation: analysis, transfer, and generation. Although the boundaries between them are not always sharp, it is a useful map for

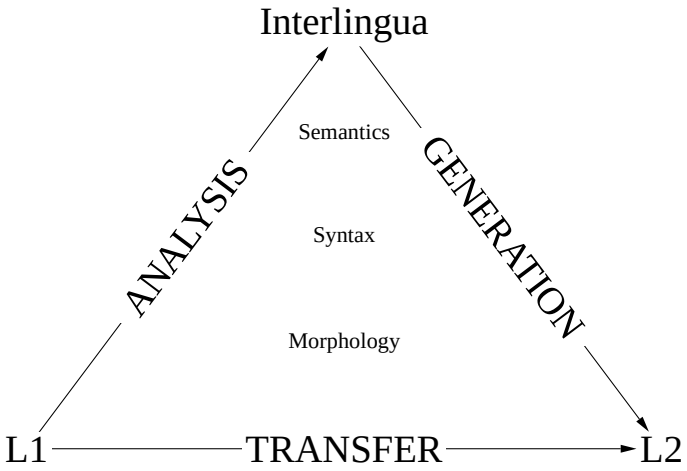


Figure 1.1: Vauquois' triangle

describing the architecture of an MT system. There are three basic approaches to MT: direct translation, interlingua, and transfer approach.

The *direct translation* method is the historically oldest approach. The core part of the system is a bilingual translation dictionary and a program for analyzing sentences of the source language and generating sentences in the target language. The system is designed to be fully specific to the selected source and target languages. The main advantage of the direct approach is that the source language has to be analyzed only to such a depth, which is sufficient for generating the target language. Thus for close languages, this can be relatively shallow. For example, the system ČESÍLKO [Hajič et al., 2003] translates from Czech to Slovak using only morphological analysis.

The *interlingua* approach is based on the assumption that the source language sentences can be converted into interlingua – a certain kind of syntactico-semantic representation, which is common to more languages, and that translations into target languages can be generated from this representation. This approach is more suitable for multilingual translation systems. The translation runs in two steps: analysis and synthesis. The main advantage of the interlingual representation is that once the sentence is analyzed into a common representation, it can be generated into all other languages, and the modules for analyzing input can be specific to a particular source language, as well as modules for generating output can be specific to the target language. On the other hand, it is very difficult to design the interlingua, the resulting representation is specific to a selected combination of languages, and adding a new language always increases the complexity of the common representation.

The *transfer* approach is a trade-off between the direct translation and the interlingua approach. The translation runs in three steps: analysis, transfer, and synthesis. Apart from the interlingua approach, input sentences are analyzed into a representation, which is still specific to the source language, and also the output sentences are synthesized from a representation specific to the target language. The analytical and synthetical steps respectively handle monolingual ambiguities within the source and target languages. The transfer step resolves ambiguities between the two languages, typically lexical issues.

1.3 Statistical Modeling in Machine Translation

1.3.1 IBM Models

At the beginning of 1990s, the computers became powerful enough to handle statistical models of machine translation. The models were described in [Brown et al., 1993], and the first results from French to English translation experiment were published in [Berger et al., 1994].

The first assumption of the statistical approach is that an English sentence $\mathbf{e}$ can be translated as any string of French words $\mathbf{f}$ with a probability $P(\mathbf{f} \mid \mathbf{e})$, and that the probability can be approximated by a statistical model and learned from a corpus of sentence-pairs.

The second assumption enables using the noisy-channel approach. We assume (however wrongly) that a French native speaker first formulates the sentence as a string of English words $\mathbf{e}$ and then, in a noisy-channel, converts it into a string of French words $\mathbf{f}$. Given the string $\mathbf{f}$, the MT system tries to reconstruct the original English sentence $\mathbf{e}$ by taking such $\hat{\mathbf{e}}$, for which the $P(\mathbf{e} \mid \mathbf{f})$ is the highest. Using the Bayes' theorem, we get

$$P(\mathbf{e} \mid \mathbf{f}) = \frac{P(\mathbf{e})P(\mathbf{f} \mid \mathbf{e})}{P(\mathbf{f})}. \tag{1.1}$$

Since the $\mathbf{f}$ is fixed, we look for the English translation as for

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} P(\mathbf{e})P(\mathbf{f} \mid \mathbf{e}). \tag{1.2}$$

The Equation (1.2) combines the translation model $P(\mathbf{f} \mid \mathbf{e})$ with a language model $P(\mathbf{e})$. In a figurative sense, it is similar to a human translator, who first tries to understand the French sentence, and then looks for a suitable English expression. The main advantage of this approach compared to modeling the translation directly as $P(\mathbf{e} \mid \mathbf{f})$ is that the translation model is designed to translate well, but not to judge the well-formedness of sentences. In another words, it is trained to concentrate its probability on such French translations that have a correct number

of correct words on roughly correct positions, but it is not even able to distinguish grammatical sentences from ungrammatical ones, moreover, it often prefers ill-formed sentences with repeating good words. That is why the language model has to be used for pruning ungrammatical hypotheses. Another reason is the training data. The translation model needs to learn from bilingual training data that are very expensive, and it is not possible to obtain more than a few million sentence-pairs, for some languages even less. On the other hand, the (n-gram) language model is trained on monolingual data, which is relatively cheap and available in high volumes. Thus the combination of a translation model and a language models can learn from much more data that a single model for direct translation.

Now we are going to describe the details of the translation models. When modeling such a complex process as the translation between two languages, it is useful to name all operations that have to be performed when transforming the source language into a target one. The "story" of the translation is then divided into several steps, each performing transformations of the same type. The whole probabilistic model is then expressed as a product of models of these partial steps.

Figure 1.2 tells one of the possible stories of translating an English sentence *"Mary did not slap the green witch"* into Spanish. In the first step, the translator had to decide for each word, how many words would be necessary to translate it. Words *Mary*, *not*, *the*, *green*, and *witch*, would be translated each as one word, *slap* would be translated using three words (so it was rewritten as triple *slap*), while the word *did* would not be translated at all (it was omitted). In the second step, the translator decided how many new words yet have to be added into the translation. In the third step, the translation dictionary was used to replace English words for Spanish ones. Finally, the translator determines the final word order of the Spanish sentence *"Mary no daba una bofetada a la bruja verde"*. The transformations in steps 1 – 4 are called *fertility*, *insertion*, *translation*, and *alignment*. In the following we will will describe their probabilities in terms of math.

Let $\mathbf{e}$ be an English sentence consisting of l words $e_1, \ldots, e_l$ and $\mathbf{f}$ be a French[1] sentence consisting of m words $f_1, \ldots, f_m$. Intuitively, the alignment between English and French words could be denoted by edges: each French word would be connected with those English words, from which it 'was born'. Since there may be $l * m$ different edges, there are 2^{lm} possible alignments[2], we must impose reasonable restrictions on this. Let $A \equiv \alpha_1, \ldots, \alpha_m$, such as $0 \leq \alpha_i \leq l$ be the alignment between French and English words. The interpretation of $\alpha_i > 0$ is that

[1] In the context of statistical translation models, the source sentence (in terms of the noisy-channel) is called English and marked $\mathbf{e}$, and the foreign sentence (the output of the noisy-channel) is usually marked as $\mathbf{f}$ (and sometimes even called *French*). Thus the mathematics remains consistent even if the pair of languages is different from the English – French pair. Statistical translation models are usually designed to be language-pair-independent

[2] For most of the sentences it is more than the estimated number of particles in the Universe.

a French word f_i was generated from an English word e_{α_i}. If $\alpha_i = 0$, the French word does not originate in any English word, because it was inserted into the sentence during translation. In another words, each French word has exactly one connection – either with an English word, or with a zero node indicating insertion.

In [Berger et al., 1994] they introduce a hierarchy of 5 models from the simplest to the most complex one. **Model 1** expressed by Equation 1.3, estimates the probability of translating English sentence $\mathbf{e}$ as French $\mathbf{f}$ with using a fixed alignment $\mathbf{a}$ as

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j}), \tag{1.3}$$

where ϵ is a constant approximating the probability of choosing the length of the French sentence $P(m \mid \mathbf{e})$, and the $t(f_j \mid e_{a_j})$ is a translation probability – the probability of translation English word e_{a_j} as the French f_j. The probability $P(\mathbf{f} \mid \mathbf{e})$ can be then estimated as a sum over all possible alignments

$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j}). \tag{1.4}$$

Model 1 is a very rough approximation, since it is only based on a table of word-to-word *translation probabilities*. If we look at our "story" of translation from Figure 1.2, we see that all other processes are almost ignored, since fertility and insertion are modeled by a single constant ϵ, and all possible word orders are considered equal.

Model 2 goes into more detail by modeling the *alignment probability* of the French word on position j coming from an English word on position i, and approximates it assuming that it depends only on the lengths of English and French sentences l and m, and the positions in these sentences i and j. The alignment probability is represented by a table $a(i \mid j, m, l)$, such that for each triple j, m, l holds.

$$\sum_{i=0}^{l} a(i \mid j, m, l) = 1. \tag{1.5}$$

The form of Model 2 is

$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j}).a(a_j \mid j, m, l). \tag{1.6}$$

Model 3 uses *fertility*–a probability of using ϕ_i French words for translating e_i, a set of *translation probabilities* $t(f \mid e_i)$, and a set of *distortion probabilities* $d(j \mid i, m, l)$. Parameters p_0 and p_1 are used for modeling the insertions of e

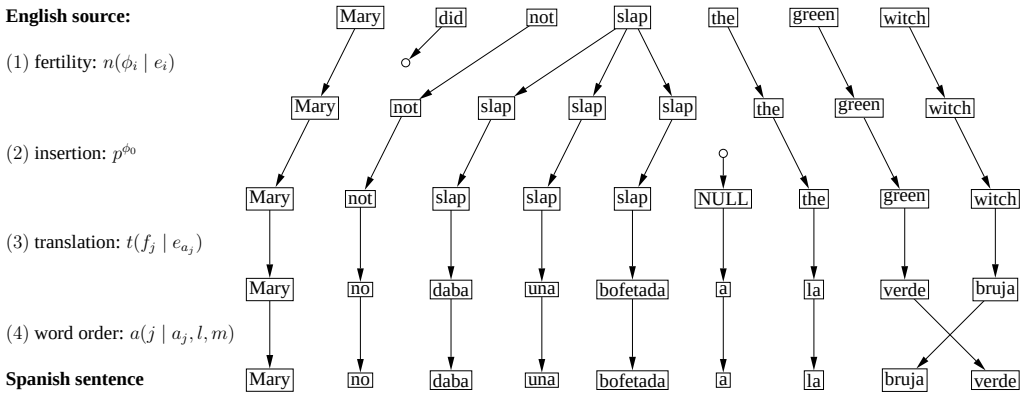


Figure 1.2: A model of a process translating an English sentence "*Mary did not slap the green witch.*" as a Spanish sentence "*Mary no daba una bofetada a la bruja verde.*"

new word, they are non-negative and sum to 1. Fertilities ϕ_i are functions of the alignment A. The formula for the model 3 is

$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \times \prod_{i=1}^{l} \phi_i! n\left(\phi_i \mid e_i\right) \times \prod_{j=1}^{m} t(f_j \mid e_{a_j}) d(j \mid a_j, m, l). \tag{1.7}$$

Models 4 and **5** improve the previously sketched approach by a finer modeling of distortions, trying to better describe the movements of larger groups of words.

The parameters of the models have to be trained from a parallel corpus.

The above mentioned models became a common ground for statistical modeling in machine translation. Apart from many partial improvements of the above mentioned models, further research in the field has not brought any substantially different approach.

The same approach was used in [Al-Onaizan et al., 1999] for building a Czech-English machine translation system. Since the translation models are designed as language independent, the general concept works for Czech-English language pair as well. On the other hand, there are typical imperfections caused by specific features of Czech. Firstly, Czech is more distant from English than French. Furthemore, Czech is a highly inflective language, most of its grammatical functions are expressed by specific suffixes. There are words that can appear in more than 10

different forms. This makes data sparseness even worse. Czech can choose word order almost freely, as syntactic and semantic roles are expressed by surface cases, not by the word order and prepositional phrases as it is in English. This is why the distortion tables and fertilities do not work as well as in English-French case. Czech is also a pro-drop language, which means that the subject of the sentence is often not present. In [Al-Onaizan et al., 1999] they used many tricks to adapt the statistical model to the specifics of Czech, mostly during the pre-processing phase. Czech words are *lemmatized* (all forms of one word replaced by a common representative), and special tokens are inserted to Czech according to the morphological information that was lost due to lemmatization. Thus the Czech becomes more similar to English. For example, to compensate for the pro-drop feature, if there is no nominal phrase in nominative, the *sb-token* is inserted. To simulate prepositional phrases as in English, the *of-token* is inserted before each nominal phrase in genitive, and many other.

1.3.2 Stochastic Inversion Transduction Grammars

Stochastic inversion transduction grammars (ITG) were firstly published in [Wu, 1997]. They were used for bracketing parallel texts—finding corresponding grammatical structures—of the English-Chinese corpus of transcriptions from Hong-Kong's parliament. Apart from the IBM models, this approach was syntactically motivated, trying to extract syntactic relations between two relatively distant languages – English and Chinese. The main assumption of this approach is that even if the two corresponding sentences in two languages have different grammatical structures, the syntactic roles can be still mapped one-to-one. ITG is capable of synchronous generation of the two sentences.

A **simple transduction grammar** (TG) (Lewis and Stearns, 1968) is a context-free grammar (CFG) that generates two output streams in two languages. It can be constructed as CFG, but in addition, its terminal symbols have to be marked by one of the languages. Thus the rule $A \rightarrow Bx_1y_2Cz_1$ generates terminals x and z of the language L_1 on stream 1, and terminal z of the language L_2 on stream 2. The same rule can be also written using a convenience notation as $A \rightarrow Bx/yCz/\epsilon$.

It is obvious, that simple transduction grammars can only generate sentence-pairs that share the same grammatical structure, the differences can only appear in the number of terminals. A small extension of the formalism can significantly enlarge the set of generated sentence-pairs, while still staying in the subset of context-free grammars. An **inversion transduction grammar** can be constructed from a TG by allowing two possible **orientations** of the production rules: **straight** and **inverted**. The straight orientation of the rule generates the right-hand-side constituents in left-to-right ordering in both languages, while the inverted orientation generates the L_2 output in right-to-left ordering. The inverted orientation

$$
\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{VP} &\rightarrow \text{RB VP} \\
\text{VP} &\rightarrow \text{VV PP} \\
\text{NP} &\rightarrow \text{Mary/Marie} \\
\text{RB} &\rightarrow \text{did/n'a not/pas} \\
\text{VV} &\rightarrow \text{slap/donne NN} \\
\text{NN} &\rightarrow \epsilon\text{/une } \epsilon\text{/gifle} \\
\text{PP} &\rightarrow \epsilon\text{/à NP} \\
\text{NP} &\rightarrow \text{the/la NN} \\
\text{NN} &\rightarrow \langle \text{green/verte witch/sorcière} \rangle
\end{aligned}
$$

Figure 1.3: An example of an inversion transduction grammar generating sentence-pair "*Mary did not slap the green witch*" and "*Marie n'a pas donne une gifle à la sorcière verte*".

is marked by operator $\langle\rangle$ around the right-hand-side of the rule. Figure 1.3 contains an example of an inversion transduction grammar and Figure 1.4 contains a parse-tree for a sample sentence-pair.

It can be shown that for every ITG G, there exists a grammar G' in **normal form**. It means that every production rule of G' has one of these forms:

$$
\begin{aligned}
\text{S} &\rightarrow \epsilon/\epsilon \\
\text{A} &\rightarrow \text{x/y} \\
\text{A} &\rightarrow \text{x}/\epsilon \\
\text{A} &\rightarrow \epsilon/\text{y} \\
\text{A} &\rightarrow \text{BC} \\
\text{A} &\rightarrow \langle \text{BC} \rangle
\end{aligned}
$$

A **stochastic inversion transduction grammar** (SITG) is a transduction grammar in normal form, and there is a probability assigned to each rule, such that for each nonterminal the sum of probabilities of all the rules, that rewrite this nonterminal is 1. The translation model assigns a probability of synchronous generation of a sentence pair $(\mathbf{e}, \mathbf{f})$ as a sum of probabilities of all derivations of $(\mathbf{e}, \mathbf{f})$. The probabilities of particular rules are obtained by EM from the parallel corpus.

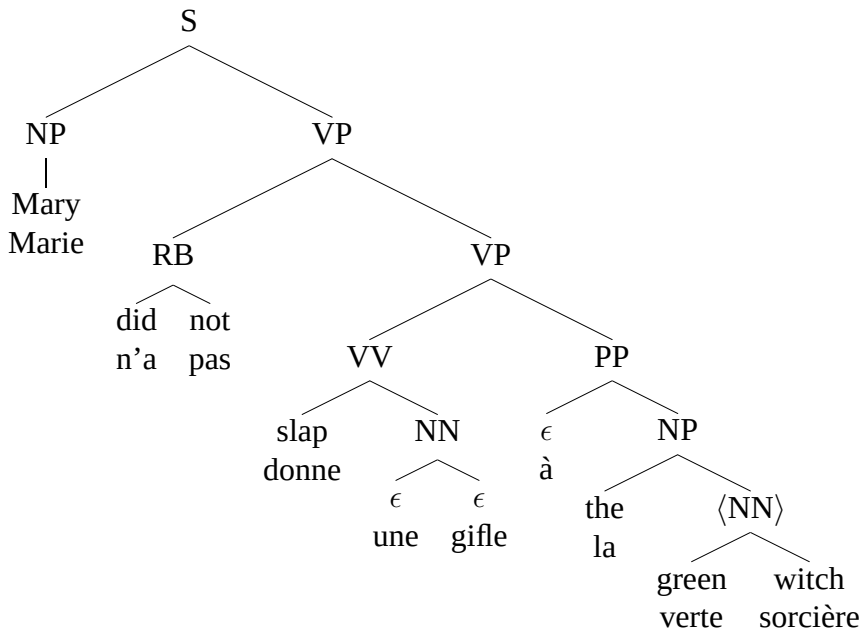


Figure 1.4: An example of an inversion transduction grammar parse-tree for a pair of sentences "*Mary did not slap the green witch"* and "*Marie n'a pas donne une gifle à la sorcière verte"*.

Unlike the IBM approach, the translation process is not modeled as a noisy-channel, instead it is based on a synchronous generation of both languages. The motivation for combining the translation model with a language model is not mathematical, but there are obvious practical reasons for it, such as the sparseness of the bilingual training data, and the advantage of using a huge amount of monolingual data.

The decoding algorithm and the results of a machine translation system based on SITGs were published in [Wu and Wong, 1998]. Some commercial systems using this approach were introduced recently, offering speech-to-speech English-Chinese translation of short phrases, and running on small devices, such as PDAs.

The main advantage of this approach is that the operation of inversion allows for learning grammatically different language-pairs, such as English and Chinese. The grammars suit languages that both use, howbeit different, fixed word ordering. English is a SVO language, while Chinese is SOV. The SITG approach was not yet tested on Czech-English language pair, but we may assume that the operation of inversion would not bring a significant advantage, since Czech is a free word order language.

original sequence	reordered sequence	$P(\text{reord} \mid \text{orig})$
NP VP	NP VP	...
	VP NP	...
VBD RB VP	VBD RB VP	...
	VBD VP RB	...
	VP VBD RB	...
	RB VBD VP	...
	RB VP VBD	...
	VP RB VBD	...
VB NP	VB NP	...
	NP VB	...
DT JJ NN	DT JJ NN	...
	DT NN JJ	...
	NN DT JJ	...
	JJ DT NN	...
	JJ NN DT	...
	NN JJ DT	...
...	...	...

Table 1.1: *r-table*

1.3.3 A Syntax-based Statistical Translation

A syntax-based statistical translation model [Yamada and Knight, 2001] was introduced in 2001, and a decoder [Yamada and Knight, 2002] in 2002. The model was tested on translations from Chinese to English. The translation system uses a noisy-channel approach and combines the English to Chinese translation probability with an English language model. Unlike the IBM approach, which models channel operations transforming strings of words (fertilities, insertions, deletions, word-to-word translations, and distortions), the syntax-based model describes the transformation of an English parse-tree into a Chinese string of words. The transformation has three steps: reordering of tree constituents, insertion of new constituents, and translation of the English lexical information into Chinese.

An example of an English-Spanish translation process is in Figure 1.5. The first, *reordering step*, is between trees (a) and (b). Two child sequences are reordered: the VP sequence VBD-RB-VP into RB-VBD-VP, and the NP sequence DT-JJ-NN into DT-NN-JJ, other sequences of child nodes do not change. A nonterminal with n child nodes has $n!$ possible reorderings. The probability of reordering depends only on the sequence of the child nodes (not on the parent node, etc.). The probabilities are stored in the so-called *r-table*, see Table 1.1.

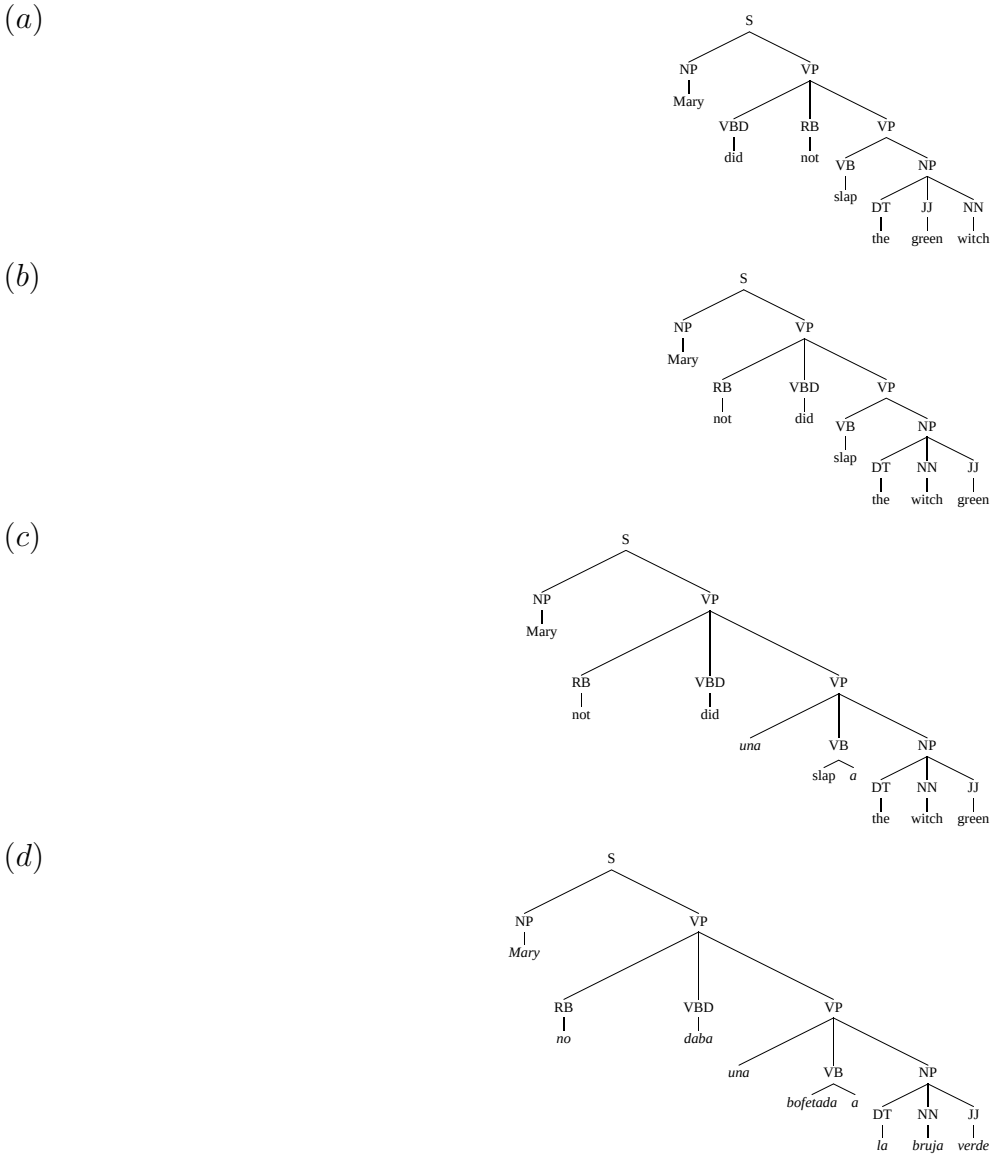


Figure 1.5: An example of (a) a parse tree for the sentence "*Mary did not slap the green witch*", (b) reordered according to target word order, (c) after insertions of words from a target language, (d) translated into a target language.

parent	TOP	S	S	VP	VP	VP	VP	VP	NP	NP	NP	...
node	S	NP	VP	RB	VBD	VP	VB	NP	DT	NN	JJ	...
P(node)	...	...	...	...	...	...	...	...	...	...	...	...
P(left)	...	...	...	...	...	...	...	...	...	...	...	...
P(right)	...	...	...	...	...	...	...	...	...	...	...	...

word	*una*	*a*	...
P(word)	...	...	...

Table 1.2: *n-tables*

The next, *insertion* step is displayed between trees (b) and (c) on Figure 1.5, where the Spanish word *una* was inserted as a left child of VP, and *a* was inserted as a right child of VB. In this step, each non-terminal of the reordered tree can either stay the same, or a new word from the target language can be inserted either to the left or right. The insertion probabilities depend on the no-terminal node, its parent, and the inserted target word. The insertion probabilities are modeled so-called $n - table$, which has two parts. The first part models the probability of inserting to the left or right, or no change for each possible pair of node and its parent. The second part models the insertion probabilities for particular target words.

The last, *translation* step is similar to that of IBM models, it uses co-called *t-table* to model word-to-word translation probabilities. An English word can be also translated as NULL, which corresponds to the deletion of the word.

The formal description of the model is as follows: Let the English parse tree $\mathcal{E}$ consist of non-terminals $\varepsilon_1, \ldots, \varepsilon_n$ and let the output foreign sentence $\mathbf{f}$ be a string of words $f_1, \ldots, f_m$. Let $\mathbf{R}$, $\mathbf{N}$, and $\mathbf{T}$ be the operations of reordering, insertion and translation. If ε_i is non-terminal, the operation $\theta_i = \langle \nu_i, \rho_i, \tau_i \rangle$ reorganizes child nodes of ε_i, inserts nodes to the left or right of ε_i or leaves it the same. If ε_i is terminal, the operation θ_i translates using a foreign word (or NULL).

The translation model assigns a probability of translating the English parse tree $\mathcal{E}$ as a foreign string of words $\mathbf{f}$

$$\begin{aligned} P(\mathbf{f} \mid \mathcal{E}) &= \sum_{\theta : Str(\theta(\varepsilon)) = \mathbf{f}} P(\theta \mid \mathcal{E}) \\ &= \sum_{\theta : Str(\theta(\varepsilon)) = \mathbf{f}} \prod_{i=1}^{n} n(\nu_i \mid \mathcal{N}(\varepsilon_i)) r(\rho_i \mid \mathcal{R}(\varepsilon_i)) t(\tau_i \mid \mathcal{T}(\varepsilon_i)), \end{aligned} \tag{1.8}$$

where $\mathcal{N}$, $\mathcal{R}$, and $\mathcal{T}$ reduce ε_i to features significant for insertion, reordering and translation.

In [Yamada and Knight, 2001] they use the model for English-Chinese data. The model requires English data to be in a form of parse trees, and the Chinese

part as a plain text. They use Collins' parser [Collins et al., 1999] to automatically parse the English part. The the EM algorithm is used to obtain parameters of the model.

1.4 Conclusion

We are interested in Czech-English machine translation. One way how to build a translation system, is to collect language-pair-specific rules and put them into the framework that interprets them, another possibility is to use a statistical approach and let the computer learn the rules from data. We understand rules as unavoidable for achieving a good quality of translation, on the other hand, we are aware of the fact that they are expensive, and also that a huge amount of rules is barely manageable. We believe that a good strategy in building machine translation system is to start with a statistical approach, and to fine-tune it using linguistic rules. Hence the design of the model must allow incorporating manual rules.

We have overviewed the major statistical approaches to machine translation above. Although all of them are not specific to any language pair, they are more suitable for languages with a constrained word order. Specific features of Czech, such as inflectiveness, free word order, and pro-drop, have to be handled in a preprocessing steps. What we are missing from the model is some kind of "native" support for these phenomena. The goal of this work is to take part in bridging the gap.

Chapter 2

A Parallel Treebank

The research in the field of dependency-based machine translation assumes experiments with a parallel corpus of structurally annotated sentences. The statistical models we introduce in Chapters 4 and 5 need large amounts of structurally annotated data to learn the transformational patterns, as well as an author of a rule-based translation system wants to observe these phenomena on a representative corpus of examples in order to write the system of rules covering as much of these transformations as possible.

The **Prague Czech-English Dependency Treebank** (PCEDT) is a project of creating a Czech-English syntactically annotated parallel corpus motivated by needs of these experiments.

Since Czech is a language with relatively high degree of word-order freedom, and its sentences contain certain syntactic phenomena, such as discontinuous constituents (non-projective constructions), which cannot be straightforwardly handled using the annotation scheme of the Penn Treebank [Marcus et al., 1993, Linguistic Data Consortium, 1999], based on phrase-structure trees, we decided to adopt for the PCEDT the dependency-based annotation scheme of the Prague Dependency Treebank – PDT [Linguistic Data Consortium, 2001]. The PDT is annotated on three levels: morphological layer (lowest), analytic layer (middle) – surface syntactic annotation, and tectogrammatical layer (highest) – level of linguistic meaning. Dependency trees, representing the sentence structure as concentrated around the verb and its valency, are used for the analytical and tectogrammatical levels, as proposed by Functional Generative Description [Sgall et al., 1986].

In Section 2.1, we describe the process of translating the Penn Treebank into Czech. The Section 2.2 describes the automatic process of parsing of Czech into analytical representation and its automatic conversion into tectogrammatical representation. The following Section 2.3 sketches the general procedure for transforming phrase topology of the Penn Treebank into dependency structure and describes the specific conversions into analytical and tectogrammatical representations. Section 2.5 briefly discusses some of the problems of annotation from the point of view of mutual compatibility of annotation schemes. Section 2.6 gives an overview of additional resources included in the PCEDT.

2.1 English to Czech Translation of the Penn Treebank

There were two possible strategies how to build The Prague Czech-English Dependency Treebank (PCEDT): either the parallel annotation of already existing parallel texts, or the translation and annotation of an existing syntactically annotated corpus. The choice of the Penn Treebank as the source corpus was also pragmatically motivated: firstly, it is a widely recognized and used linguistic resource, and secondly, the translators were native speakers of Czech, capable of high quality translation into their native language.

The translators were asked to translate each English sentence as a single Czech sentence and to avoid unnecessary stylistic changes of translated sentences. The translations are being revised on two levels, linguistic and factual. About half of the Penn Treebank has been translated so far (currently 21,628 sentences), the project aims at translating the whole Wall Street Journal part of the Penn Treebank.

For the purpose of quantitative evaluation methods, such as NIST or BLEU, for measuring performance of translation systems, we selected a test set of 515 sentences and had them retranslated from Czech into English by 4 different translator offices, two of them from the Czech Republic and two of them from the U.S.A.

2.2 Czech Data Processing

2.2.1 Morphological Tagging and Lemmatization

The Czech translations of the Penn Treebank were automatically tokenized and morphologically tagged, each word form was assigned a basic form – *lemma* by Hajič and Hladká [Hajič and Hladká, 1998] tagging tools.

Analytical Parsing

The analytical parsing of Czech runs in two steps: the statistical dependency parser, which creates the structure of a dependency tree, and a classifier assigning analytical functors. We carried out two parallel experiments with two parsers available for Czech, parser I [Hajič et al., 1998] and parser II [Charniak, 1999]. In the second step, we used a module for automatic analytical functor assignment [Žabokrtský et al., 2002].

Conversion into Tectogrammatical Representation

During the tectogrammatical parsing of Czech, the analytical tree structure is converted into the tectogrammatical one. These automatic transformations are based

on linguistic rules [Böhmová, 2001]. Subsequently, tectogrammatical functors are assigned by the C4.5 classifier [Žabokrtský et al., 2002].

2.3 Analytical Representation of English

2.3.1 Automatic Conversion of the Penn Treebank into Analytical Representation

The transformation algorithm from phrase-structure topology into dependency one, similar to transformations described by [Xia and Palmer, 2001], works as follows:

- Terminal nodes of the phrase are converted to nodes of the dependency tree.
- Dependencies between nodes are established recursively: The root node of the dependency tree transformed from the head constituent of a phrase becomes the governing node. The root nodes of the dependency trees transformed from the right and left siblings of the head constituent are attached as the left and right children (dependent nodes) of the governing node, respectively.
- Nodes representing traces are removed and their children are reattached to the parent of the trace.

2.3.2 Preprocessing of the Penn Treebank

Several preprocessing steps preceded the transformation into both analytical and tectogrammatical representations.

Marking of Heads in English

The concept of the head of a phrase is important during the transformation described above. For marking head constituents in each phrase, we used Jason Eisner's scripts ([Eisner, 2001]).

Lemmatization of English

Czech is an inflective language, rich in morphology, therefore lemmatization (assigning base forms) is indispensable in almost any linguistic application. Mostly for reasons of symmetry with Czech data and compatibility with the dependency annotation scheme, the English part was also automatically lemmatized.

We have learned the correspondence between pairs of word form and morphological tag on one side and lemma on the other side from a large corpus of English text [Linguistic Data Consortium, 1995] (365M words, 13M sentences) automatically tagged by MXPOST tagger [Ratnaparkhi, 1996] and lemmatized by the *morpha* tool [Minnen et al., 2001]. The Penn Treebank POS tags were assigned manually, and this information makes an automatic lemmatization procedure more reliable.

Lemmatization procedure makes two attempts to find a lemma:

- first, it tries to find a triple with a matching word form and its (manually assigned) POS;
- if it fails, it makes a second attempt with the word form converted to lowercase.

If it fails in both attempts, then it chooses the given word form as the lemma. For technical reasons, a unique identifier is assigned to each token in this step. Figure 2.1 contains an example of a lemmatized sentence with marked heads.

```
wsj_1700.mrg:5::
(S (NP~-SBJ (DT @the the)
            (@NN @aim aim))
   (@VP (MD @would would)
        (@VP~ (@VB @be be)
              (S~-PRD (NP~-SBJ-1 (@-NONE- @* *))
                      (@VP (TO @to to)
                           (@VP~ (@VB @end end)
                                 (NP~ (@NP (DT @the the)
                                           (NN @guerrilla guerrilla)
                                           (@NN @war war))
                                      (PP (@IN @for for)
                                          (NP~ (@NP (@NN @control control))
                                               (PP (@IN @of of)
                                                   (NP~ (@NPR (@NNP @Cambodia Cambodia)))))))
                                 (PP-MNR (@IN @by by)
                                         (S~-NOM (NP~-SBJ (@-NONE- @*-1 *-1))
                                                 (@VP (@VBG @allowing allow)
                                                      (NP~ (DT @the the)
                                                           (@NPR (NNP @Khmer Khmer)
                                                                 (@NNP @Rouge Rouge)))
                                                      (NP~ (@NP (DT @a a)
                                                                (JJ @small small)
                                                                (@NN @share share))
                                                           (PP (@IN @of of)
                                                               (NP~ (@NN @power power))))))))))))
   (. @. .))
```

Figure 2.1: Example of a lemmatized sentence with marked heads: *"The aim would be to end the guerrilla war for control of Cambodia by allowing the Khmer Rouge a small share of power."*. Terminal nodes consist of a sequence of part-of-speech, word form, lemma, and a unique id. The names of the head constituent names start with @. (In the noun phrase *Khmer Rouge* the word *Rouge* was marked as the head by mistake.)

Unique Identification

For technical reasons, a unique identifier is assigned to each sentence and to each token of the Penn Treebank.

2.3.3 English Analytical Dependency Trees

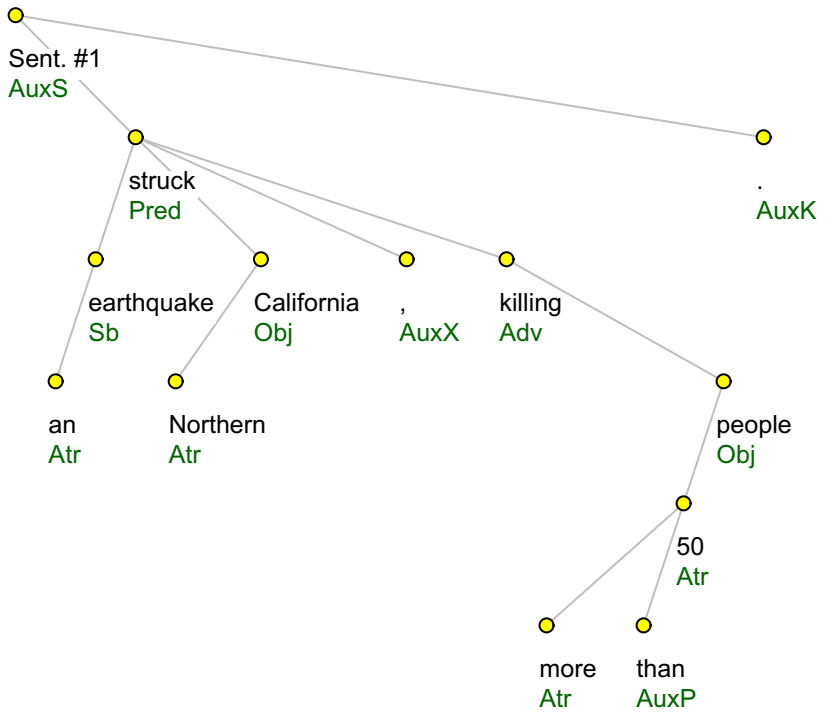


Figure 2.2: Analytical tree for the sentence "*An earthquake struck Northern California, killing more than 50 people.*"

This section describes the automatic process of converting the Penn Treebank annotation into analytical representation.

The **structural transformation** works as described above. Because the handling of coordination in PDT is different from the Penn Treebank annotation style and the output of Jason Eisner's head assigning scripts, in the case of a phrase containing a coordinating conjunction (`CC`), we consider the rightmost `CC` as the head. The treatment of apposition is a more difficult task, since there is no explicit annotation of this phenomenon in the Penn Treebank; constituents of a noun phrase

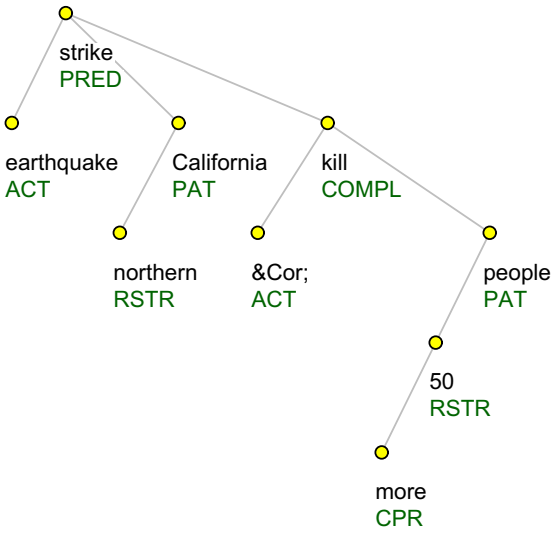


Figure 2.3: Tectogrammatical tree for the sentence "*An earthquake struck Northern California, killing more than 50 people.*"

enclosed in commas or other delimiters (and not containing CC) are considered to be in apposition and the rightmost delimiter becomes the head.

The information from both the phrase tree and the dependency tree is used for the **assignment of analytical functions**:

- The Penn Treebank function tag to analytical function mapping: some function tags of a phrase tree correspond to analytic functions in an analytical tree and can be mapped to them:

  $\texttt{SBJ} \rightarrow \texttt{Sb}$,

  $\{\texttt{DTV}, \texttt{LGS}, \texttt{BNF}, \texttt{TPC}, \texttt{CLR}\} \rightarrow \texttt{Obj}$,

  $\{\texttt{ADV}, \texttt{DIR}, \texttt{EXT}, \texttt{LOC}, \texttt{MNR}, \texttt{PRP}, \texttt{TMP}, \texttt{PUT}\} \rightarrow \texttt{Adv}$.

- Assignment of analytical functions using local context of a node: for assigning analytical functions to the remaining nodes, we use rules looking at the current node, its parent and grandparent, taking into account POS and the phrase marker of the constituent in the original phrase tree headed by the node. For example, the rule

  $$\texttt{mPOS} = \texttt{DT} | \texttt{mAF} = \texttt{Atr}$$

assigns the analytical function `Atr` to every determiner, the rule

$$\texttt{mPOS} = \texttt{MD}|\texttt{pPOS} = \texttt{VB}|\texttt{mAF} = \texttt{AuxV}$$

assigns the function tag `AuxV` to a modal verb headed by a verb, etc. The attribute `mPOS` representing the POS of a node is obligatory for every rule. The rules are examined primarily in the order of the longest prefix of the POS of the given node and secondarily in the order as they are listed in the rule file. The ordering of rules is important, since the first matching rule found assigns the analytical function and the search is finished.

Specifics of the PDT and the Penn Treebank annotation schemes, mainly the markup of coordinations, appositions, and prepositional phrases are handled separately:

- Coordinations and appositions: the analytical function that was originally assigned to the head of a coordination or apposition is propagated to its child nodes by attaching the suffix `_Co` or `_Ap` to them, and the head node gets the analytical function `Coord` or `Apos`, respectively.
- Prepositional phrases: the analytical function originally assigned to the preposition node is propagated to its child and the preposition node is labeled `AuxP`.
- Sentences in the PDT annotation style always contain a root node labeled `AuxS`, which, as the only one in the dependency tree, does not correspond to any terminal of the phrase tree; the root node is inserted above the original root. While in the Penn Treebank the final punctuation is a constituent of the sentence phrase, in the analytical tree it is moved under the technical sentence root node.

Compare the phrase structure and the analytical representation of a sample sentence from the Penn Treebank in Figures 2.4 and 2.2.

2.4 English Tectogrammatical Dependency Trees

2.4.1 Automatic Conversion of Penn Treebank into Tectogrammatical Representation

The transformation of the Penn Treebank phrase trees into tectogrammatical representation consists of a **structural transformation**, and an assignment of a **tectogrammatical functor** and a set of **grammatemes** to each node.

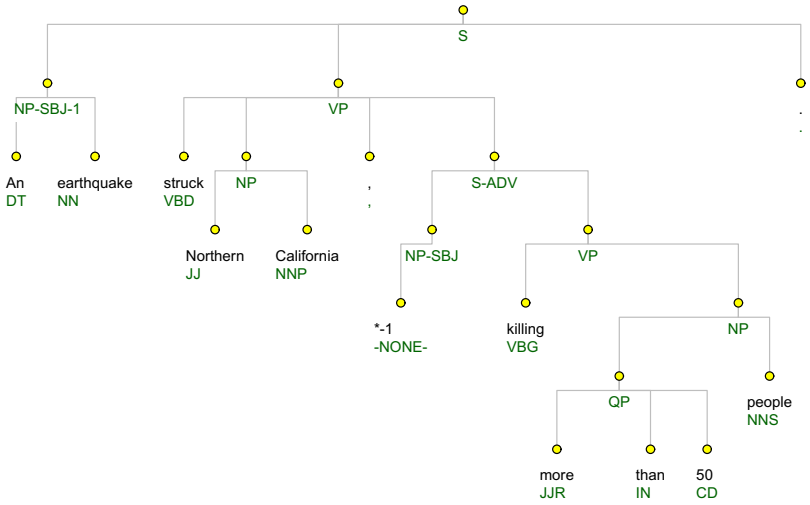


Figure 2.4: Penn Treebank annotation of the sentence "*An earthquake struck Northern California, killing more than 50 people.*"

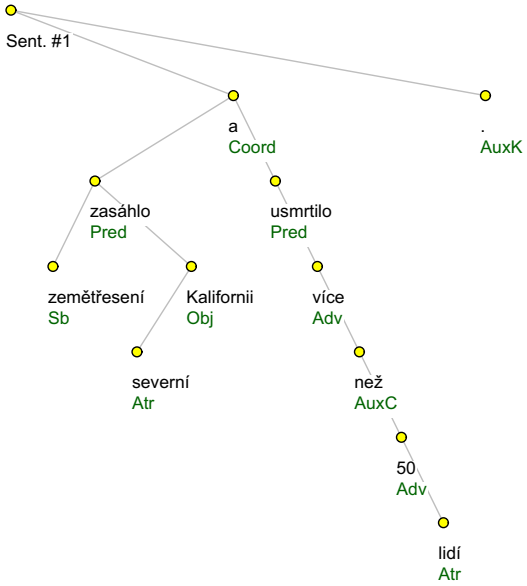


Figure 2.5: Analytical tree for the Czech translation "*Zemětřesení zasáhlo severní Kalifornii a usmrtilo více než 50 lidí.*"

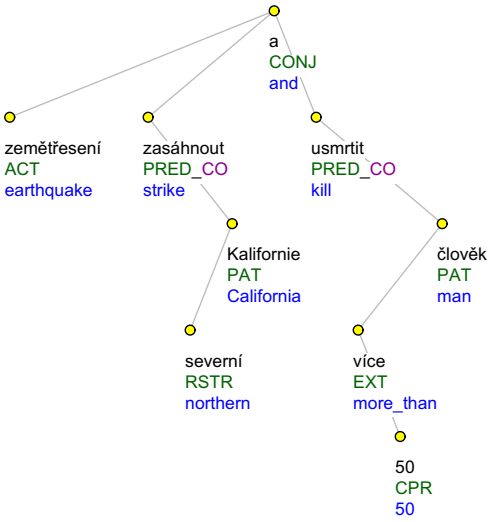


Figure 2.6: Tectogrammatical tree for the Czech translation "*Zemětřesení zasáhlo severní Kalifornii a usmrtilo více než 50 lidí.*"

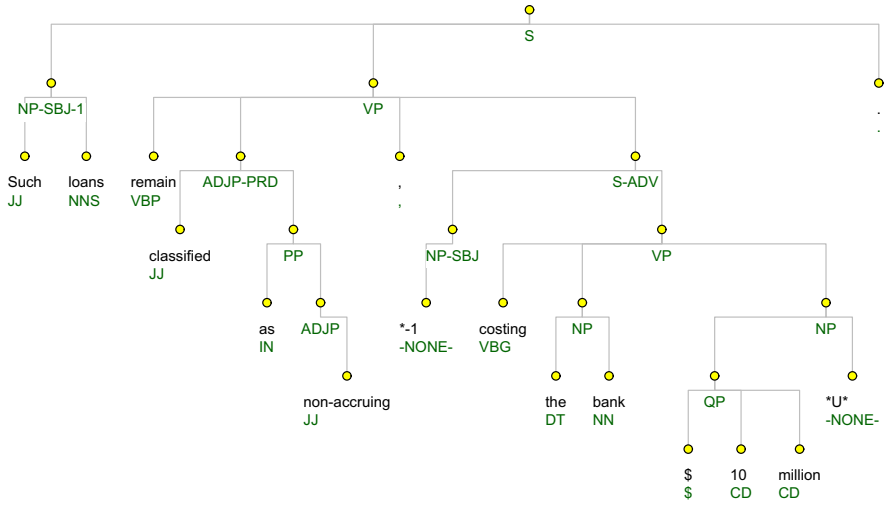


Figure 2.7: Penn Treebank annotation of the sentence "*Such loans remain classified as non-accruing, costing the bank $10 million.*"

At the beginning of the structural transformation, the initial dependency tree is created by a general transformation procedure as described above. However, functional (synsemantic) words, such as prepositions, punctuation marks, determiners, subordinating conjunctions, certain particles, auxiliary and modal verbs are handled differently. They are marked as "hidden" and information about them is stored in special attributes of their governing nodes (if they were to head a phrase, the head of the other constituent became the governing node in the dependency tree).

The well-formedness of a tectogrammatical tree structure requires the valency frames to be complete: apart from nodes that are realized on surface, there are several types of "restored" nodes representing the non-realized members of valency frames (cf. pro-drop property of Czech and verbal condensations using gerunds and infinitives both in Czech and English). For a partial reconstruction of such nodes, we can use traces, which allow us to establish coreferential links, or restore general participants in the valency frames.

For the assignment of tectogrammatical functors, we can use rules taking into consideration POS tags (e.g. PRP $\rightarrow$ APP), function tags (JJ $\rightarrow$ RSTR, JJR $\rightarrow$ CPR, etc.) and lemma ("not" $\rightarrow$ RHEM, "both" $\rightarrow$ RSTR).

Grammateme Assignment – morphological grammatemes (e.g. tense, degree of comparison) are assigned to each node of the tectogrammatical tree. The assignment of the morphological attributes is based on the Penn Treebank tags and reflects basic morphological properties of the language. At the moment, there are no automatic tools for the assignment of syntactic grammatemes, which are designed to capture detailed information about deep syntactic structure.

The whole procedure is described in detail in [Kučerová and Žabokrtský, 2002].

In order to gain a "gold standard" annotation, 1,257 sentences have been annotated manually (the 515 sentences from the test set are among them). These data are assigned morphological grammatemes (the full set of values) and syntactic grammatemes, and the nodes are reordered according to topic-focus articulation (information structure).

The quality of the automatic transformation procedure described above, based on comparison with manually annotated trees, is about 6% of wrongly aimed dependencies and 18% of wrongly assigned functors.

See Figure 2.3 for the manually annotated tectogrammatical representation of the sample sentence.

2.5 Problems of Dependency Annotation of English

The manual annotation of 1,257 English sentences on tectogrammatical level was, to our knowledge, the first attempt of its kind, and was based especially on the instructions for tectogrammatical annotation of Czech. During the process of annotation, we have experienced both phenomena that do not occur in Czech and those phenomena, whose counterparts in Czech occur rarely, and therefore the guidelines for tectogrammatical annotation of Czech do not handle them thoroughly. To mention just a few, among the former belongs the annotation of articles, certain aspects of the system of verbal tenses, and phrasal verbs. A specimen of a roughly corresponding phenomenon occurring both in Czech and English is the gerund. It is a very common means of condensation in English, but its counterpart in Czech (usually called transgressive) has fallen out of use and is nowadays considered rather obsolete.

The guidelines for Czech require the transgressive to be annotated with the functor `COMPL`. The reason why it is highly problematic to apply them straightforwardly also to the annotation of English, is that the English gerund has a much wider range of functions than the Czech transgressive. The gerund can be seen as a means of condensing subordinated clauses with in principle adverbial meaning (as it is analyzed in the phrase-structure annotation of the Penn Treebank). Since the range of functors with adverbial meaning is much more fine-grained, we deem it inappropriate to mark the gerund clauses in such a simple way on the tectogrammatical level.

From the point of view of machine translation, the gerund constructions pose considerable difficulties because of the many syntactic constructions suitable as their translations corresponding to their varied syntactic functions.

We present two examples illustrating the issues mentioned above. Each example consists of three figures, the first one presenting the Penn Treebank annotation of a (in the second case simplified) sentence from the Penn Treebank, the second one giving its tentative tectogrammatical representation (according to the guidelines for Czech applied to English), and the third one containing the tectogrammatical representation of its translation into Czech (cf. Figures 2.4, 2.3, 2.6, and Figures 2.7, 2.8, 2.9). Note that in neither of the two examples the Czech transgressive is used as the translation of the English gerund; a coordination structure is used instead.

On the other hand, we have also experienced phenomena in English whose Penn Treebank style of annotation is insufficient for a successful conversion into dependency representation.

For example, the usage of constructions with nominal premodification is very frequent in English, and the annotation of such noun phrases is often flat, grouping together several constituents without reflecting finer syntactic and semantic rela-

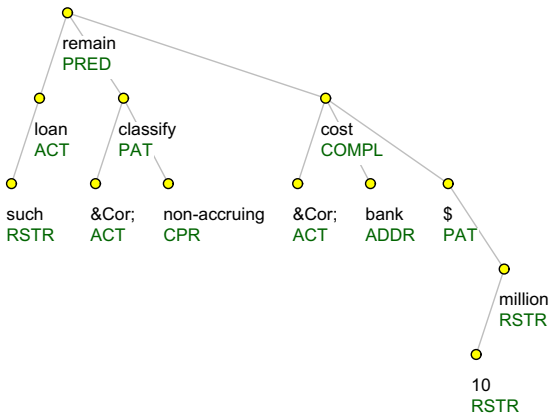


Figure 2.8: Tectogrammatical tree for the sentence "*Such loans remain classified as non-accruing, costing the bank $10 million.*"

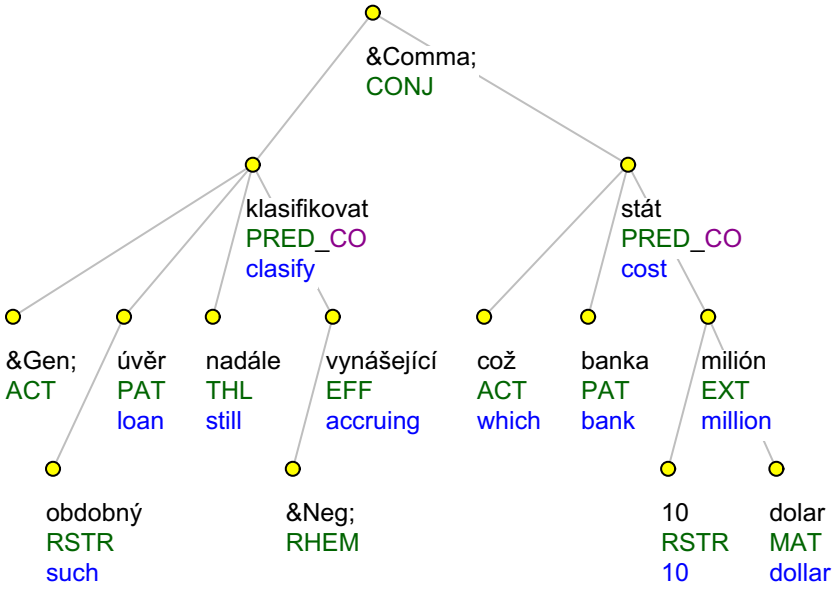


Figure 2.9: Tectogrammatical tree for the Czech translation "*Obdobné úvěry jsou nadále klasifikovány jako nevynášející, což banku stálo 10 millionů dolarů.*"

tions among them. See Figure 2.10 for an example of such a noun phrase. In fact, the possible syntactic and especially semantic relations between the members of the noun phrase can be highly ambiguous, but when translating such a noun phrase into Czech, we usually are not able to preserve the ambiguity and are forced to resolve it by choosing the realization of one of the readings (cf. Figure 2.11).

Sometimes we even may be forced to insert new words explicitly expressing the semantic relations within the nominal group. An example of an English noun phrase and the tectogrammatical representation of its Czech translation with an inserted word "podnikající" ('operating') can be found in Figures 2.12 and 2.13.

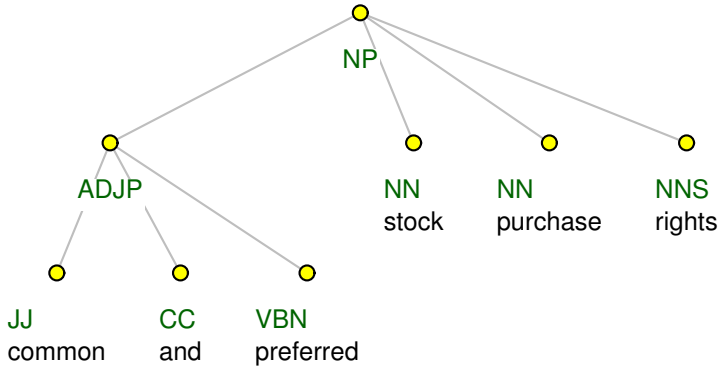


Figure 2.10: Penn Treebank annotation of the noun phrase "*common and preferred stock purchase rights*".

2.6 Other Resources Included in PCEDT

2.6.1 Reader's Digest Parallel Corpus

Reader's Digest parallel corpus contains raw text in 53,000 aligned segments in 450 articles from the Reader's Digest, years 1993–1996. The Czech part is a free translation of the English version. The final selection of data has been done manually, excluding articles whose translations significantly differ (in length, culture-specific facts, etc.). Parallel segments on sentential level have been aligned by Dan Melamed's aligning tool [Melamed, 1996]. The topology is 1–1 (81%), 0–1 or 1–0 (2%), 1–2 or 2–1 (15%), 2–2 (1%), and others (1%).

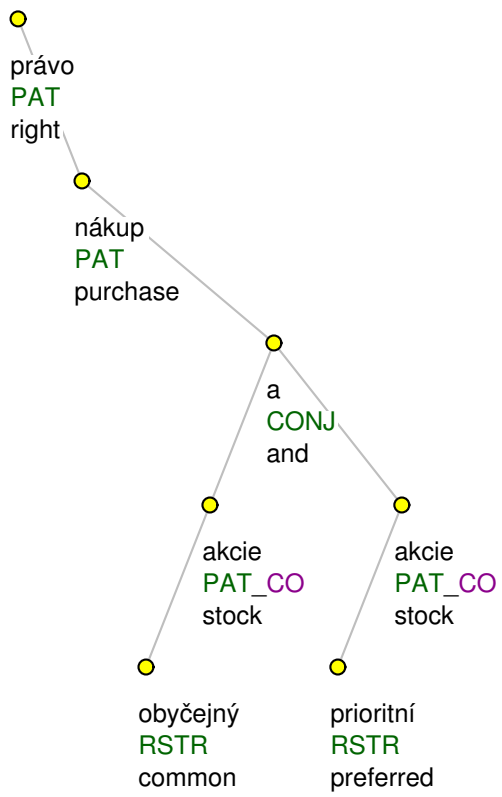


Figure 2.11: Tectogrammatical tree for the Czech translation "*právo na nákup obyčejných a prioritních akcií*".

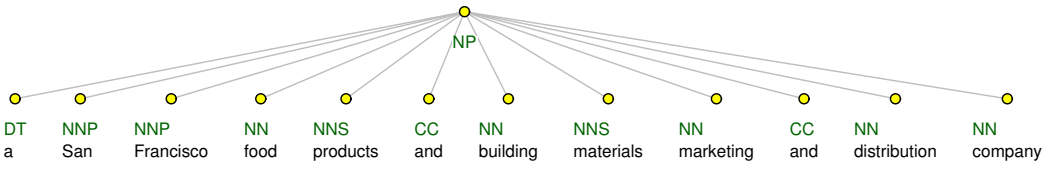


Figure 2.12: Penn Treebank annotation of the noun phrase "*a San Francisco food products and building materials marketing and distribution company*".

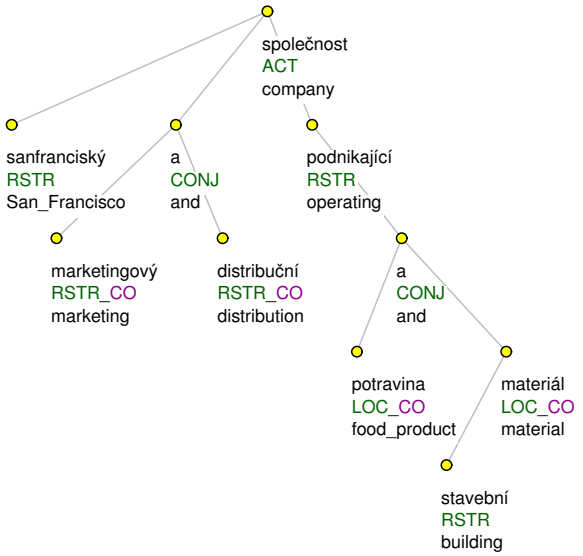


Figure 2.13: Tectogrammatical tree for the Czech translation "*sanfranciská marketingová a distribuční společnost podnikající v potravinách a stavebních materiálech*".

2.6.2 Dictionaries

The PCEDT comprises also a translation dictionary compiled from three different Czech-English manual dictionaries: two of them were downloaded form the Web and one was extracted from Czech and English EuroWordNets. Entry-translation pairs were filtered and weighed taking into account the reliability of the source dictionary, the frequencies of the translations in Czech and English monolingual corpora, and the correspondence of the Czech and English POS tags. Furthermore, by training GIZA++ [Och and Ney, 2003] translation model on the training part of the PCEDT extended by the manual dictionaries, we obtained a probabilistic Czech-English dictionary, more sensitive to the domain of financial news specific for the Wall Street Journal.

The resulting Czech-English probabilistic dictionary contains 46,150 entry-translation pairs in its lemmatized version and 496,673 pairs of word forms in the version where for each entry-translation pair all the corresponding word form pairs have been generated.

2.6.3 Tools

The following tools are a part of the PCEDT distribution:

- **SMT Quick Run** is a package of scripts and instructions for building statistical machine translation system from the PCEDT or any other parallel corpus. The system uses models GIZA++ and ISI ReWrite decoder [Germann et al., 2001].

- **TrEd** is a graphical editor and viewer of tree structures. Its modular architecture allows easy handling of diverse annotation schemes, it has been used as the principal annotation environment for the PDT and PCEDT.

- **Netgraph** is a multi-platform client-server application for browsing, querying and viewing analytical and tectogrammatical dependency trees, either over the Internet or locally.

2.7 Conclusion

Building a large-scale parallel treebank is a demanding challenge. We have created a parallel corpus for a pair of languages with a relatively different typology, Czech and English, and made an attempt to bridge between two linguistic theories commonly used for their description.

We are convinced that the PCEDT will be useful for further experiments in Czech-English machine translation. A certain disproportion between the English part converted from a manual annotation and the Czech part automatically parsed from plain text corresponds to the real situation in Czech-English machine translation, where modules for transfer and generation have to adapt to errors caused by automatic analysis of the input language. Several input options for Czech (plain text, analytical and tectogrammatical representations–both automatic and manual) and a test set for quantitative evaluation can be used in various experimental settings, allowing to identify insufficiencies in analysis, transfer, and generation.

Chapter 3

Rule-based machine translation system using tectogrammatical representation

In this chapter, we describe an attempt to develop a full machine translation based on tectogrammatical dependency trees. The system is designed to translate a broad domain of Wall Street Journal newspaper texts from Czech to English. The approach combines statistical methods for analyzing the source language and producing its tectogrammatical representation, and a set of rules for lexical transfer and generation into English. Results of the system were published in [Čmejrek et al., 2003a].

The system works as follows. The Czech sentence is analyzed into its tectogrammatical representation using the same sequence of steps as already described in Section 2.2. The lexical transfer step using a translation dictionary prepared as listed in Section 3.1 then transforms the Czech tectogrammatical trees into so-called *Czenglish* trees as explained in Section 3.2. Section 3.3 describes the rule-based generation of the English output. An example of resulting translations is detailed in Section 3.4, and the BLEU evaluation of these results can be found in Section 3.5.

3.1 Czech-English Word-to-Word Translation Dictionaries

When constructing the translation dictionary for the MT system, we have followed two main criteria: first, the dictionary should cover as much vocabulary as possible, and second, possible translation alternatives have to be organized in such a way that translations specific to a given domain of text have higher priority than other translations. We have used several sources of dictionaries that were available on the Internet, merged them and compiled a translation dictionary sensitive to the domain of Wall Street Journal.

3.1.1 Manual Dictionary Sources

There were three different sources of Czech-English manual dictionaries available, two of them were downloaded from the Web (WinGED, GNU/FDL), and

one was extracted from the Czech and English EuroWordNet. See dictionary parameters in Table 3.1.

3.1.2 Dictionary Filtering

For a subsequent use of these dictionaries in a simple Czech-English transfer of tectogrammatical trees (see Section 3.2), a relatively large number of possible translations for each entry[1] had to be filtered out. The aim of the filtering is to exclude synonyms from the translation list, i.e. to choose a single representative per meaning.

First, all dictionaries are converted into a unified XML format and merged preserving information about the source dictionary. Figure 3.1 contains an example of the format.

This merged dictionary, consisting of entry/translation pairs (Czech entries and English translations in our case), is enriched with the following information:

- The word occurence frequency, as obtained from a large English monolingual corpus [Linguistic Data Consortium, 1995], is added to all translations of each entry. (See description of the corpus in Section 2.3.2).
- The Czech POS tag and stem are added to each entry using the Czech morphological analyzer [Hajič and Hladká, 1998].
- The English POS tag is added to each translation. If there is more than one English POS tag obtained from the English morphological analyzer [Ratnaparkhi, 1996], the English POS tag is "disambiguated" according to the Czech POS in the corresponding entry/translation pair.

Then, the selection of the relevant translations for each entry is done based on the sum of the weights of the source dictionaries (see dictionary weights in Table 3.1), the frequencies from English monolingual corpora, and the correspondence of the Czech and English POS tags.

3.1.3 Scoring Translations Using GIZA++

To make the dictionary more sensitive to a given domain, which is financial news in our case, we used a parallel corpus consisting of the training part of the English-Czech WSJ parallel corpus, extended by the parallel corpus of entry/translation pairs from the manual dictionary. We then created a probabilistic Czech-English dictionary by running a GIZA++ training (translation models 1–4, see [Och and

[1] For example, the WinGED dictionary has 2.44 translations per entry in average; excluding 1-1 entry/translation pairs, this number jumps to 4.51 translations/entry.

```
<TransDictionary src_lang="Cz" tgt_lang="En">

    <Entry literal="vyber">
        <WordSet>
            <Word index="1">
                <Form>vyber</Form>
                <Lemma>vyber</Lemma>
                <Tag>N</Tag>
            </Word>
       </WordSet>

       <Translations>
           <Translation literal="choice">
               <WordSet>
                   <Word index="1">
                       <Form>choice</Form>
                       <Tag>N</Tag>
                   </Word>
               </WordSet>
               <Sources>
                   <Source src="EWN"/>
                   <Source src="GNU/FDL"/>
                   <Source src="GIZA++"/>
               </Sources>
               <Counts>
                   <Count src="WSJtrn">1189</Count>
               </Counts>
               <Probs>
                   <Prob src="GIZA++">0.404815</Prob>
               </Probs>
               <Selections>
                   <Selection src="DictSelect"/>
                   <Selection src="GIZA++Select"/>
                   <Selection src="FinalSelect"/>
               </Selections>
           </Translation>
           <Translation literal="selection">
               <WordSet>
                   <Word index="1">
                       <Form>selection</Form>
                       <Tag>N</Tag>
                   </Word>
               </WordSet>
               <Sources>
                           ...
```

Figure 3.1: Sample of the XML format of merged Czech-English manual dictionaries.

dictionary	*#entries*	*#transl*	*weight*
EuroWordNet	12,052	48,525	**3**
GNU/FDL	12,428	17,462	**2.5**
WinGED	16,296	39,769	**2**
merged	33,028	87,955	—

Table 3.1: Dictionary parameters and weights

Ney, 2000]) on this corpus. As a result, the entry/translation pairs seen in the parallel corpus of WSJ become more probable. For entry/translation pairs not seen in the parallel text, the probability distribution among translations is uniform. The translation is "GIZA++ selected" if its probability is higher than a threshold, which is in our case set to 0.10.

The final selection contains translations selected by both the dictionary and GIZA++ selectors. In addition, translations not covered by the original dictionary can be included into the final selection, if they were both newly discovered in the parallel corpus by GIZA++ training, and their probability is significant (higher than the most probable translation so far, in our case).

The translations from this final selection are then used in the transfer. See a sample of the dictionary in Figure 3.2.

3.2 Lexical Transfer

In lexical transfer, tectogrammatical trees automatically created from Czech input text are transferred into "English" tectogrammatical trees. The transfer procedure itself is a lexical replacement of the tectogrammatical base form attribute of autosemantic nodes (*trlemma*) by its English equivalent found in the Czech-English probabilistic dictionary.

Because of multiple translation possibilities, the output structure is a forest of "Czenglish" tectogrammatical trees represented in a packed-tree format [Langkilde, 2000]. Figure 3.3 contains an example of the "Czenglish" tectogrammatical packed-tree.

For practical reasons such as time efficiency, the first experiments used just a simplified implementation of the transfer, taking into account only the most probable translation. Also, 1–2 translations were handled as 1–1, i.e. two words for one trlemma attribute. Later experiments developed all hypotheses stored in the packed tree and rescored them using an n-gram language model.

You may see an example of a Czech tectogrammatical tree after the lexical transfer step (Figure 3.5), and compare it to the original English sentence in Fig-

```
<e>zesílit<t>V

   [FSG]<tr>increase<trt>V<prob>0.327524
   [FSG]<tr>reinforce<trt>V<prob>0.280199
   [FSG]<tr>amplify<trt>V<prob>0.280198
   [G]<tr>re-enforce<trt>V<prob>0.0560397
   [G]<tr>reenforce<trt>V<prob>0.0560397

<e>výběr<t>N

   [FSG]<tr>choice<trt>N<prob>0.404815
   [FSG]<tr>selection<trt>N<prob>0.328721
   [G]<tr>option<trt>N<prob>0.0579416
   [G]<tr>digest<trt>N<prob>0.0547869
   [G]<tr>compilation<trt>N<prob>0.0547869
   []<tr>alternative<trt>N<prob>0.0519888
   []<tr>sample<trt>N<prob>0.0469601

<e>selekce<t>N

   [FSG]<tr>selection<trt>N<prob>0.542169
   [FSG]<tr>choice<trt>N<prob>0.457831
```

Figure 3.2: Sample of the Czech-English probabilistic dictionary used for the transfer. [S]: dictionary weight selection, [G]: GIZA++ selection, [F]: final selection.

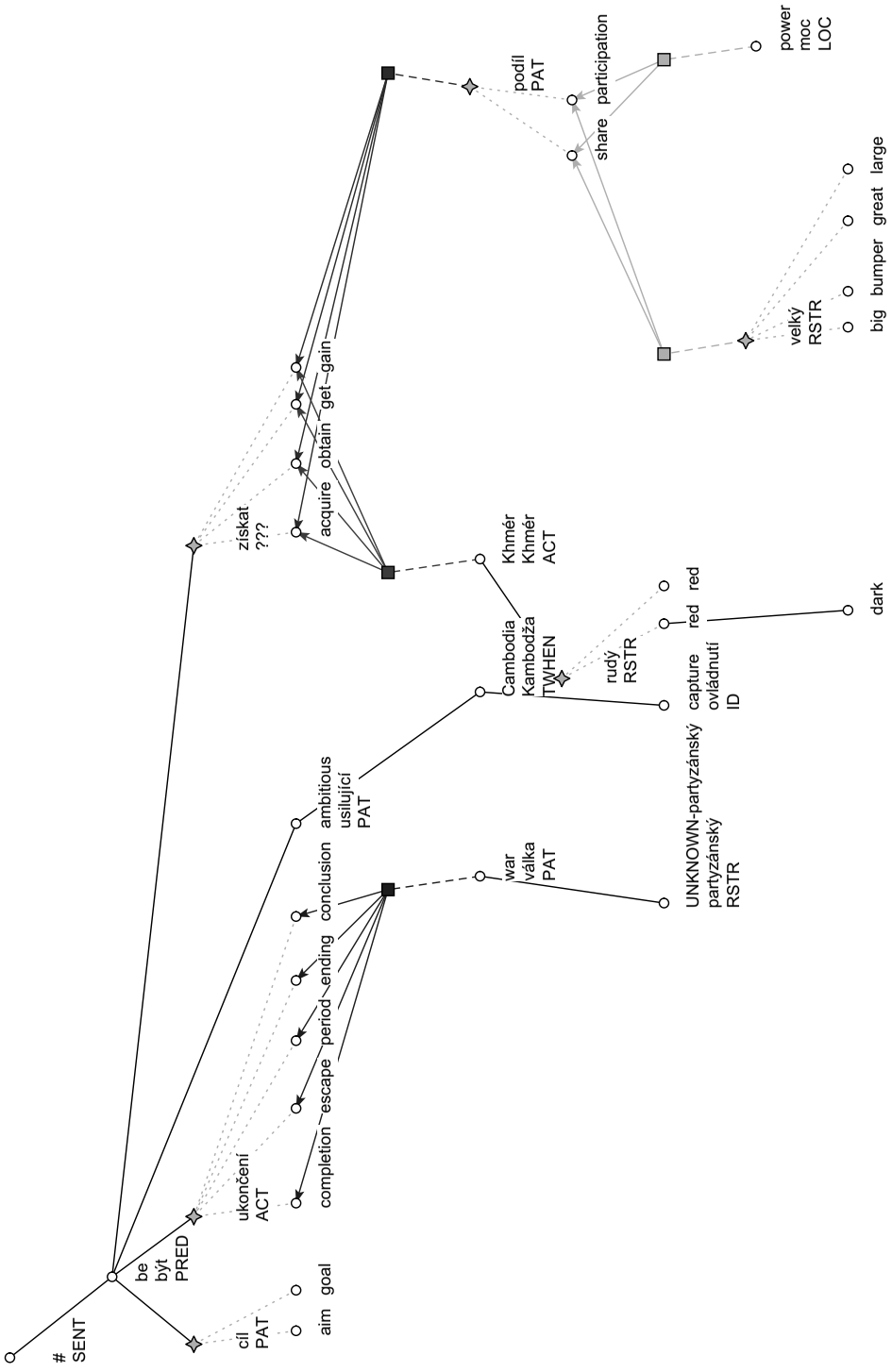


Figure 3.3: Example of a packed tree representation of a forest of Czenglish tectogrammatical trees resulting from the sentence: "*Cílem by bylo ukončení partyzánské války usilující o ovládnutí Kambodže, přičemž by Rudí Khmérové získali nevelký podíl na moci.*"

Algorithm 3.1 Translation equivalent replacement algorithm (for 1-1 and 1-2 entry-translation mapping).

1. **for** each Czech tectogrammatical tree (TGTree) **do**
2. Start at the root
3. In the dictionary, find translation equivalents for ”trlemma” of this node
4. **if** there is only one translation
5. Add the appropriate TN-tags to this node, continue with step 14
6. **else**
7. Change the current node into OR_node
8. **for** each child of the current node
9. Create a new ID_node,
10. Set the parent of the child to this ID_node
11. Create new WORD_node for each translation variant,
set parents of the new nodes to the OR_node.
If there is a multi-word translation, choose the head of the translation as the WORD_node and create nodes for other dependent nodes.
12. **for** each ID_node created in step 8
13. set multiple parents to all WORD_nodes created in step 11
14. Backtrack to the next node in TGTree and continue with step 3

ure 3.4.

3.3 Rule-based Text Generation from English Tectogrammatical Representation

When generating English text from the tectogrammatical representation, two kinds of operations (although often interfering) have to be performed: lexical insertions and transformations modifying word order.

Since only autosemantic (lexical) words are represented in the tectogrammatical structure of the sentence, a successful generation of English plain-text output needs the insertion of synsemantic (functional) words (such as prepositions, auxiliary verbs, and articles). Unlike in Czech, where different semantic roles are expressed by different cases, English uses both prepositions and word order to convey this information.

In our implementation, the generation process consists of the following six consecutive groups of generation tasks:

1. determining contextual boundness,
2. reordering of constituents,

Original: Kaufman & Broad, a home building company, declined to identify the institutional investors.
Czech: Kaufman & Broad, firma specializující se na bytovou výstavbu, odmítla institucionální investory jmenovat.
R1: Kaufman & Broad, a company specializing in housing development, refused to give the names of their corporate investors.
R2: Kaufman & Broad, a firm specializing in apartment building, refused to list institutional investors.
R3: Kaufman & Broad, a firm specializing in housing construction, refused to name the institutional investors.
R4: Residential construction company Kaufman & Broad refused to name the institutional investors.

Figure 3.4: A sample English sentence from WSJ, its Czech translation, and four reference retranslations.

3. generating verb forms,
4. inserting prepositions and articles,
5. generating morphological forms,
6. LM rescoring of multiple hypotheses.

In each of these task, the whole tectogrammatical tree is traversed while task rules are applied. Considering the nature of the selected data, i.e. WSJ financial news, our system is limited to declarative sentences only.

Determination of Contextual Boundness

Since neither the automatically created nor the manually annotated tectogrammatical trees capture any topic–focus articulation (information structure), we use the fact that Czech is a language with a relatively high degree of word order freedom and uses mainly the left to right ordering to express the information structure. In written text, given information (contextually bound) tends to be placed at the beginning of the sentence, while new information (contextually non-bound) is expressed towards the end of the sentence. The degree of communicative dynamism increases from left to right, and the boundary between the contextually bound nodes on the left-hand side and the contextually non-bound nodes on the right-hand side is the verb. We consider information structure to be recursive in the dependency tree, and use it both for the reordering of constituents in the English

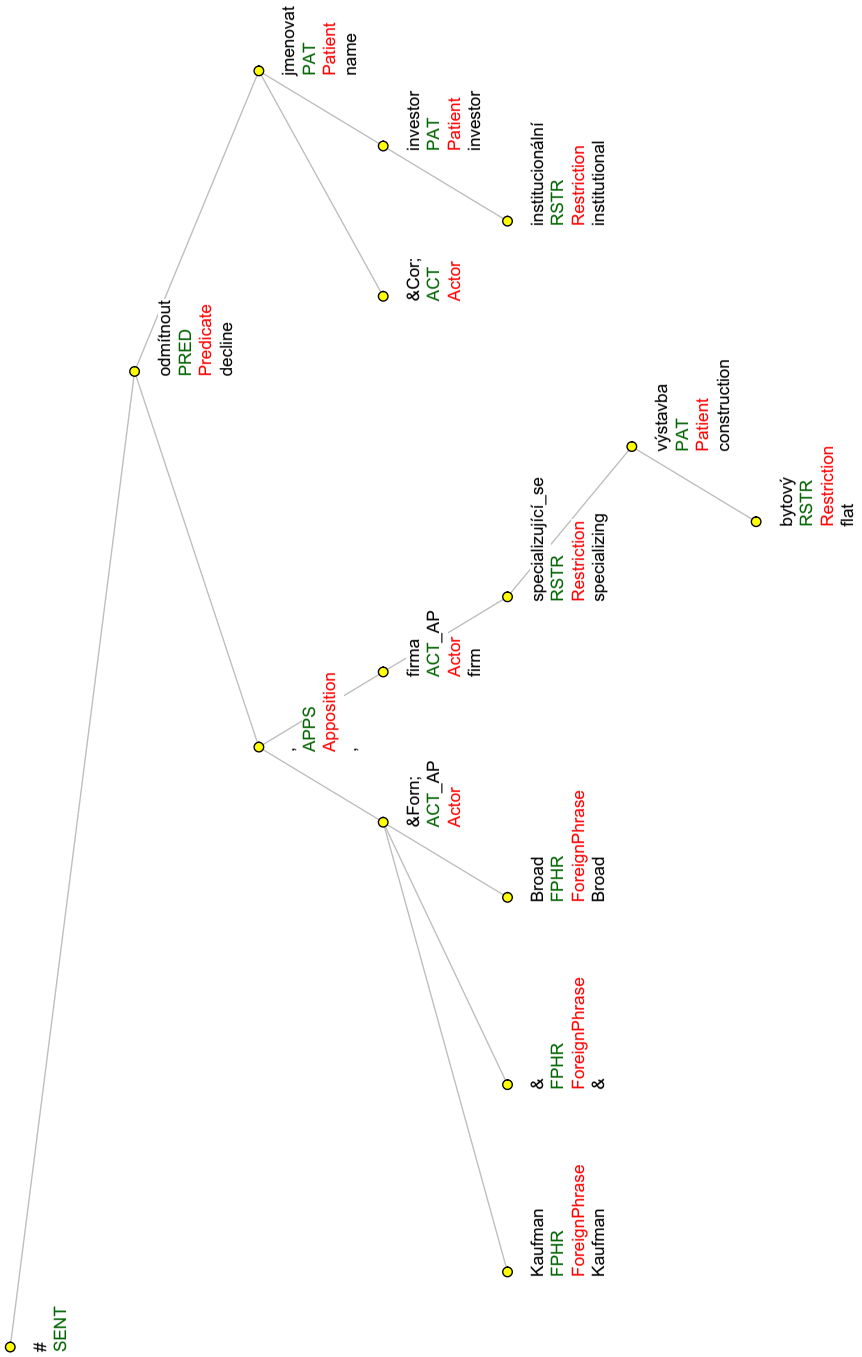


Figure 3.5: An example of a manually annotated Czech tectogrammatical tree with Czech lemmas, tectogrammatical functors, their glosses, and automatic word-to-word translations to English.

counterpart of the Czech sentence, and for determining the definiteness of noun phrases in English.

Reordering of Constituents

Unlike Czech, English is a language with a quite rigid SVO word order, therefore, according to the sentence modality, verb complements and adjuncts have to be rearranged to conform to the constraints of English grammar. In the basic case of a simple declarative sentence, we first place the contextually bound adjuncts, then the subject, the verb, the verb complements (such as direct and indirect objects), and finally contextually non-bound adjuncts, always preserving the relative order of constituents in each group. The functors in a tectogrammatical tree denote the semantic role of nodes. So we can use the contextual boundness/non-boundness of ACTor (deep subject), PATient (deep object), or ADDRessee, and realize the most contextually bound node as the surface subject.

Generation of Verb Forms

According to the semantic role selected as the subject of the verb, the active or passive voice of the verb is chosen. Categories such as tense and mood are taken over from the information stored in the Czech tectogrammatical node. The person is determined by agreement with the subject. Auxiliary verbs needed to create a complex verb form are inserted as separate children nodes of the lexical verb.

Insertion of Prepositions and Articles

Establishing the correspondence between tectogrammatical functors and auxiliary words is a complex task. In some cases, there is one predominant surface realization of the functor, but, unfortunately, in other cases, there are several possible surface realizations, none of them significantly dominant (mostly in cases of spatial and temporal adjuncts). For deciding on the appropriate surface realization of a preposition, both the original Czech preposition and the English lexical word being generated should be taken into account.

The task of generating articles in English is non-trivial and challenging due to the absence of articles in Czech. The first hint about what article should be used is the contextual boundness/non-boundness of a noun phrase. The definite article is inserted when the noun phrase is either contextually bound, postmodified, or is premodified by a superlative adjective or an ordinal numeral. Otherwise, the indefinite article is used.

An article may be prevented from being inserted altogether in case of uncountable or proper nouns, or when the noun phrase is predetermined by some

MT system	*BLEU – devtest*	*BLEU – evaltest*
DBMT with parser I	0.1857	0.1634
DBMT with parser II	0.1916	0.1705
DBMT on manually annotated trees	0.1974	0.1704
DBMT with parser II, LM rescoring	0.1921	0.1705
DBMT on manually annotated trees, LM rescoring	0.1968	0.1731
GIZA++ & ReWrite – plain text	0.0971	0.0590
GIZA++ & ReWrite – lemmatized	0.2222	0.2017
MAGENTA WS'02	0.0640	0.0420
Avg. BLEU score of human retranslations	—	0.5560

Table 3.2: BLEU score of different MT systems

other means (such as possessive or demonstrative pronouns).

Generating Morphological Forms

When generating the surface word form, we are searching through the table of triples [word form, morphological tag, lemma] (see Section 2.3.2) for the word form corresponding to the given lemma and morphological tag. Should we fail to find it, we generate the form using simple rules, e.g. attaching suffix for plural, etc. Also, the appropriate form of the indefinite article is selected according to the immediately following word.

LM Rescoring of Multiple Hypotheses

We also built a system that develops multiple translation hypotheses at the same time, and rescores them by a language model. We have experimented with multiple variants of insertions of preposition and articles, but did not allow variants of lexical nodes. The language model used for the rescoring was the trigram LM with Good-Turing discounting and Katz back-off for smoothing. These were trained with the SRILM language modeling toolkit [Stolcke, 2002] on the 52 million words selected from the monolingual North American News Text Corpus of the Wall Street Journal from years 1995 and 1996 [Linguistic Data Consortium, 1995].[2]

[2]The Penn Treebank data are from other years.

3.4 An Example

Figure 3.6 illustrates the whole process of translating a sample Czech sentence to English, starting from its manually annotated tectogrammatical representation (Figure 3.5). The first line contains lemmas of the autosemantic words of the sample sentence from Figure 3.4. The next line, labeled 0, shows their word-to-word translations. The remaining lines correspond to the generation steps described in Section 3.3.

The order of nodes is used to determine their contextual boundness (line 1, contextually non-bound nodes are in italics). In line 2, the constituents are reordered according to contextual boundness and their tectogrammatical functors. The form of the complex verb is handled in step 3. On the next time, prepositions and articles are inserted. However, not every functor's realization can be reconstructed easily, as can be seen in the case of the missing preposition "in". It is also hard to decide whether a particular word was used in an uncountable sense (see the wrongly inserted indefinite article). The last line contains the final morphological realization of the sentence.

3.5 Evaluation of Results

We evaluated our translations with IBM's BLEU evaluation metric [Papineni et al., 2001], using the same evaluation method and reference retranslations that were used for evaluation at the HLT Workshop 2002 at CLSP [Hajič et al., 2002]. We used four reference retranslations of 490 sentences selected from the WSJ sections 22, 23, and 24, which were themselves used as the fifth reference. The evaluation method used is to hold out each reference in turn and evaluate it against the remaining four, then averaging the five BLEU scores.

Table 3.2 shows final results of our system compared with GIZA++ and MAGENTA's results.

The DBMT with parser I and parser II experiments represent a fully automated translation, while the DBMT experiment on manually annotated trees uses Czech tectogrammatical trees prepared by human annotators.

We can see that the experiments with rescoring using a language model did not bring any convincing improvement. In the case of manually annotated trees, the BLEU score was even worse. Since the lexical variability was allowed only on positions of prepositions and articles, it shows that the lexical information about the original Czech preposition is very important for a successful generation of the English preposition. Nevertheless, experiments with LM rescoring of multiple hypotheses should be evaluated also for multiple variants of semantic words.

For the purposes of comparison, the GIZA++ statistical machine translation

Cz	Kaufman & Broad		firma	specializující-se		bytový	výstavba	odmítnout		instit.	investor	jmenovat
0.	Kaufman & Broad		firm	specializing		flat	construction	decline		instit.	investor	name
1.	Kaufman & Broad		firm	specializing		flat	*construction*	decline		instit.	investor	*name*
2.	Kaufman & Broad		firm	specializing		flat	construction	decline		name		instit. investor
3.	Kaufman & Broad		firm	specializing		flat	construction	decline	to	name		instit. investor
4.	Kaufman & Broad	DEF	firm	specializing	INDEF	flat	construction	decline	to	name	DEF	instit. investor
5.	Kaufman & Broad	the	firm	specializing	a	flat	construction	declined	to	name	the	instit. investors

Figure 3.6: An illustration of the generation process for the resulting English sentence: "*Kaufman & Broad, the firm specializing a flat construction declined to name the institutional investors*". The numbering of steps corresponds to the sequence of generative tasks from the beginning of the Section 3.3.

toolkit with the ReWrite decoder was customized to translate from Czech to English, and two experiments with different configurations were performed. The first one takes Czech plain text as input, the second one translates from lemmatized Czech. In addition, the word-to-word dictionary described in Section 2.6.2 was added to the training data (every entry-translation pair as one sentence pair). The language model was trained on a large monolingual corpus from Wall Street Journal containing about 52M words. This corpus was selected from the corpus mentioned in Section 2.3.2.

All systems were evaluated against the same set of references.

Both our experiments show a considerable improvement over MAGENTA's performance, they also score better than GIZA++/ReWrite trained on word forms. We were still outperformed by GIZA++/ReWrite trained on lemmas and making use of a large language model.

3.6 Conclusion

This chapter describes a complete translation system from Czech plain text to English plain text. It integrates the latest results in analytical and tectogrammatical parsing of Czech, experiments with existing word-to-word dictionaries combined with those automatically obtained from a parallel corpus, lexical transfer, and simple rule-based generation from the tectogrammatical representation.

In spite of certain known shortcomings of state-of-the-art parsers of Czech, we are convinced that the most significant improvement of our system can be achieved by further refining and broadening the coverage of structural transformations and lexical insertions. We consider allowing multiple translation possibilities even for lexical words and using additional sources of information relevant for surface realization of tectogrammatical functors.

Chapter 4

Tree-to-Tree Transducer

The idea of a *Synchronous Tree Substitution Grammar* was first sketched in [Hajič et al., 2002] and [Eisner, 2003]. A rule of such a grammar has the form of a pair of so-called *little trees* with *aligned frontier nodes* that constrain both the positions, where other little trees can attach, and their type. The tree-to-tree transformation process covers the source tree by the source little trees from the rule-set, the output tree is then being constructed from the corresponding target little trees.

This chapter defines the theory of such synchronous tree substitution grammars and elaborates on the mathematical details that were not published yet. It starts with the monolingual case, then extends it into the synchronous case. We also present algorithms for training the models on a corpus of parallel trees, and the decoding algorithm necessary for producing translation.

We are aware of that the new theory is quite complicated. In order to help the reader to understand the new concepts, we start with the Section 4.1 giving an informal overview of the theory, jumping into the middle of the problem and trying to explain it using a "common sense". We hope that this figurative explanation makes the reading of the following pages easier.

4.1 Informal Motivation

Figure 4.1 contains an example of a tectogrammatical tree for a sample Czech sentence and an analytical tree for its English equivalent. The Figure 4.2 then contains both trees split into chunks. The chunk is usually formed by two *little trees* with filled (black) and empty (white) nodes.

The filled nodes are called *internal*, the empty nodes are called *frontier* nodes. The frontier nodes are connected by bows. The bows can be called *alignment*, *matching*, or *mapping*, and it always means the same thing.

The chunk of both trees with aligned frontier nodes is a *rule* of the *Synchronous Tree Substitution Grammar*.

The meaning of the first rule in Figure 4.2 is that the Czech *informovat nesprávně* is translated as the English *were misinformed*. The alignment between the frontier node PAT of the Czech tree and the frontier node Sb of the English

tree means that the frontier (white) nodes must be filled at the same time by one rule. In this example, it is the rule *vedení* ↔ *executives*. If a frontier node is not aligned, it means that it is not translated in the other tree.

If we follow the vertical alignment of frontier nodes and roots of the little trees below them, we get the whole "parse tree"—in another words we can find a rule that will be "plugged" into the aligned pair of frontier nodes.

The last but not least, the frontier nodes are labeled with syntactical functions[1], we call it *frontier state*. The *Probabilistic Synchronous Tree Substitution Grammar* models the probability that a rule will be plugged into a given pair of matching frontier nodes with a given frontier state.

The idea of tree-to-tree transductions is so general that it can be applied to transformations between any two types of trees. In Czech to English machine translation, configurations transferring from Czech trees to English trees can operate either on the same analytical or tectogrammatical level, or they can go diagonally, e.g. from the Czech tectogrammatical trees to the English analytical trees. We can transform the trees in "Czenglish" tectogrammatical representation to the English analytical one. The tree-to-tree transductions could also be used for the "parsing" step from the analytical to the tectogrammatical representation.

4.2 Tree-to-Tree Mappings

Our goal is to describe the transformations of sentence structures that we may observe during the process of translation between two languages. Comparing the tectogrammatical tree for a sample Czech sentence "*Podle jeho názoru bylo vedení UAL o financování původní transakce nesprávně informováno.*", with the analytical tree for its English translation "*According to his opinion UAL's executives were misinformed about the financing of the original transaction.*" in Figure 4.1, we can find the corresponding groups of nodes (chunks) and list some of the mismatches that we observe:

1. The 2–1 match between the $PRED$ (predicate) of the Czech sentence *informovat nesprávně* and its English counterpart *misinformed*,

2. the elision (a 1–0 match) of the generated ACT (actor) of the Czech sentence,

3. the three 1–1 matches (*on* ↔ *his; původní* ↔ *original; vedení* ↔ *executives*),

[1] But we could consider any reasonable labeling.

4. the Czech $CRIT$ (criterium) *názor* expressed by the English Adv (adverbial) phrase *according to ... opinion* can be either classified as a 1–3 match, or we can say that the tectogrammatical functor $CRIT$ forms the English Adv subtree *According to* and that the lemma *názor* matches 1–1 with *opinion*,

5. the EFF (effect) *financování* can be taken either as in a 1–3 match with $AuxP$ *about the financing*, or we can think that the functor EFF generates the Adv node *about* and the lemma *financování* generates *the financing*,

6. the PAT (patient) *transakce* generates $AuxP$ *of the ... transaction* or, in two steps, the functor PAT gives birth to $AuxP$ *of*, the lemma *transakce* translates to *the transaction*, and there is a 1–2 match *transakce* $\leftrightarrow$ *the transaction*.

Such an informal description of observed transformations mixes lexical, functional, and structural information present in this tree-pair. In the following, we will have to proceed through several steps towards formal rules that capture all three types of information.

We can split the tree pair into corresponding chunks and number them as in Figure 4.2. A translation rule is represented by a pair of corresponding chunks. Filled nodes carry the lexical information; the other nodes are marked by their syntactical functions and can be substituted by other chunks with the same syntactical function[2]. Finally, the dashed bows between unfilled nodes indicate that the substitutions at these two nodes must proceed synchronously.

For example, the rule 1[3] formalizes our observation from item 1, i.e. that the part of the Czech tectogrammatical tree *informovat nesprávně*, preceded by some subtrees of ACT, $CRIT$, PAT, and EFF, will be translated by the part of the English analytical tree *were misinformed*, preceded by some subtrees of Adv and Sb, and followed by some $AuxP$. Rule 1 also specifies that the three pairs of subtrees of $CRIT$ and Adv, PAT and Sb, and EFF and $AuxP$ will be substituted at the same time, or in other words, that these pairs of subtrees will be translations of one another. Finally, the ACT node will not have any counterpart in the English tree.

Rule 2 corresponds to the observation 2, i.e. the generated actor is not translated into English. This rule maps the Czech chunk to a special $null$ chunk on the English side.

The informal observation mentioned in item 4 is expressed by rules 3 and 4. Rule 3 says that functor $CRIT$ should be translated as *according to*, and rule

[2]The label with the syntactical function refers to both the unfilled node and the substituting chunk below it.

[3]See numbers above the root node of each chunk in the Figure 4.2.

4 dictates the synchronous translation of the actual lexical information *názor* $\leftrightarrow$ *opinion*. [4]

4.3 A Probabilistic Synchronous Tree Substitution Grammar

In this section, we describe the details of the probabilistic model of the transduction, the method of parameter estimation, and the decoding algorithm.

In our formal description of the Synchronous Tree Substitution Grammar, we stick to the symbolic markup used in [Hajič et al., 2002, Eisner, 2003] where possible.

We start with the definition of the non-synchronous Tree Substitution Grammar, and then extend to the synchronous case. As an example, we will use again the same tree pair from Figure 4.1, as in the previous section.

4.3.1 Non-synchronous Tree Substitution Grammar (TSG)

The **Tree Substitution Grammar** (TSG) is defined as follows:

1. Let Q be the set of **states** [5], and let $Start \in Q$ be the name reserved for the initial state.

2. Let L be the set of **labels** on the nodes (words) and edges (grammatical roles).

3. Let τ be the set of **little trees** t defined as tuples $\langle V, V^i, E, l, q, s\rangle$, where

   - V is a set of **nodes**,
   - $V^i \subseteq V$ is a subset of **internal nodes** and its complement $V^f = V - V^i$ is a set of **frontier nodes**,
   - $E \subseteq V^i \times V$ is a set of directed edges that can start from internal nodes only. The graph $\langle V, E\rangle$ must form a directed and acyclic tree. [6]
   - The function $s : V^f \rightarrow Q$ assigns a **frontier state** to each frontier node.

[4] One could object that there is no mechanism that would prevent from using rule 4 first (substituting at $CRIT$ and Adv frontier nodes of rule 1), so that using rule 3 would not be possible any more, and the resulting sentence (missing the words *According to*) would be ungrammatical. This can be fixed by extending the set of syntactical functions, e.g. the unfilled Czech and English nodes of rule 3 could have labels $CRIT'$ and Adv', respectively. An alternative way of fixing this problem is to consider one larger rule $CRIT$ *názor* $\leftrightarrow$ Adv *According to ... opinion.*

[5] In our example we use the grammatical roles from the PDT

[6] We can see that the tree representing the whole sentence complies with the definition of the little tree with the empty set of V^f.

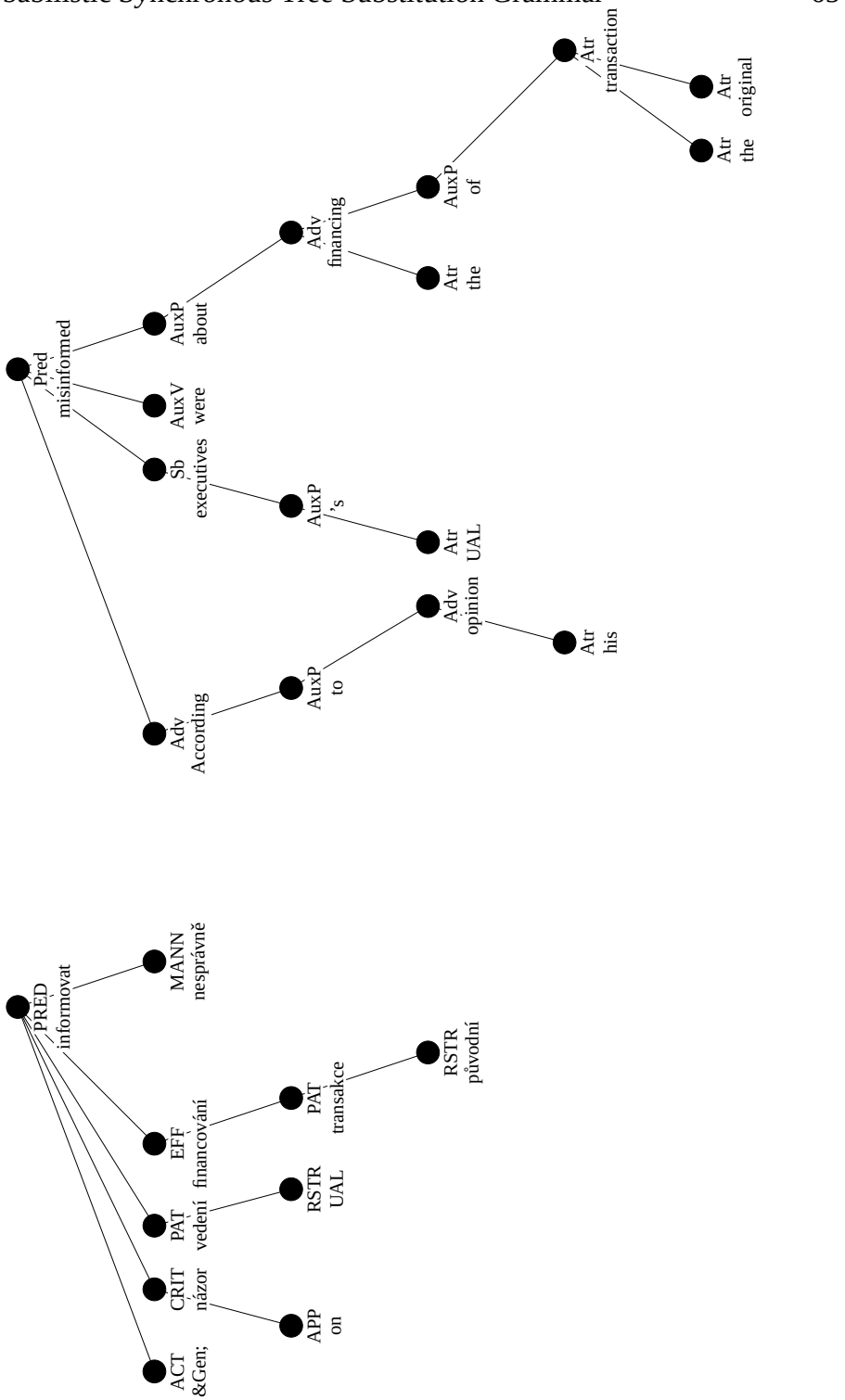


Figure 4.1: The tree pair for the tectogrammatical representation of the Czech sentence "*Podle jeho názoru bylo vedení UAL o financování původní transakce nesprávně informováno.*" and the analytical representation of the corresponding English translation "*According to his opinion UAL's executives were misinformed about the financing of the original transaction.*"

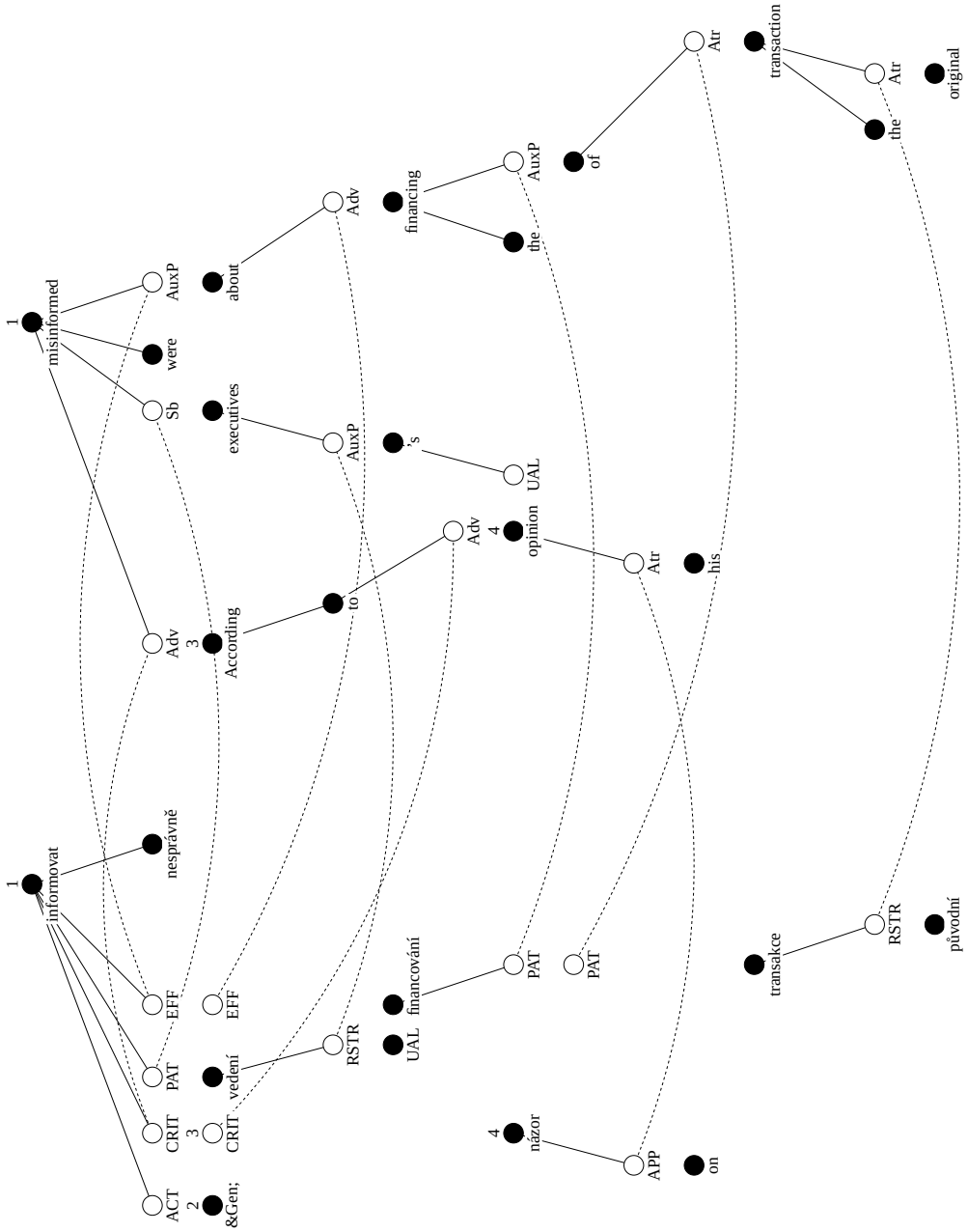


Figure 4.2: Aligned chunks of the tree structure for the tectogrammatical representation of the Czech sentence "*Podle jeho názoru bylo vedení UAL o financování původní transakce nesprávně informováno.*" and the analytical representation of the corresponding English translation "*According to his opinion UAL's executives were misinformed about the financing of the original transaction*".

- Let $r \in V$ be the **root** node of the tree, and let the **root state** q be assigned to the root. [7]
- Let $l : (V^i \cup E) \rightarrow L$ be a function assigning a label to each internal node or edge.

4. Finally, the Tree Substitution Grammar (TSG) is defined as the tuple $\langle Q, L, \tau \rangle$.

For convenience, we will use the shorthand $t.q$ for the root state, and other shortcuts for all other properties of $t \in \tau$ using the same analogy.

Let $d \in V^f$ be a frontier node of t, and t' be a little tree such that $t.s(d) = t'.q$ – in other words, the frontier state of d matches the root state of t'. We may define the operation of **substituting** t at d with t'. The result of this operation is defined as little tree:

$$\begin{aligned} SUBST(t, d, t') = \langle & t.V \cup t'.V - \{d\}, \\ & t.V^i \cup t'.V^i, \\ & t.E^* \cup t'.E, \\ & t.l \cup t'.l, \\ & t.q, \\ & t.s \cup t's - \{\langle d, t'.q \rangle\} \rangle. \end{aligned} \tag{4.1}$$

We obtain $t.E^*$ from $t.E$, by "redirecting" the edge originally pointing to d to the root of t'.

The process of **derivation** from the initial state $Start$ in the TSG is described by Algorithm 4.1:

Algorithm 4.1 The derivation process in TSG.

1. Start with any little tree $t \in TSG$, such that $t.q = Start$.
2. **while** $t.V^f \neq \emptyset$
3. **select** $d \in t.V^f$
4. **select** t' such that $t.s(d) = t'.q$
5. $t := SUBST(t, d, t')$

Line 4 of the algorithm hinted us to model the probability distribution over all possible little trees with root state q. Then the tree t' would be chosen with the probability $p(t' \mid q)$.

[7] If the root r is a frontier node, we can consider s such that $s(r) \neq q$.

Thus the probability of the derivation $qt^0 \dots t^k$ starting from t_0 and using k substitutions with little trees $t_1, \dots, t_k$ can be computed as:

$$p(q, t^0, \dots, t^k) = p(t_0|q) * \prod_{i=1}^{k} p(t'_i \mid t'_i.q) \tag{4.2}$$

The probabilistic TSG does not require $t.s(d)$ to be the same as $t'.q$.

4.3.2 Inside-outside Algorithm for TSG

The probabilities $p(t \mid q)$ can be automatically obtained from a treebank using the EM algorithm. By analogy with the measures and quantities used for the training of probabilistic context-free grammars [Jelinek, 1985], we will define inside and outside probabilities, expected counts, and state the re-estimation formula.

We say that "the tree t' **fits** node d" if there is some derivation, in which t' substitutes t at d and the result of the derivation is T. Note that the root state of t' can be any of Q, since the nodes of the resulting tree T do not imply any restrictions on states used during the derivation. Thus the iteration over all little trees t' fitting d includes variants for all $q \in Q$.

The probability that the grammar TSG generates the tree T from the state $Start$ is the sum of probabilities of all possible derivations, and can be computed as the **inside probability** $\beta_{T.r}(Start)$ by the induction in Algorithm 4.2:

Algorithm 4.2 The inductive algorithm for computing inside probabilities.

1. **for** each node c of T in bottom-up order
2. **for** each $q \in Q$, **let** $\beta_c(q) = 0$
3. **for** each little tree t that fits c, in a *safe order*
4. increment $\beta_c(t.q)$ by $p(t \mid t.q) \cdot \prod_{d \in t.Vf} \beta_d(t.s(d))$

The natural-language definition of the inside probability $\beta_c(t.q)$ is the probability of generating the whole subtree of T rooted at node c with the root state q. The Algorithm 4.2 is an example of the well-known chart-parsing approach. It starts with the leaf nodes, their inside probabilities $p(t \mid t.q)$ are retrieved from the probabilistic model. Then the algorithm traverses the tree in bottom-up ordering and collects inside probabilities for the nodes higher up in the tree.

Line 3 must iterate the little trees in a *safe order*. The little trees with frontier root nodes can be selected only after all other little trees with the internal root node have been evaluated.

The **outside probabilities** $\alpha_{t.r}(q)$ can be computed by the Algorithm 4.3:

Algorithm 4.3 The inductive algorithm for computing outside probabilities.

1. **for** each little tree t that fits $T.r$
2. **for** each $q \in Q$
3. **if** $q = Start$ **let** $\alpha_{t.r}(q) = 1$
4. **else** $\alpha_{t.r}(q) = 0$
5. **for** each node c of T, in top-down order
6. **for** each little tree t that fits c
7. **for** each $d \in t.V^f$
8. **for** each t' that fits d
9. increment $\alpha_d(t'.q)$ by $p(t' \mid t.s(d)) \cdot \alpha_{t.r}(t.q) \cdot \prod_{d' \in t.V^f - d} \beta_{d'}(t.s(d'.))$

The natural definition of the outside probability $\alpha_d(t.q)$ is the probability of starting with the root state $T.q$, generating all parts of the tree T outside of the subtree rooted at c, and generating any subtree rooted at c with the root state $t.q$

The **expected count** $C(q, t)$ of a little tree t used in the derivation of T can be computed by Algorithm 4.4:

Algorithm 4.4 The algorithm for computing expected counts.

1. Initialize $C(\_, \_) = 0$
2. **for** each node c of T
3. **for** each little tree t that fits c
4. Increment $C(q, t)$ by $p(t \mid q) \cdot \alpha_{t.r}(q) \cdot \prod_{d \in t.V^f} \beta_d(t.s(d))$

And finally, the re-estimation formula 4.3 for $p(t \mid q)$:

$$p(t \mid q) = \frac{C(q, t)}{\sum_{t'} C(q, t')} \tag{4.3}$$

In each iteration, the EM algorithm first computes the inside probabilities, the outside probabilities, the expected counts, and finally uses the re-estimation formula to obtain the new values of $p(t \mid q)$. Iterations are repeated until the $p(t \mid q)$ converges.

4.3.3 Synchronous Tree Substitution Grammar

We can extend the TSG to model the synchronous generation of a tree pair $T = (T_1, T_2)$. For this we will join two TSGs, $TSG_1 = \langle Q_1, L_1, \tau_1 \rangle$ and $TSG_2 = \langle Q_2, L_2, \tau_2 \rangle$, such that TSG_1 generates T_1 and TSG_2 generates T_2, with some restrictions on the operation of substitution.

The **synchronous tree substitution grammar** ($STSG$) is a tuple $\langle Q, L, \tau \rangle$, where

1. Q is a set of **synchronous root states**, $Start$ being as before a special initial state.[8]

2. $L = L_1 \times L_2$.

3. $\tau = \tau_1 \times \tau_2$ is a set of **little tree pairs**. The little tree pair t is a tuple $\langle t_1, t_2, q, m, s \rangle$, where the little trees $t_i = \langle V_i, V_i^f, E_i, l_i \rangle$ have a common synchronous root state q.

   The alignment of frontier nodes m is called **matching**, and is defined as a 1-to-1 correspondence (pairing) between subsets of V_1^f and V_2^f, such that unmatched frontier nodes are mapped to $null$. For 1-0 or 0-1 mappings, we use the concept of a $null$ tree that has empty sets of internal and frontier nodes[9]. The function $s : m \rightarrow Q$ assigns common frontier states to pairs of aligned frontier nodes.

The operation $SUBST(t, d, t')$ of **substituting** t at d with t' for aligned node pairs $d = (d_1, d_2)$ is defined such that $d \in m$ and $t.s(d) = t'.q$. The result of this substitution is a little tree pair

$$\begin{aligned} SUBST(t, d, t') = \langle & SUBST(t_1, d_1, t'_1), \\ & SUBST(t_2, d_2, t'_2), \\ & q, \\ & t.m \cup t'.m - (d_1, d_2), \\ & t.s \cup t'.s - (d_1, d_2, t'.q) \rangle. \end{aligned} \tag{4.4}$$

The process of **derivation** from the initial state $Start$ in $STSG$ is described by Algorithm 4.5:

Algorithm 4.5 The derivation process in $STSG$.

1. Start with any little tree pair $t \in TSG$, such that $t.q = Start$.
2. **while** $t.m \neq \emptyset$
3. **select** $d \in t.m$
4. **select** t' such that $t.s(d) = t'.q$
5. $t := SUBST(t, d, t')$

[8] For convenience, we may think of $Q = Q_1 \times Q_2$, but generally, the Q can be any set of synchronous root states.

[9] Note that the concept of the $null$ little tree is compliant with the rest of the definitions, except from the root of the $null$ tree, and the root state of the little tree pair containing $null$ tree. We leave this up to our intuition.

The formula 4.2 for computing the probability of the derivation can be used for the synchronous case as well.

4.3.4 Inside-outside Algorithm for $STSG$

In order to train the probabilities $p(t \mid q)$ of the $STSG$, we have to rework the Algorithms 4.2, 4.3, and 4.4 for the inside and outside probabilities, and expected counts, as well as the re-estimation Formula 4.3.

The definition of fitting has to be updated for the synchronous case: We say that t' **fits** node pair d, if t'_i fits d_i for $i = 1, 2$.

The Algorithm 4.6 computes the **inside probability** $\beta_{T.r}(Start)$, in other words, the probability that the $STSG$ generates a tree pair T from the initial symbol $Start$.

Algorithm 4.6 The inductive algorithm for computing inside probabilities for $STSG$.

```
1.  for each node c_1 of T_1, in bottom-up order
2.      for each node c_2 of T_2, in bottom-up order
3.          for each q ∈ Q, let β_{c_1,c_2}(q) = 0
4.          for each little tree t_1 that fits c_1
5.              for each little tree t_2 that fits c_2
6.                  for each probable matching m of frontier nodes of t_1 and t_2
7.                      construct t from q, t_1, t_2, and m
8.                      increment β_c(q) by p(t | t.q) · ∏_{d∈m} β_d(t.s(d))
```

Lines 4 and 5 must iterate little trees fitting a node pair c in a *safe order*. First, we have to evaluate the pairs with the $null$ little tree, then the little tree pairs with internal root nodes, and finally the little tree pairs with a frontier root nodes.

The Algorithm 4.7 computes the **outside probability** $\alpha_{t.r}(q)$:

Algorithm 4.7 The inductive algorithm for computing outside probabilities for $STSG$.

1. **for** each node c_1 of T_1, in top-down order
2. **for** each node c_2 of T_2, in top-down order
3. **for** each $q \in Q$
4. **if** $q = Start$ **let** $\alpha_c(q) = 1$
5. **else** $\alpha_c(q) = 0$
6. **for** each little tree t_1 that fits c_1
7. **for** each little tree t_2 that fits c_2
8. **for** each probable matching m of frontier nodes t_1 and t_2
9. construct t from q, t_1, t_2, and m
10. **for** each pair of matching frontier nodes $f \in m$
11. increment $\alpha_f(t.s(f))$ by $p(t \mid q) \cdot \alpha_{t.r}(q) \cdot \prod_{d \in m - \{f\}} \beta_d(t.s(d))$

The expected counts $C(q, t)$ are computed using the Algorithm 4.8:

Algorithm 4.8 The algorithm for computing expected counts for $STSG$.

1. Initialize $C(\_, \_) = 0$
2. **for** each node c_1 of T_1
3. **for** each node c_2 of T_2
4. **for** each little tree t_1 that fits c_1
5. **for** each little tree t_2 that fits c_2
6. **for** each probable matching m of frontier nodes t_1 and t_2
7. construct t from q, t_1, t_2, and m
8. Increment $C(q, t)$ by $p(t \mid q) \cdot \alpha_{t.r}(q) \cdot \prod_{d \in m} \beta_d(t.s(d))$

Finally, the probabilities are re-estimated using Formula 4.3, but this time the tree pairs t and t' iterate through all possible t_1, t_2, and m.

4.3.5 Decoding Algorithm for $STSG$

Once we have trained the probabilities $p(t \mid q)$ on a parallel treebank, we can use them for decoding. For a tree T_1, we want to find its translation T_2. The decoding process tries to cover the T_1 by the left sides of rules, and take the resulting tree on the right side as the result. In other words, find the most probable synchronous derivation that generates T_1 on the left hand side, and take the tree T_2 generated on the right hand side.

The most probable derivation is computed by a chart-parsing Algorithm 4.9.

Algorithm 4.9 The decoding algorithm for $STSG$.

1. **for** each node $c_1 = null$, and then $c_1 \in T_1.V$ in bottom-up order
2. **for** each $g \in Q$ let $\beta_{c_1}(q) = -\infty$
3. **for** each little tree t_1 that fits c_1 in a *safe* order
4. **while** t=*proposeNewRule*() //we have to try all possible t_2, q, m, s
5. find max $p(t \mid t.q) \cdot \prod_{d \in m} \beta_{d_1}(t.s(d))$ and store the t and $\beta_{c_1}(q)$ in the chart

As opposed to computing the inside probabilities, we do not have to fit nodes of T_2, and therefore no restrictions are put on the choice of t_2. That is also why the inside probabilities are indexed by c_1 only.

4.4 Conclusion

We have presented the details of the probabilistic Synchronous Tree Substitution Grammars, a new method for learning tree-to-tree transformations between non-isomorphic trees. The level of details published here is—to our knowledge—the first of its kind.

In the following Chapter 5 we will try to show that the presented method is appropriate, and that it is possible to implement a model that learns alignment between trees representing the sentence structure.

Chapter 5

Implementations of the Tree-to-Tree Transducer

As written above, the $STSG$s can be used to model transformations of tress of any type. One always has to define the conversion algorithm between the current tree format and the "computational" format of the tree: i.e. the set of states of the frontier and root nodes, and node labels.

As is usual, the ubiquitous limitations, such as speed, memory, and data sparseness are the main criteria for the implementation. To fit into memory and in order to perform in reasonable time, the allowed shapes of the rules have to be *restricted*, as well as the rule set has to be *pruned*. On the other hand, *unseen* rules have to be modeled using a *back-off scheme*.

The actual implementation of mapping, rule restrictions, pruning, and back-off scheme are always highly specific to a given matter of modeled trees. In this section, we describe our attempts to use the framework for three tasks: extraction of mappings between tectogrammatical and analytical representation of Czech, and for training the transfer from Czech tectogrammatical or analytical tree structures into English analytical tree structures.

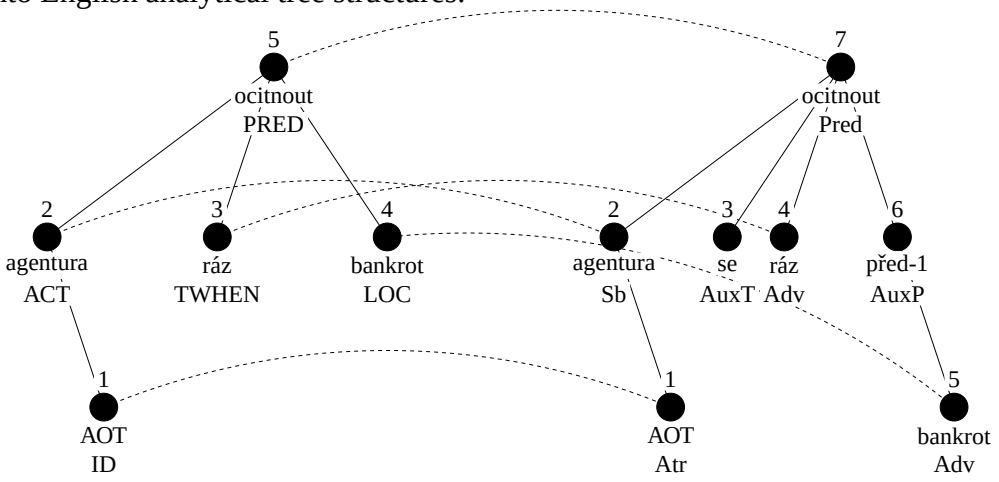


Figure 5.1: The tree pair for the tectogrammatical and analytical representations of the Czech sentence: "*Agentura AOT se rázem ocitla před bankrotem.*"

5.1 Creating the Set of Rules

The Figure 5.1 contains an example of the tectogrammatical and analytical representations of the Czech sentence: "*Agentura AOT se rázem ocitla před bankrotem*". Both trees were converted into the computational format, the differences from the PDT original are the following: The technical root nodes and nodes of the final punctuations were removed. The function labels are attached to nodes, not to edges. The corresponding nodes are connected by dashed lines (based on the PDT annotation), and finally – all nodes are numbered by postfix ordering.

The $STSG$ describing the transformations between the tectogrammatical and the analytical structures can be constructed as follows:

1. The set of frontier states will include all combinations of tectogrammatical and analytical functors, and the initial state $Q = \{Start\} \cup \{TF \times AF\}$.

2. The set of labels will contain all pairs of tectogrammatical and analytical lemmas used in PDT (node labels), as well as pairs of functors (edge labels): $L = \{TL \times AL\} \cup \{TF \times AF\}$.

3. The set of rules is constructed by going through all tree-pairs in the corpus, all their nodes where a little tree can be rooted, all combinations of possible shapes of the little tree-pairs, and all possible matchings of their frontier nodes:

   (a) A little tree rooted in some node c of the dependency tree is constructed as follows: Any continuous subtree rooted at c can be taken as the set of internal nodes V^i, or V^i can be empty. The set of frontier nodes V^f contains all children of internal nodes, such that the children themselves are not among internal nodes, $V^f \cap V^i = \emptyset$.

   (b) In case of the empty set of internal nodes, c is the only frontier node. The function s assigns a pair of tectogrammatical and analytical functions from the edges incoming into the pair of frontier or root nodes[1], and the labeling l assigns lemmas to nodes and syntactical functions to edges.

   (c) The matching m of frontier nodes is chosen freely, or we can also impose various constraints taking into consideration the original annota-

[1]Note that this setup does not allow derivations with "renaming" frontier states mentioned in Footnote 4 in Chapter 4. To allow this, we have to consider all variants of frontier and root states s and q. This is technically impossible in the general case, but still doable for some specific cases, e.g. if one of the little trees consists of just one frontier node r, we can assign a root state q different from the frontier state of r.

tion of PDT, for example that matches present in the PDT annotation must be preserved in the rule.

4. Because all derivations have to begin with the $Start$ state, we have to add special rules with the root state $Start$. The rules have a form of a pair of little trees, each consisting of one frontier node. The frontier state can be any combination of syntactical functions that can possibly occur in the root of the PDT trees.

5. Due to speed and memory limitations, we exclude trees with nodes having more than 6 children from the training corpus.

6. Also the size of the little trees we substitute with, will be limited to a maximum of 2 internal nodes. Note that the sample rules given in the Figure 4.2 are still compliant with these constraints.

5.2 Finding a Back-off Scheme

There are two reasons why we always have to find suitable back-off schemes: the lack of generalization, and memory limitations. Even after applying the above-mentioned restrictions on the rules, the data is still sparse.

To better explain the seriousness of the problem we face, let us imagine that even the simplest shape of the synchronous rule – consisting of one internal node on both sides – is equivalent to the t-table with entry-translation pairs used by IBM models. In this case, the size of the dictionary is still reasonable but the OOV (out-of-vocabulary) rate is an issue. Moreover, we have to train on structurally annotated data, which is harder to find than a parallel corpus of plain text, and thus we are limited to roughly 20,000 sentence pairs contained.

When we consider more complex synchronous rules, the memory limitations become crucial and the data sparseness severe.

On one hand, if we want to make use of the syntactic structure, we have to consider larger little trees with more nodes, but on the other hand, it is very hard to find a more complex synchronous rule that repeats more than once in training data of about $20,000$ tree-pairs. And it implies a problem during the decoding: when a new tectogrammatical tree is being transformed into an analytical one, the most of the hypothesized more complex synchronous rules would be seen for the first time, and thus scored as unknown rules. Such a model would not generalize at all.

Memory and time limitations are also a serious problem. As the Table 5.1 shows, the average number of observed little trees per one “big” tree is around

	Tectogrammatical trees	*Analytical trees*
Avg. nodes per sentence	11.4	13.0
Avg. rules per node	3.9	3.9
Avg. rules per sentence	44.7	50.9
Total sentences	10,000	10,000
Total nodes	114,174	129,771
Total rules	446,696	509,084

Table 5.1: Non-synchronous rules statistics on Prague Dependency Treebank

50. The number of combinations of both sides grows with the square of the sentence length. This number has to be multiplied by possible alignments of frontier nodes.[2] This leads to tens of millions rules observed on a corpus of $10,000$ sentence pairs.[3] If we represent one rule using 100 bytes on average, we see that a model with 10 million rules would use 1 GB of memory.

In the following, we will try to find experimentally, what parts of information can be omitted from the rules. We are looking for such back-off models that can fit in memory, generalize observed phenomena and still capture the syntactical relations between tree components. We feel that finding a good back-off scheme is crucial to make the tree-to-tree models work, and that it has to be done iteratively, in many experiments.

Intuition tells us to divide the problem into parts, and then to model these parts separately. Roughly speaking, a synchronous rule consists of two (non-synchronous) little trees and a mapping of their frontier nodes. Firstly we will focus on single little trees and analyze the histograms of tree structures for various back-off models. Later, we will discuss how to aggregate pairs of little trees and how to model mappings between their frontier nodes in order to model the whole synchronous rules.

5.2.1 Single Little Trees Back-off

We understand a single little tree as an object carrying three types of information: the structure, the frontier states, and the lexical information. Each of these tree types can be represented using a certain level of detail. In the following, we will experiment with various representations of single little trees. Table 5.2 shows counts of unique non-synchronous rules in tectogrammatical and analytical lay-

[2] It is hard to estimate the average numbers, but the experiments showed that there are 300 million rule observations for $10,000$ sentence pairs. Most of them can be filtered by pruning.

[3] For comparison, the translation table used in GIZA models trained on $20,000$ sentence pairs has around $100,000$ entry-translation pairs.

ers of PDT. Figure 5.3 shows the histogram of single little trees occurring in the analytical part of the PDT.

Let us consider these levels of details:

- **The structure**

  1. The structure of the little tree can be fully represented. Since the little trees are projective, a linearized form using bracketing can represent them.
  2. In the next step, the brackets can be removed from the representation and the little tree becomes an n-gram of nodes in the original word order. However, it can be shown that if the number of internal nodes is already limited to 2, and if the little trees are projective, this representation is equivalent to the full representation.
  3. The next step is to ignore the word order and to represent the little tree as a bag of nodes.

- **The frontier state**

  1. The frontier states are fully represented by tectogrammatical or analytical functors.
  2. In order to reduce the number of functors, it is useful to merge similar functors into a smaller number of classes. On the other hand, when fine-tuning the system, we may split some functors into more variants in order to model some syntactical phenomena better, e.g. to propagate number from the morphological tag into the frontier state in order to model agreement.

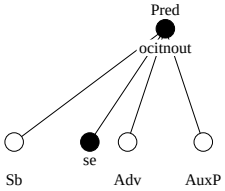
- **The lexical information**

  1. The full lexical information can be represented either by the word form (or by the lemma and the morphological tag).
  2. In order to reduce the number of variations, the word form can be replaced by its lemma, or by its morphological tag. It would be also possible to use syntactic-semantic classes of words [Brown et al., 1992] instead of word forms.
  3. Further on, the number of recognized positions in the morphological tag can be reduced to the most basic ones, such as part of speech.
  4. Finally, the lexical information can be ignored at all – all words are mapped to one class. For tectogrammatical little trees, we should keep a special class for generated nodes.

We look for a back-off model that uses some of the previously mentioned approximations. Let us try the following combinations:

- The representation of the **FULL MODEL** contains full information about the rule: the tree structure, the lemma for each internal node, and the functors for each frontier node.

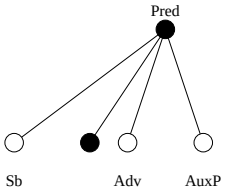
  FULL MODEL

  The rule can be also represented in a linearized form [4] :
  $Pred \rightarrow Sb$ **se** Adv **ocitnout** $AuxP$.

  We can see that the number of full little trees is in linear correlation to the number of sentences.

- The first and natural proposal is to ignore the lexical information of the little trees, and to work only with the structure and functions. The representation of **w/o LEXICAL** model contains the tree structure, the functors for each frontier node, but the lemmas for internal nodes are stripped off.

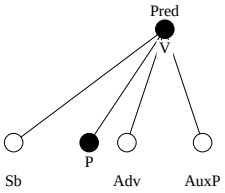
  The rule can be linearized, the lexical nodes are replaced by placeholders: [4]
  $Pred \rightarrow Sb$ _ Adv _ $AuxP$

- The **w/o LEXICAL A** back-off model ignores the structure and lexical information of internal nodes:
  $Pred \rightarrow Sb\ Adv\ AuxP$

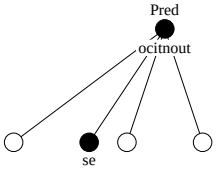
  There are no placeholders for internal nodes.

- The model **MORPHOLOGY** is a half-way between keeping and ignoring the lexical information of the internal nodes.

[4] Due to the limitation restricting the number of internal nodes to 2, the linearization does not loose any structural information.



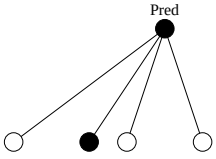
- Another proposal is symmetric to the previous ones: in the representation of the **w/o FUNCTIONS** model, we store the tree structure and lemmas for internal nodes, and omit the functors of frontier nodes.



The rule can be linearized, the frontier nodes are represented by placeholders [4] :
$Pred \rightarrow$ _ **se** _ **ocitnout** _

- The **w/o FUNCTIONS A** back-off model ignores the structure and functional markup of the rule, and represents a little tree only by its lexical information present in its internal nodes, i.e. by n-grams. There are no placeholders for frontier nodes:
$Pred \rightarrow$ **se ocitnout**

- The representation of the back-off model **STRUCTURE** combines the previous two approaches: it contains only the tree structure, lemmas and functors are omitted.



When analyzing the Table 5.2, we may see that representations containing the lexical information, i.e. models FULL MODEL, W/O FUNCTIONS, and W/O FUNCTIONS A, grow in the number of unique rules almost linearly with the size of the training data. It means that within the range of the training data in thousands of sentence-pairs the OOV problem cannot be significantly improved by adding more data. The curves of the back-off models, which do not contain the full lexical information give us certain hope (though a little one) that the growth would slow down with more data. At least, the absolute numbers of unique rules are lower, as the Table 5.2 shows.

#sentences	1,000		5,000		10,000		15,000		20,000		30,000	
	AR	*TR*	*AR*	*TR*	*AR*	*TR*	*AR*	*TR*	*AR*	*TR*	*AR*	*TR*
FULL MODEL	12,717	12,018	54,099	51,163	109,326	103,937	164,158	157,369	213,628	208,233	308,656	304,718
W/O LEXICAL	4,957	5,891	16,461	20,515	29,066	37,359	40,714	53,633	50,009	68,704	66,428	94,674
W/O LEXICAL A	3,553	4,440	11,131	14,591	19,270	26,388	26,686	37,092	32,433	46,627	42,551	63,443
MORPHOLOGY	7,884	7,007	28,742	25,079	52,800	45,777	75,807	66,009	94,951	84,929	129,819	117,631
W/O FUNCTIONS	12,087	11,465	50,588	47,624	102,099	96,452	152,480	144,921	197,448	190,258	283,149	275,246
W/O FUNCTIONS A	10,970	10,366	43,298	40,632	85,910	81,329	126,729	120,592	161,822	156,134	228,491	222,679
STRUCTURE	1,894	2,117	3,860	4,319	5,432	5,897	6,415	7,062	7,024	7,941	7,973	9,051

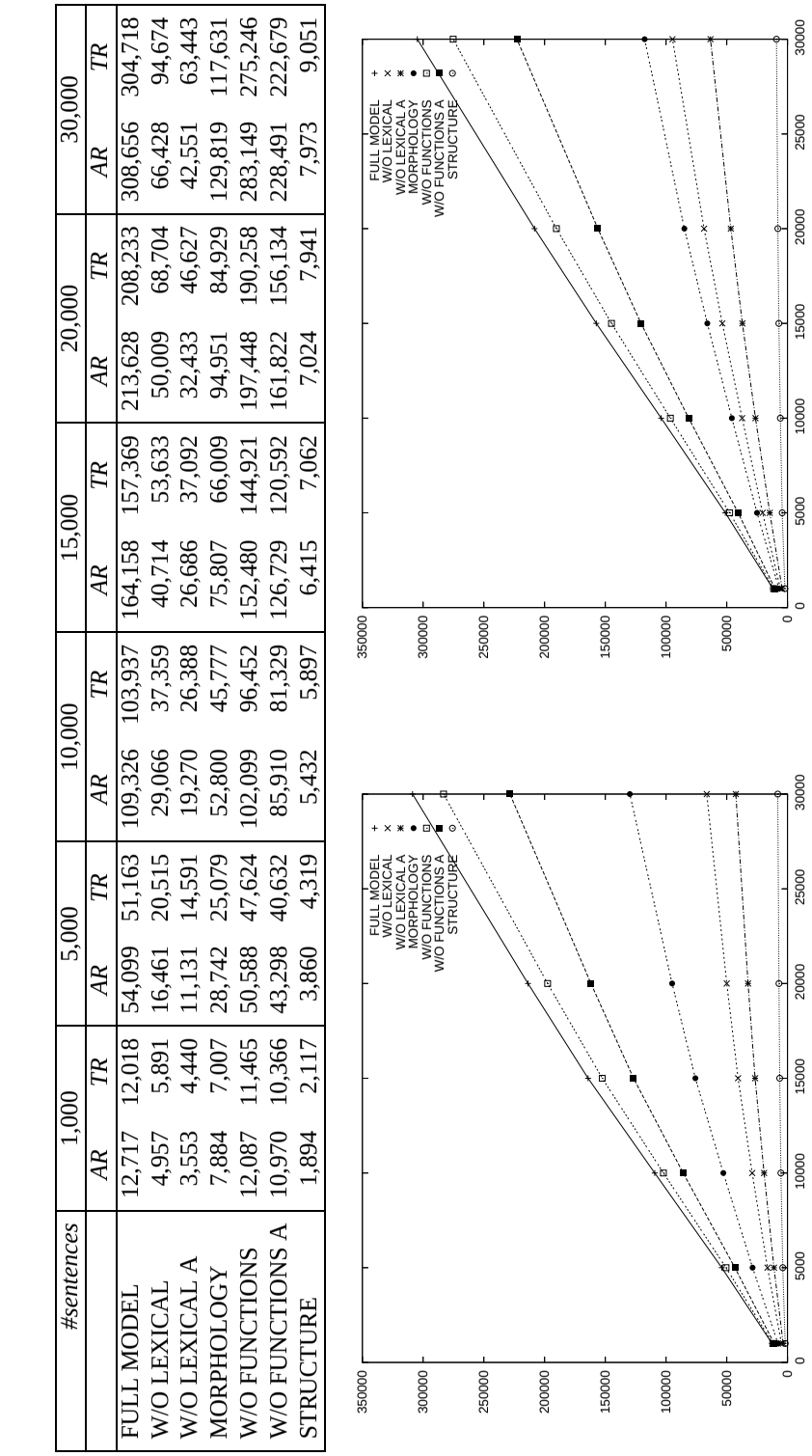


Figure 5.2: Rule counts of TSG non-synchronous rules for tectogrammatical and analytical representations in PDT.

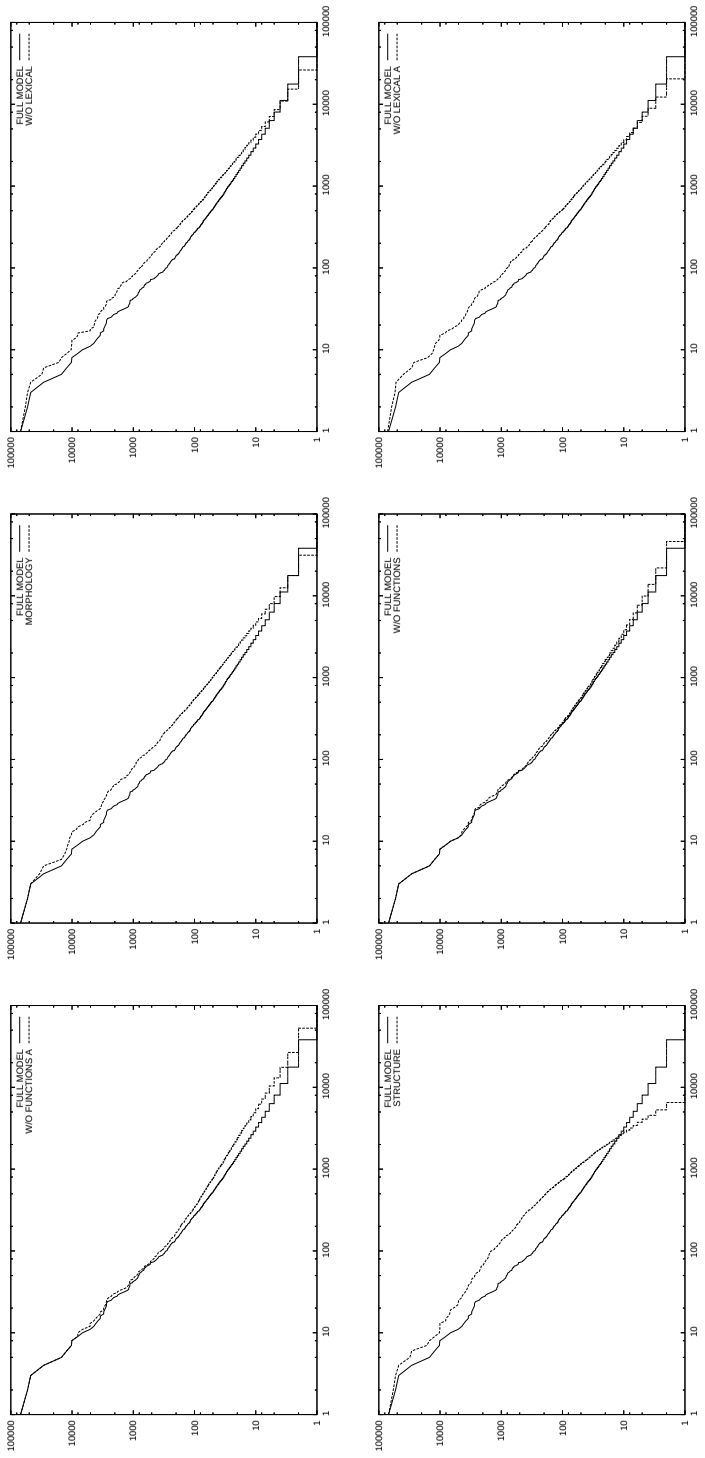


Figure 5.3: Frequencies of non-synchronous little trees occurring in the analytical part of the PDT, for listed back-off schemes.

5.2.2 Modeling Synchronous Rules

In order to model synchronous rules describing the transformations between Czech tectogrammatical and analytical trees, we have to aggregate a pair of single little trees and to define *mapping* between their frontier nodes. The mapping usually strongly depends on the content of both little trees, since it defines the pair of frontier nodes at which we substitute by a corresponding subtrees. For example, we can see that the probability of whether an analytical frontier node Sb is aligned either with a tectogrammatical ACT or PAT depends on whether the voice of the governing node is active or passive.

Also the possible representations of the mapping is applicable only for such a back-off models that represent frontier nodes, and their ordering.

We can consider an exact representation that can be obtained if we number the frontier nodes of the left little tree from left to right from 1 to m, and the frontier nodes of the right little tree from 1 to n. For numbers x and y, where $1 \leq x \leq m$ and $1 \leq y \leq n$, the symbol (x–0) means deletion, (0–y) means insertion, and (x–y) means synchronous substitution at frontier nodes x and y. This representation is applicable to back-off models **FULL MODEL**, **w/o LEXICAL**, **w/o FUNCTIONS**, **STRUCTURE**, **w/o FUNCTIONS B**, **w/o LEXICAL A**, and **w/o LEXICAL B**.

If we rewrite the probability of a synchronous rule as:

$$p(t \mid q) = p(t_1, t_2 \mid q)p(m \mid t_1, t_2, q) \tag{5.1}$$

where m can be any mapping of frontier nodes t_1 and t_2, we can experiment with various approximations and ignore some parts of information present in t_1, t_2, and q. For example, we can ignore the root state q and the ordering of frontier nodes, and model the probability of mapping as $p(ACT - Sb) \cdot p(TWHEN - Adv) \cdot p(AuxP - LOC)$.

5.2.3 Using a Translation Dictionary

The $STSG$ as defined in Chapter 4 does not contain any parameters related to the size of the little trees nor the number of derivation steps, nor any penalizations of insertions and deletions. A practical implementation of a training algorithm that has to deal with data sparseness, prune the rules with low probabilities, and use back-off models ignoring most of the lexical information, easily runs into difficulties. To overcome this drawback, we had to add a heuristic scoring function to the probabilistic model. Let us have a closer look at the problem:

- As the first, there is a problem resulting from the fact that the models taking into account lexical information cannot be used for their memory requirements and data sparseness. This would cause problems during the decoding,

since the scoring of hypotheses would not reflect lexical criteria. Intuitively, some additional rescoring based on translation dictionary seems necessary to bring back the lexical criteria to the model.

- As the second, it is obvious that a rule consisting of smaller little trees has a higher chance to occur more frequently than a larger rule. This is also the case of rules with frontier nodes aligned to $null$. A practical implementation of the training algorithm tends to prefer small rules to rules containing larger little trees. It also prefers rules with insertions and deletions to rules with mutually aligned frontier nodes. On the other hand, the motivation for $STSG$ was to capture syntactic phenomena, it means to have larger rules on both sides with aligned frontier nodes. The lexical scoring function should favor larger little trees and suppress deletions.

We have experimented with two variants of scoring function, both of them were based on a translation dictionary:

1. The scoring function is computed as follows: The lexical information[5] from internal nodes $V^i_{t_1}$ and $V^i_{t_2}$ is respectively collected into "French" $\mathbf{f}$ and "English" $\mathbf{e}$ sentences, and they are rescored by

$$score = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} p_{l,m} \prod_{j=1}^{m} t(f_j \mid e_{a_j}). \tag{5.2}$$

   Since the number of internal nodes is restricted to ≤ 2, the Equation 5.2 is a simplified version of IBM model 1 from Equation 1.4. The parameter $p_{l,m}$ gives us the way to introduce compensations for various numbers of internal nodes on both sides.

2. The second version is simpler. We sum the probabilities of all entry-translation pairs

$$score = \sum_{i=0,\ldots,l} \sum_{j=0,\ldots,m} t(f_j|e_i) \tag{5.3}$$

   This version does not have any parameter for balancing different rule sizes as in the previous case. The sum of probabilities already favorises larger rules.

[5]Here we mean the full lexical information, not the back-off representation.

5.3 Pruning the Rules

It is obvious, that most of the observed rules do not have any linguistic sense, and that we are interested only in a little fraction of them.

For the reasons of efficiency and memory limitations, the large pruning of the huge space of possible rules is necessary, but still we must be able to consider rules, which are capable of doing the structural transformation.

The current implementation is restricted to trees with nodes having max. 6 child nodes, and we deal with rules consisting of a pair of little trees with has 0–2 internal nodes each.

5.3.1 Using PDT Links

We may use the fact that the tectogrammatical and the analytical nodes are in the PDT linked together. We can design a heuristic pruning method based on the links between the nodes, so that we do not have to evaluate every combination of little trees on both sides.

Here is a list of rules we have experimented with:

- Discard rules, where an internal node of one little tree is PDT-aligned with a node outside of the other little tree.
- Discard rules, where an internal node is PDT-aligned with a frontier node. This rule should not be applied neither to hidden nor generated nodes.
- If there is a PDT-alignment between two frontier nodes of the rule, then discard all rules that do not align these two nodes.
- Discard rules that "do not make any progress", such as rules consisting of one frontier node on both sides.[6]

5.3.2 Threshold for Expected Counts

Another possible pruning criterion is $\texttt{low}\Delta$ – a simple threshold for the increment of the expected count. When the Algorithm 4.8 computes the expected counts of synchronous rules, the new rule is added into the model (line 8 of the algorithm) only if its increment of the expected count is higher than $\texttt{low}\Delta$.

[6] This a reasonable restriction of the grammar similar to excluding CFG rules of type $A \to A$. Since we do not allow renaming the frontier state, we do not have problem with rules $A \to B$ that could bring in cycles.

5.3.3 Lazy Pruning

When the size (number of rules) of the whole model exceeds the threshold `max_model_size`, the global pruning can be triggered in order to reduce the model size to some required limit. The pruning throws away the least probable rules.

5.3.4 Pruning by Non-synchronous Rules

It is possible to count all non-synchronous little trees in the training data (It was possible to keep all of them in memory), and to impose a `nonsynch_threshold` for the frequency of the non-synchronous little trees. Then the training algorithm evaluates only synchronous rules that consist of non-synchronous trees with frequency higher than the threshold.

5.4 Computational Aspects

Memory and speed are the crucial limitations. The success of the statistical model depends on the number of rules that fit into memory and on the time spent for accessing the rules. Here we describe some optimizations we have implemented.

5.4.1 Representation of the Synchronous Rules

It is necessary to use a memory-efficient representation of rules. The synchronous rules are stored in a serialized form similar to examples in 5.2. Each back-off representation of the same rule has different serialization. The serializations are keys for accessing the probability $p(t \mid q)$. The model is stored in a form of TreeMap structure (TRIE), which allows for efficient memory usage and fast insertions.

5.4.2 Synchronous Rule Iterators

When iterating the rules observed in the sentence pair, we use a representation of rules, which is optimized for speed. There are two functionalities that have the strongest influence on performance:

1. `findNextRule()` – a method for finding the next synchronous rule. The method iterates the tree-pair in both prefix or postfix ordering, traversing through all possible combinations of little trees and alignments of their frontier nodes.

2. `getSerialization()` – a method for serializing the current synchronous rule (both little trees and alignment of frontier nodes).

When profiling and optimizing the computation, we found out that most of the processor's time is spent on string operations due to serializations of synchronous rules. Since the right hand side of the synchronous rule iterates through the whole variety of non-synchronous little trees all over again for each possible left hand side (as the inner of the two nested loops), we have decided to cache the serializations of the little trees for each sentence pair. This technique speeded up the computation 5 times.

5.5 Training

The training of the tree-to-tree transducer model is an iterative procedure. There are two models used in each iteration: The *scoring model* is used for computing inside and outside probabilities, and expected counts, and the *new model* is used for storing and incrementing the expected counts. After the whole traversal of the training portion of the data, the expected counts are used to re-estimate the probabilities.

One iteration of EM algorithm visits all sentence pairs from the training corpus. Upon entering each sentence pair, each tree is labeled by indexes in postfix order. Then the *chart* is created to store the data needed by algorithms 4.6, 4.7, and 4.8 – inside, outside probabilities, optionally also the Viterbi probability and the Viterbi rule. The chart is a 2-dimensional array, the data for a given pair of nodes is stored on position defined by the postfix indexes of both nodes.[7] The probabilities are stored as logarithms.

5.6 Transformations between the Tectogrammatical and Analytical Representations in PDT

The transformations between the two layers of structural annotation of the Prague Dependency Treebank seem to be one of the easier tasks for the $STSG$: the trees are annotated manually on both analytical and tectogrammatical levels, so the data are as clean as possible. Also the annotation process, where the analytical trees are sequentially modified into the tectogrammatical ones, gives us the reason to believe that the structures are more similar than if they were created by two independent annotation processes, and that they are even more similar than if they originated in two sources, such as in translation task. Moreover, the nodes of the tectogrammatical trees contain links to the analytical nodes they come from, and

[7] If we allow "renaming" frontier states, then the chart becomes a 3 dimensional array – one additional dimension for the frontier/root state.

this information can be also used for pruning the rules, and for initial estimation of rule probabilities.

The probabilistic translation dictionary $t(f|e)$ maps tectogrammatical lemmas to analytical ones and was created from the PDT as relative counts of all pairs of tectogrammatical and analytical nodes.

The resulting tree-to-tree alignments presented in Appendix A.1 were obtained by the following sequence of steps:

1. Czech tectogrammatical-to-analytical "translation" dictionary was created by traversing the whole PDT and using links between tectogrammatical and analytical nodes.

2. A set of non-synchronous rules occurring more than once in the training data was collected for the purposes of pruning.

3. An initial iteration was run with scoring model with uniform probabilities. These probabilities were rescored by the heuristics 5.3 based on the "translation" dictionary.

4. A simple smoothing of back-off models was applied. The back-off models were tested on 100 sentence pairs from held-out data for a set of Λ parameters, and the winning Λ was used in the second iteration.

5. The second iteration was run with the model trained in the initial iteration. Viterbi alignments were traced during the computation of inside probabilities.

The following setup has been used:

- The size of the training data was $20,000$ sentence pairs. After applying the restrictions for the maximum number of children per node, the training data shrunk to $13,910$ sentence pairs.

- The following back-off schemes were used: **LINEARIZED**, **MORPHOLOGY**, **W/O LEXICAL**, and **STRUCTURE**.

- The **MORPHOLOGY** back-off model used a reduction of the full positional morphological tag to three positions. The reduction went as follows:

  - The tag was initially set to `---`.
  - The first position of the tag was always copied from the original tag.

  - If the word was noun or adjective, the case and number were respectively copied to positions 2 and 3. If the word was verb, the person and number were copied.

- The pruning by the PDT links was not applied.

- Due to the pruning by non-synchronous rules the model could fit into memory for the whole training data, thus the lazy pruning was not applied.

One iteration took approximately 12 hours.

5.7 Transfer between Czech Tectogrammatical and English Analytical Representations in PCEDT

Training alignments between the Czech tectogrammatical and English analytical representations is a great challenge, since the $STSG$ has to model two processes: the transfer between the two languages and the transition from deeper level of representation to a shallower one.

The experiments were carried out on the PCEDT corpus. Resulting tree-to-tree alignments were obtained by almost the same method as described in Section 5.6, the differences were the following:

- The training data were the first $20,000$ sentence pairs from the training section of the PCEDT. After applying the restrictions for the maximum number of children per node, the training data shrunk to $13,002$ sentence pairs.

- A parallel corpus of plain text was extracted from the PCEDT: the Czech part contained tectogrammatical lemmas prepended with the first position of the morphological tag, i.e. the POS; the English part contained analytical lemmas prepended by the first position of the analytical morphological tag.

- A probabilistic dictionary $t(f|e)$ was trained on these data using GIZA++ [Och and Ney, 2000]. Figure 5.4 contains a sample from this dictionary.

- The following back-off schemes were used: **LINEARIZED**, **MORPHOLOGY**, **W/O LEXICAL**, and **STRUCTURE**.

Czech entry	English translation	$t(e \mid c)$
V–chodit	N–Tigers	0.0487005
V–chodit	N–nomination	0.0807828
V–chodit	R–forth	0.121751
V–chodit	V–refer	0.207133
V–chodit	V–shuttle	0.243414
V–chodit	V–walk	0.298218
V–chodit_se	N–stereo	1
V–chovat_se	V–act	0.310038
V–chovat_se	V–behave	0.299106
V–chovat_se	V–live	0.140955
V–chovat_se	V–perform	0.113101
V–chovat_se	V–treat	0.136801
V–chrlit	V–spew	1

Figure 5.4: An example of the Czech – English probabilistic dictionary.

5.8 Transfer between Czech and English Analytical Representations in PCEDT

Experiments training alignments between the Czech and English analytical representations were carried out to compare the behavior of the $STSG$ implementation on a different type of data. On one hand, the transfer is done on the same levels of representation, and we may expect that there is a certain group of constructions common to both languages, which uses the same tree structure, on the other hand, the layers are relatively shallow, so there will be also the other group of constructions using very different tree structures. It will be interesting to compare the resulting alignments with those extracted in experiment described in Section 5.7.

- The experiments were carried out on the same portion of PCEDT as in Section 5.7. The training set of $20,000$ sentence pairs shrunk to $14,530$ after applying the restrictions for the maximum number of children per node.
- A parallel corpus of plain text for GIZA++ extraction of the translation dictionary consisted of analytical lemmas. The Czech lemmas were prepended with three positions copied by the same way as described in Section 5.6, the English lemmas were prepended only with the first position of the morphological tag.
- It is interesting that the computation process did not fit in memory when the same back-off and pruning schemes as in Sections 5.6 and 5.7 were

	Czech tecto-gramma-tical trees	*Czech ana-lytical trees*	*English analytical trees*
Avg. nodes per sentence	14.9	19.1	19.9
Avg. rules per node	3.9	3.9	3.9
Avg. rules per sentence	58.5	75.5128	78.79
Total sentences	10,000	10,000	10,000
Total nodes	148,835	191,282	199,475
Total rules	585,340	755,128	787,900

Table 5.2: Non-synchronous rules statistics on Prague Czech-English Dependency Treebank

used. The process allocated about three times as much memory as in the tectogrammatical-to-analytical experiment. The explanation is that training on analytical tree pairs generates about tree times more rule observations than observing tectogrammatical-to-analytical pairs, e.g. the same (the first) sentence pair generated $536,983$ rules for the analytical trees, but only $214,451$ rules for the tectogrammatical-to-analytical pair.

To explain this disproportion, it may be useful to compare Tables 5.1 and 5.2. They show that the data in PDT are different from those in PCEDT. The PCEDT sentences are longer on average, and also the ratio of tectogrammatical nodes to analytical ones is 78% in PCEDT, while in PDT it is 88%. The number of non-synchronous tree pairs grows with the square of sentence lengths, and yet it must be multiplied by the possible alignments to get the number of synchronous rules. That is why the model grows so rapidly.

Thus only the **W/O LEXICAL**, and **STRUCTURE** models could be used.

5.9 Evaluation of Results

The results of the alignment method were evaluated manually. The resulting Viterbi alignments for the first 20 tree pairs were examined. The training algorithm automatically records the Viterbi alignment in the form of the LaTeXsources that can be compiled and visually evaluated in GSview. Examples of the Viterbi alignments are in Appendix A.

One of the conclusions from the experiments we have carried out is that the data is extremely sparse. We realized that the highest sum of inside probabilities

of the held-out data is for Λ, which multiplies the simplest model **STRUCTURE** by 0.95, while the other models do not influence the score almost at all.

Another experience we have gained is about the absolute importance of the lexical rescoring. If the lexical rescoring is turned off, the method does not work at all, the system aligns trees using m-0 and 0-n rules.

We can also say that the limitation to ≤ 2 internal nodes is not so severe, and that most of the tree pairs can be correctly aligned within these restrictions.

5.9.1 Evaluation of Czech Tectogrammatical-to-Analytical Alignments

The task of finding alignments between tectogrammatical and analytical representations for the same Czech sentence works well, and almost all alignments are correct. Nevertheless, there are some transformations that cannot be handled by the $STSG$, such as in Figure A.2: The nodes for word *téměř* are misaligned, because there is no mechanism in $STSG$ that would model this transformation. Due to the restricted number of internal nodes, the nodes *téměř* and *být* cannot be in the same synchronous rule. Nor any kind of "late binding" is possible. If we consider a rule that has a frontier node on position of the tectogrammatical *téměř*, then it must be aligned either with some frontier node of the analytical little tree, or with $null$.

5.9.2 Evaluation of Czech-English Alignments

Figure A.3 contains a pair of the tectogrammatical tree for the Czech sentence *"Asociace uvedla, že domácí poptávka v září stoupla o 8,8 %."* and an analytical tree for the English sentence "*The association said domestic demand grew 8.8% in September.*". We can see that the Czech tree has 7 nodes, while the English one has 10, three nodes were added: *the*, 8.8, and *in*. There is also one structural change: the Czech LOCation *září* depends on the ACTor, while the English adverbial phrase *in September* depends on the $Pred$icate. Figure A.4 presents the Viterbi alignment. We can see that rules on lines 3, 4, and 6 handle insertions of these extra nodes, while the rule on line 2 handles the change of the tree structure. Table A.2 contains the computational chart. The first two columns identify the node pair in which the synchronous rule is rooted. The following three columns contain in sequence the inside probability, the outside probability, and the Viterbi probability. The last column is occupied by the Viterbi rule.

Although the results seem to be very promising, the space for improvements is still huge, e.g. Figure A.6 contains Viterbi alignment for sentence pair "*Poptávka trvale stoupá za podpory prospotřebitelské vládní politiky, řekl mluvčí asociace.*" and "*Demand has been growing consistently under the encouragement of pro-consumption government policies, an association spokesman said.*". The 3^{rd} syn-

chronous rule rooted at nodes 7 and 12 represents wrong alignment, since the internal structure *stoupat* is wrongly mapped to *be grow*, as well as the 5^th rule inserts *consistently*, and the 7^th rule maps *trvale* to *have*. Better solution would be either to map *trvale* to *consistently*, and to insert *have*. When inspecting the reasons that caused this mismatch, we found that the entry-translation pair *trvale-consistently* was not found in the translation dictionary. Here is our interpretation of what happened: The relation of leaf nodes was 3 (nodes 1, 2, and 6) to 5 (nodes 1, 2, 3, 4, 11). Since the Viterbi probabilities of synchronous substitution at (1 - 1) and at(6 - 11) gave a good score (see the chart in Table A.5) the Viterbi alignment had to solve 4 nodes, which were lexically unmatched: *trvale*, *consistently*, *have*, and *be*. It "had to pay" deletion of one of the nodes *have*, *be* or *consistently* anyway, and to align the remaining 2 nodes either as internal – within the same rule – or as one substitution by 1-1 synchronous rule. Since there were no other clues, it wrongly selected *consistently* for deletion.

If we had a better lexical rescoring that would prefer the deletion of *be* and *have* (e.g. using some fertility table), then the problem could have been fixed. Also a better modeling of the mapping of frontier nodes that would for example prefer the pair THL-Adv would help.

For comparison, the alignment between the analytical trees for the same sentence pair on Figure A.10 is correct.[8] The reason is that the ratio of leaves is 4 (1, 2, 7, 8) to 5 (1, 2, 3, 4, 11), and that there are 4 pairs of frontier nodes with good Viterbi scores. Thus there is no need to delete node *be* by (0-1) rule, it can be done automatically. The Viterbi alignment decided to align *trvale* with *consistently* because of the back-off model **w/o LEXICAL**, which contains rule ($Adv \rightarrow$ _)($Adv \rightarrow$ _).

Since the purpose of these experiments was to show that the model of synchronous tree-to-tree transductions can be implemented, we consider this method of evaluation appropriate. On the other hand, we are very much aware of the importance of a quantitative evaluation and we discuss its possibilities in Section 5.10.

5.10 Proposed Further Directions of Research

The idea of $STSG$ is unexplored, these experiments have shown promising results, but they still have to be considered preliminary. One of the reasons is that the implementation of the $STSG$ computational framework is very basic, and that many other experiments still have to be carried out. Another reason is that this approach was not yet applied to some practical problem. In this section we

[8] We made sure that there is no pair *trvale - consistently* in the translation dictionary as in the previous case of the TR - AR alignment.

list both—possible ways that could improve this method, and a list of areas where using $STSG$ could help.

5.10.1 Preprocessing the Input

When inspecting the Viterbi alignments, we have realized that many of the wrong alignments were caused by an *error in the translation dictionary*. Most of these errors come from the input that was not normalized enough, such as numbers, symbols, or proper names. We believe that most of these errors can be fixed by an improved *preprocessing of the input*, *normalization* and *canonicalization*.

Another problem is the *punctuation*. As the first, punctuation increases the number of child nodes, and most of the tree pairs were excluded from the training data just because of the punctuation. Moreover, punctuation also increases the data sparseness.[9] Therefore some special handling of punctuation is unavoidable.

5.10.2 Quantitative Evaluation of Results

One of the drawbacks of the evaluation based on Viterbi alignments is that it is very difficult to implement any quantitative evaluation method. Nevertheless, we can see two ways to achieve this.

One possibility is to manually annotate the data (add links between nodes) and to impose a heuristic scoring function that would evaluate the matching little trees.[10] The drawback of this approach is that it is hard to define the border between the good rules and bad rules. Imposing a scoring function without having in mind could easily result in misleading interpretations of results.

Another way is to apply the $STSG$ approach on some existing problem and the evaluation metrics for that problem. For example to implement a new MT system, or improve some part of it, and to use BLEU score.

5.10.3 Improving the Back-off Scheme

The data is so sparse that even the most generalized back-off model **STRUCTURE** does not stop growing after $30,000$ sentence pairs, as Figure 5.2 shows. It is necessary to develop new methods approximating probabilities of unseen rules based on similarity with known rules. These methods can be either statistical or heuristic.

[9] For example, two clauses separated by dash form one rule, and the same two clauses separated by colon form a different rule

[10] Again, here we could reuse the links between tectogrammatical and analytical nodes in the PDT.

5.10.4 Filtering out Low-confidence Matches

Since the shape of little trees that we take into consideration is limited, it may happen very often that the correct alignment does not exist because the structure of both trees is so different. Table A.5 shows that wrong matches have considerably lower Viterbi score. It would be useful (and hopefully not very difficult) to implement a confidence scoring method based on Viterbi probabilities in the chart. The confidence score could be used to detect that the aligning procedure experiences problems on a given tree pair, or on a given subtree. These pairs of trees or their subtrees could be excluded from the training.

The pairs of trees or subtrees where the alignment fails could be the hint for extending the set of allowed shapes of little trees.

5.10.5 Integration of Manually Defined Rules

This approach also opens a space for combining manually defined rules with the statistically extracted ones. These manually defined rules will have the same form as the synchronous rules. It is possible either to store them into the scoring model and to use them during the training process, or to use them during the decoding.

Since the number of the manually defined rules will be is reasonable, it is not necessary to apply the restrictions used for the statistically extracted rules. The manual rules can have more internal nodes and thus they can handle more complex structural changes.

5.10.6 Training on Plain Text

The experiments described above use PCEDT as training data. The Czech trees are created fully automatically from the plain text, but the English part is converted from the structurally annotated Penn Treebank.

The next step would be to use a statistical parser of English, create the English analytical trees automatically, and to train the alignment on plain text. This would open a perspective for training on more data, such as on the Czech-English Reader's Digest Corpus and Prague Tribune.

5.10.7 Decoding

The main motivation of the research in $STSG$s was to build an MT system. The Algorithm 4.9 describes the decoder, which is the heart of such a system. Line 4 of the decoding algorithm is crucial. It supposes a mechanism which *proposes hypotheses* of synchronous rules.

Let's think of the functionality of such a proposer. It has to offer all possible rules such that the left little tree is t_1. The easiest part of the task is to offer hy-

potheses that have already been seen. They can be derived from the probabilistic model model. But this does not suffice at all, since the data is sparse and the good rule will be most probably unknown. Moreover, as we know from the experiments with alignment, it is not realistic to keep rules with full lexical information in the model because of the memory limitations. Thus we have to come up with a mechanism that overgenerates the list of possible rules based on some heuristics.

5.10.8 Aligning Templates

The technique of *aligning templates* model was introduced by [Och, 2002], it is also referenced as *phrase-based* model, e.g. in [Koehn et al., 2003]. Apart from the standard IBM models that allow 1-n mappings only, the aligning templates allow for m-n mappings. In another words, they try to find phrasal translations – subsequences of possibly more than one word in both sentences. Most often used approach starts with modeling the 1-m translations in both directions. The resulting alignments are then combined, using various methods based on intersection and union of both unidirectional mappings.

We see one of the possible advantages of using the tree-to-tree mappings in improving the training phase of aligning templates.

The method of aligning templates is based on modeling transformations of strings of words. The integration of tree-to-tree mappings would require parsing of both input streams, and we believe that the knowledge of the sentence structure and its alignment would improve the quality of the extracted aligning templates.

5.11 Conclusion

The goal of the experiments described in this chapter was to show that it is possible to implement and train statistical model of $STSG$. Three experiments with extracting alignment have been implemented: Czech tectogrammatical-to-analytical alignment, Czech tectogrammatical to English analytical alignment, and Czech-English analytical alignment. All of these experiments have shown promising results, since most of the alignments are correct. On the other hand, the results still have to be considered preliminary. A method of quantitative evaluation is missing, but we have discussed the possible implementations of it. The space for improvements is huge, several ways of further research were suggested as well as a candidates where the application of the new approach could gain some improvement.

Chapter 6

Conclusions

Let us briefly summarize the most important contributions of this work to the body of research in the field of machine translation.

- We have implemented, and—to our knowledge—for the first time published the mathematical details of a new method for learning non-isomorphic tree-to-tree transformations based on probabilistic Synchronous Tree Substitution Grammars [Eisner, 2003, Hajič et al., 2002].
- We have applied this new method in three configurations: on Czech tectogrammatical-to-analytical tree pairs, Czech tectogrammatical to English analytical tree pairs, and on Czech-English analytical tree pairs.
- We have presented tree-to-tree alignments resulting from all these three implementations. Although the results are still preliminary, they can be considered very promising and there is a hope that they could improve some existing methods, such as the approach of aligning templates [Och, 2002] used for decoding.
- We propose several directions of further research in order to improve the method, as well as towards the implementation of the decoder, which is necessary for a full-scale machine translation system based on the presented new method.
- In order to make these experiments possible, it was necessary to build a large parallel corpus of Czech and English trees. The author has made significant contribution to the **Prague Czech-English Dependency Treebank**, especially on conversions of Penn Treebank format into analytical representation, and the integration of existing techniques of the automatic analysis of Czech for building the Czech part of the PCEDT.
- A baseline rule-based system for Czech-English machine translation. The system uses a statistical parser and rule-based conversions to obtain the Czech tectogrammatical representation of the sentence, then applies a set

of rules performing the transfer and the generation into the surface English sentence.

- Almost all partial results except Chapters 4 and 5 have been published at well recognized conferences abroad. For example, the work related to the PCEDT corpus was published as [Čmejrek et al., 2004a, Čmejrek et al., 2005, Cuřín et al., 2004a], the and the rule-based MT system has been published as [Čmejrek et al., 2003a].

We hope that the ideas sketched in this work will be further developed and bring some improvement in the field of machine translation.

Appendix A

Examples of Tree-to-Tree Alignments

A.1 Tree-to-Tree Alignment between Tectogrammatical and Analytical Representations of Czech

Results of the automatic alignment between Czech tectogrammatical and analytical trees are presented here. For each sentence pair the presentation contains the original tree pair, the Viterbi alignment, and the computational chart.

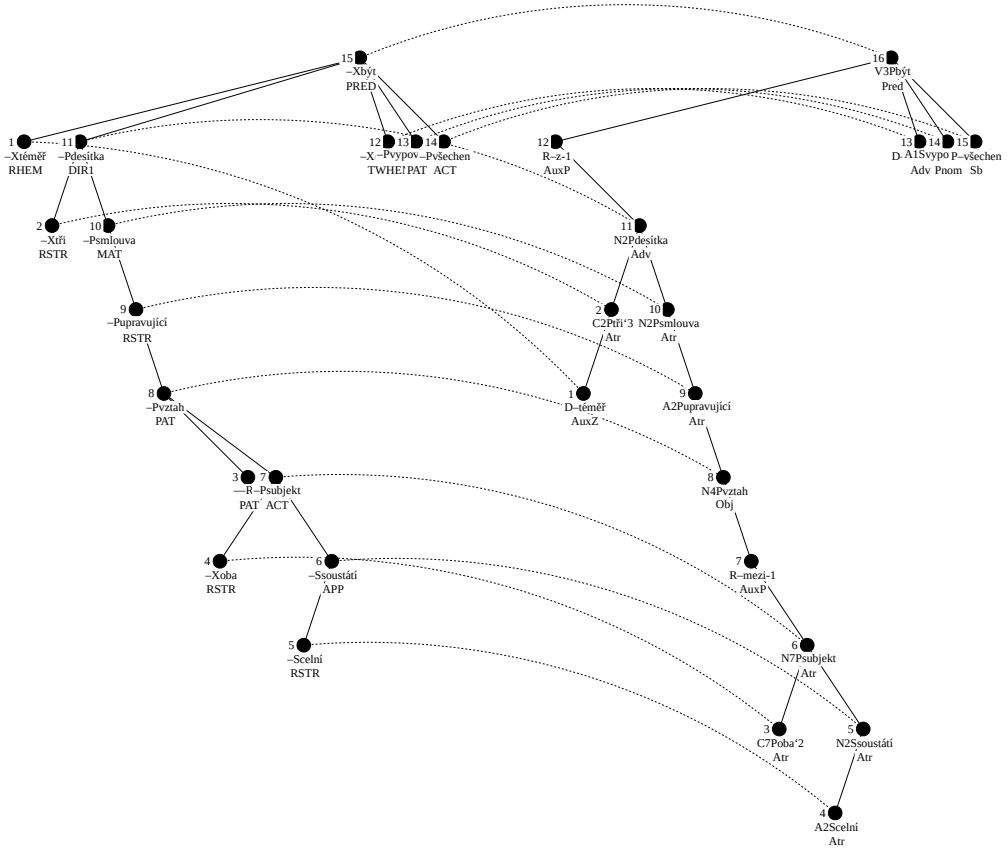


Figure A.1: A pair of tectogrammatical and analytical trees for Czech sentence *"Z téměř tří desítek smluv upravujících vztahy mezi oběma subjekty celního soustátí jsou okamžitě vypověditelné všechny"*.

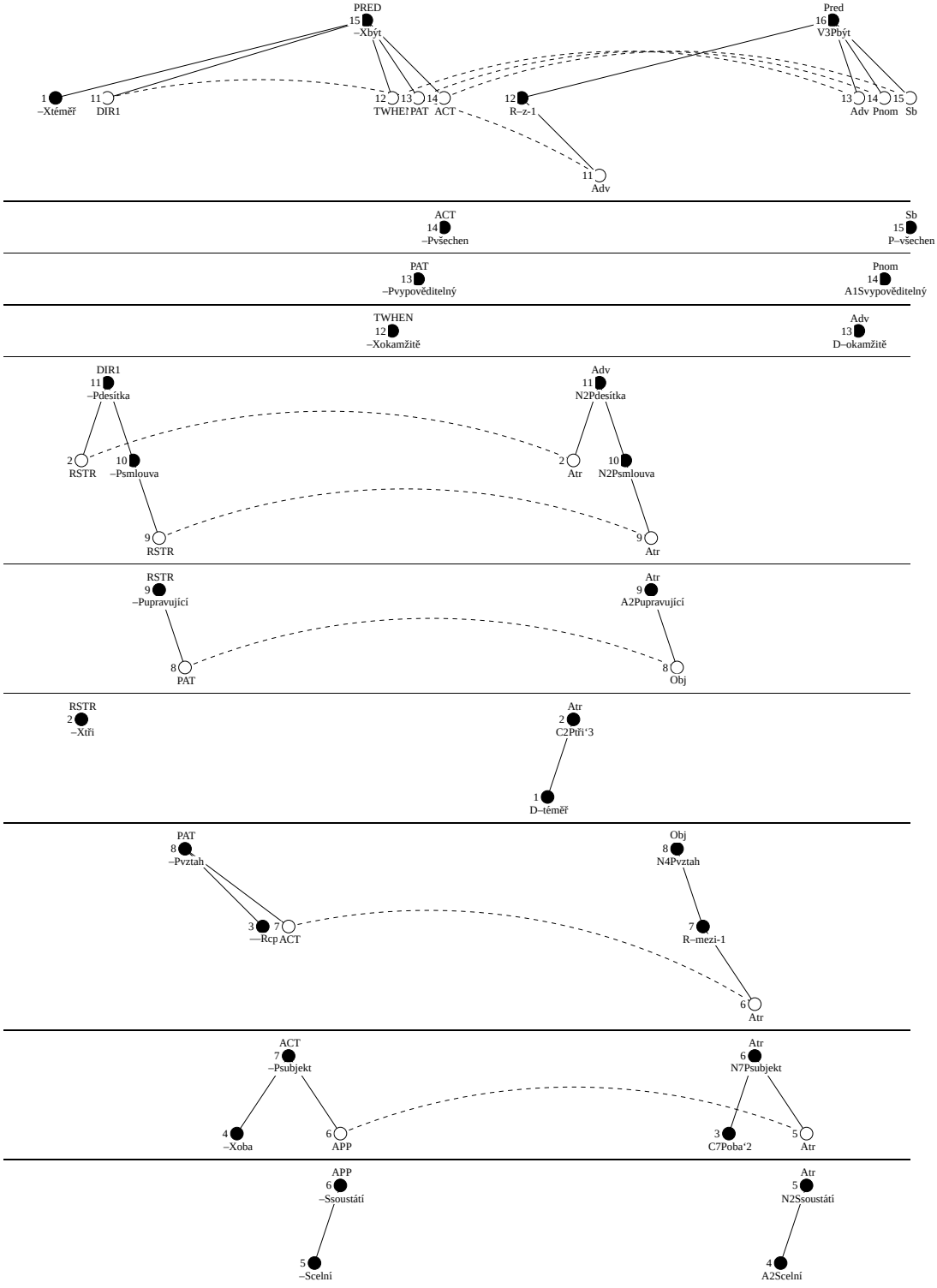


Figure A.2: Viterbi alignment of little trees for a pair of tectogrammatical and analytical trees for the Czech sentence *“Z téměř tří desítek smluv upravujících vztahy mezi oběma subjekty celního soustátí jsou okamžitě vypověditelné všechny”*.

nodes		inside	outside	Viterbi	rule
2	0	-1.7665	-18.421	-4.539	(RSTR→(I–Xtři))(NullSyntFunc→(FNullSyntFunc))()
2	1	-4.539	-11.897	-4.539	(RSTR→(I–Xtři))(AuxZ→(ID–téměř))()
2	2	-3.265	-9.750	-3.2659	**(RSTR→(I–Xtři))(Atr→((ID–téměř)IC2Ptři‘3))()**
2	3	-4.539	-44.255	-4.539	(RSTR→(I–Xtři))(Atr→(IC7Poba‘2))()
2	4	-4.539	-50.82	-4.539	(RSTR→(I–Xtři))(Atr→(IA2Scelní))()
2	5	-5.919	-47.032	-5.919	(RSTR→(I–Xtři))(Atr→((IA2Scelní)IN2Ssoustátí))()
2	6	-15.638	-36.595	-16.737	(RSTR→(FRSTR))(Atr→((IC7Poba‘2)IN7Psubjekt (FAtr)))((1-1))
6	1	-9.212	-41.872	-9.212	(APP→((I–Scelní)I–Ssoustátí))(AuxZ→(ID–téměř))()
6	2	-10.80	-40.702	-10.811	(APP→((I–Scelní)I–Ssoustátí))(Atr→((ID–téměř)IC2Ptři‘3))()
6	3	-10.812	-19.528	-10.814	(APP→((I–Scelní)I–Ssoustátí))(Atr→(IC7Poba‘2))()
6	4	-2.2163	-15.904	-2.216	(APP→((I–Scelní)I–Ssoustátí))(Atr→(IA2Scelní))()
6	5	-0.9114	-12.388	-1.1750	**(APP→((I–Scelní)I–Ssoustátí))(Atr→((IA2Scelní)IN2Ssoustátí))()**
6	6	-9.81	-11.858	-11.811	(APP→((FRSTR)I–Ssoustátí))(Atr→((FAtr)IN7Psubjekt ((FAtr)IN2Ssoustátí)))((0-1)(1-2))
6	7	-14.897	-11.87	-17.41	(APP→(FAPP))(AuxP→(IR–mezi-1 ((FAtr)IN7Psubjekt (FAtr))))((0-1)(1-2))
7	2	-13.76	-37.17	-15.263	(ACT→((I–Xoba)I–Psubjekt (FAPP)))(Atr→((FAuxZ)IC2Ptři‘3))((1-1))
7	3	-12.218	-16.518	-15.003	(ACT→((I–Xoba)I–Psubjekt (FAPP)))(Atr→(IC7Poba‘2))((1-0))
7	4	-8.257	-21.17	-8.266	(ACT→((I–Xoba)I–Psubjekt (FAPP)))(Atr→(FAtr))((1-1))
7	5	-4.184	-15.824	-7.042	(ACT→((FRSTR)I–Psubjekt ((FRSTR)I–Ssoustátí)))(Atr→((FAtr)IN2Ssoustátí))((2-1)(1-0))
7	6	-3.735	-9.334	-4.460	**(ACT→((I–Xoba)I–Psubjekt (FAPP)))(Atr→((IC7Poba‘2)IN7Psubjekt (FAtr)))((1-1))**
7	7	-7.1518	-8.62	-7.734	(ACT→((FRSTR)I–Psubjekt (FAPP)))(AuxP→(IR–mezi-1 ((FAtr)IN7Psubjekt (FAtr))))((2-2)(1-1))
7	8	-10.531	-8.059	-11.264	(ACT→(FACT))(Obj→(IN4Pvztah (IR–mezi-1 (FAtr))))((1-1))
8	4	-7.414	-32.505	-9.827	(PAT→(I–Pvztah (I—Rcp)(FACT)))(Atr→(FAtr))((1-1))
8	5	-5.072	-23.8	-8.603	(PAT→(I–Pvztah (I—Rcp)(FACT)))(Atr→(FAtr))((1-1))
8	6	-3.963	-15.234	-6.021	(PAT→(I–Pvztah (I—Rcp)(FACT)))(Atr→(FAtr))((1-1))
8	7	-4.73	-10.49	-6.016	(PAT→(I–Pvztah (I—Rcp)(FACT)))(AuxP→(IR–mezi-1 (FAtr)))((1-1))
8	8	-4.476	-8.713	-5.800	**(PAT→(I–Pvztah (I—Rcp)(FACT)))(Obj→(IN4Pvztah (IR–mezi-1 (FAtr))))((1-1))**
8	9	-7.918	-10.579	-9.078	(PAT→(I–Pvztah (I—Rcp)(FACT)))(Atr→(IA2Pupravující (IN4Pvztah (FAuxP))))((1-1))
8	10	-11.91	-12.299	-12.825	(PAT→(I–Pvztah (I—Rcp)(FACT)))(Atr→(IN2Psmlouva (IA2Pupravující (FObj))))((1-1))
9	4	-11.631	-34.58	-14.464	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Atr→(FAtr))((2-1)(1-0))
9	5	-7.591	-27.139	-13.240	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Atr→(FAtr))((2-1)(1-0))
9	6	-6.641	-17.350	-10.658	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Atr→(FAtr))((2-1)(1-0))
9	7	-6.966	-14.246	-10.556	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(AuxP→(IR–mezi-1 (FAtr)))((2-1)(1-0))
9	8	-5.316	-10.581	-8.850	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Obj→(IN4Pvztah (IR–mezi-1 (FAtr))))((2-1)(1-0))
9	9	-5.361	-9.23	-7.411	**(RSTR→(I–Pupravující (FPAT)))(Atr→(IA2Pupravující (FObj)))((1-1))**
9	10	-6.079	-8.10	-7.41	(RSTR→(I–Pupravující (FPAT)))(Atr→(IN2Psmlouva (IA2Pupravující (FObj))))((1-1))
9	11	-23.0	-7.819	-24.750	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Adv→((FAtr)IN2Pdesítka (IN2Psmlouva (FAtr))))((2-2)(1-1))
9	12	-19.573	-8.280	-23.107	(RSTR→(FRSTR))(AuxP→(IR–z-1 ((FAtr)IN2Pdesítka (FAtr))))((0-1)(1-2))
9	13	-14.613	-54.17	-23.062	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Adv→(FAdv))((2-1)(1-0))
9	14	-14.614	-52.70	-23.062	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Pnom→(FPnom))((2-1)(1-0))
9	15	-14.602	-53.95	-22.878	(RSTR→(I–Pupravující (I–Pvztah (FPAT)(FACT))))(Sb→(FSb))((2-0)(1-1))
11	7	-10.550	-34.29	-14.359	(DIR1→((I–Xtři)I–Pdesítka (FMAT)))(AuxP→(IR–mezi-1 (FAtr)))((1-1))
11	8	-9.706	-26.785	-14.220	(DIR1→((I–Xtři)I–Pdesítka (FMAT)))(Obj→(FObj))((1-1))
11	9	-8.912	-23.523	-11.696	(DIR1→((I–Xtři)I–Pdesítka (FMAT)))(Atr→(FAtr))((1-1))
11	10	-8.034	-15.968	-11.389	(DIR1→((I–Xtři)I–Pdesítka (FMAT)))(Atr→(FAtr))((1-1))
11	11	-9.342	-4.637	-12.309	**(DIR1→((FRSTR)I–Pdesítka (I–Psmlouva (FRSTR))))(Adv→((FAtr)IN2Pdesítka (IN2Psmlouva (FAtr))))((2-2)(1-1))**
11	12	-10.419	-2.98	-12.516	(DIR1→((FRSTR)I–Pdesítka (FMAT)))(AuxP→(IR–z-1 ((FAtr)IN2Pdesítka (FAtr))))((2-2)(1-1))
11	13	-14.7	-48.86	-26.790	(DIR1→((I–Xtři)I–Pdesítka (FMAT)))(Adv→(FAdv))((1-1))
11	14	-14.462	-47.6	-25.58	(DIR1→((I–Xtři)I–Pdesítka (FMAT)))(Pnom→(FPnom))((1-1))
12	8	-23.918	-40.609	-28.43	(TWHEN→(FTWHEN))(Obj→(IN4Pvztah (IR–mezi-1 (FAtr))))((1-1))
12	9	-30.912	-39.9	-37.86	(TWHEN→(I–Xokamžitě))(Atr→(IA2Pupravující (IN4Pvztah (FAuxP))))((0-1))
12	10	-33.12	-29.79	-39.25	(TWHEN→(I–Xokamžitě))(Atr→(IN2Psmlouva (IA2Pupravující (FObj))))((0-1))
12	11	-48.19	-21.68	-59.50	(TWHEN→(I–Xokamžitě))(Adv→((FAtr)IN2Pdesítka (IN2Psmlouva (FAtr))))((0-2)(0-1))
12	12	-44.89	-16.804	-54.947	(TWHEN→(FTWHEN))(AuxP→(IR–z-1 ((FAtr)IN2Pdesítka (FAtr))))((0-1)(1-2))
12	13	-0.001	-13.059	-0.001	**(TWHEN→(I–Xokamžitě))(Adv→(ID–okamžitě))()**
12	14	-10.815	-15.717	-10.816	(TWHEN→(I–Xokamžitě))(Pnom→(IA1Svypovědítelný))()
12	15	-10.814	-17.01	-10.816	(TWHEN→(I–Xokamžitě))(Sb→(IP–všechen))()
12	16	-57.81	-24.61	-71.23	(TWHEN→(FTWHEN))(Pred→((FAuxP)IV3Pbýt (FAdv)(IA1Svypovědítelný)(FSb)))((0-3)(0-1)(1-2))
13	9	-30.707	-38.37	-37.86	(PAT→(I–Pvypovědítelný))(Atr→(IA2Pupravující (IN4Pvztah (FAuxP))))((0-1))
13	10	-32.91	-28.25	-39.25	(PAT→(I–Pvypovědítelný))(Atr→(IN2Psmlouva (IA2Pupravující (FObj))))((0-1))
13	11	-47.85	-20.14	-59.50	(PAT→(I–Pvypovědítelný))(Adv→((FAtr)IN2Pdesítka (IN2Psmlouva (FAtr))))((0-2)(0-1))
13	12	-44.5	-15.262	-54.947	(PAT→(FPAT))(AuxP→(IR–z-1 ((FAtr)IN2Pdesítka (FAtr))))((0-1)(1-2))
13	13	-10.815	-15.717	-10.816	(PAT→(I–Pvypovědítelný))(Adv→(ID–okamžitě))()
13	14	-1.6103	-11.863	-1.6103	**(PAT→(I–Pvypovědítelný))(Pnom→(IA1Svypovědítelný))()**
13	15	-10.670	-15.408	-10.673	(PAT→(I–Pvypovědítelný))(Sb→(IP–všechen))()
13	16	-58.033	-24.02	-72.84	(PAT→(FPAT))(Pred→((FAuxP)IV3Pbýt (ID–okamžitě)(FPnom)(FSb)))((0-3)(0-1)(1-2))
14	11	-47.52	-20.68	-59.50	(ACT→(I–Pvšechen))(Adv→((FAtr)IN2Pdesítka (IN2Psmlouva (FAtr))))((0-2)(0-1))
14	12	-44.222	-16.49	-54.947	(ACT→(FACT))(AuxP→(IR–z-1 ((FAtr)IN2Pdesítka (FAtr))))((0-1)(1-2))
14	13	-10.599	-17.018	-10.600	(ACT→(I–Pvšechen))(Adv→(ID–okamžitě))()
14	14	-10.815	-15.40	-10.816	(ACT→(I–Pvšechen))(Pnom→(IA1Svypovědítelný))()
14	15	-0.3096	-12.781	-0.30968	**(ACT→(I–Pvšechen))(Sb→(IP–všechen))()**
14	16	-57.30	-24.433	-71.7	(ACT→(FACT))(Pred→((FAuxP)IV3Pbýt (FAdv)(IA1Svypovědítelný)(FSb)))((0-2)(0-1)(1-3))
15	0	-28.132	-62.59	-54.34	(PRED→((FRHEM)(FDIR1)I–Xbýt (FTWHEN)(I–Pvypovědítelný)(FACT)))(NullSyntFunc→(FNullSyntFunc))((4-0)(3-0)(2-0)(1-0))
15	1	-29.658	-65.66	-50.66	(PRED→((FRHEM)(FDIR1)I–Xbýt (FTWHEN)(I–Pvypovědítelný)(FACT)))(AuxZ→(FAuxZ))((4-0)(3-0)(2-0)(1-1))
15	13	-24.611	-56.44	-45.13	(PRED→((FRHEM)(FDIR1)I–Xbýt (FTWHEN)(I–Pvypovědítelný)(FACT)))(Adv→(FAdv))((4-0)(3-1)(2-0)(1-0))
15	14	-25.57	-56.03	-46.51	(PRED→((FRHEM)(FDIR1)I–Xbýt (FTWHEN)(FPAT)(I–Pvšechen)))(Pnom→(FPnom))((4-1)(3-0)(2-0)(1-0))
15	15	-24.736	-56.61	-45.21	(PRED→((FRHEM)(FDIR1)I–Xbýt (FTWHEN)(I–Pvypovědítelný)(FACT)))(Sb→(FSb))((4-1)(3-0)(2-0)(1-0))
15	16	-12.97	0.0	-17.20	**(PRED→((I–Xtéměř)(FDIR1)I–Xbýt (FTWHEN)(FPAT)(FACT)))(Pred→((IR–z-1 (FAdv))IV3Pbýt (FAdv)(FPnom)(FSb)))((4-3)(3-2)(2-1)(1-4))**

Table A.1: Computational chart with Viterbi probabilities for a pair of tectogrammatical and analytical trees for the Czech sentence *“Z téměř tří desítek smluv upravujících vztahy mezi oběma subjekty celního soustátí jsou okamžitě vypověditelné všechny”*.

A.2 Tree-to-Tree Alignment between Czech Tectogrammatical and English Analytical Representations

Results of the automatic alignment between Czech tectogrammatical and English analytical trees are presented here. The tree structures are results of an automatic annotation procedures, thus may contain errors. For each sentence pair the presentation contains the original tree pair, the Viterbi alignment and the computational chart.

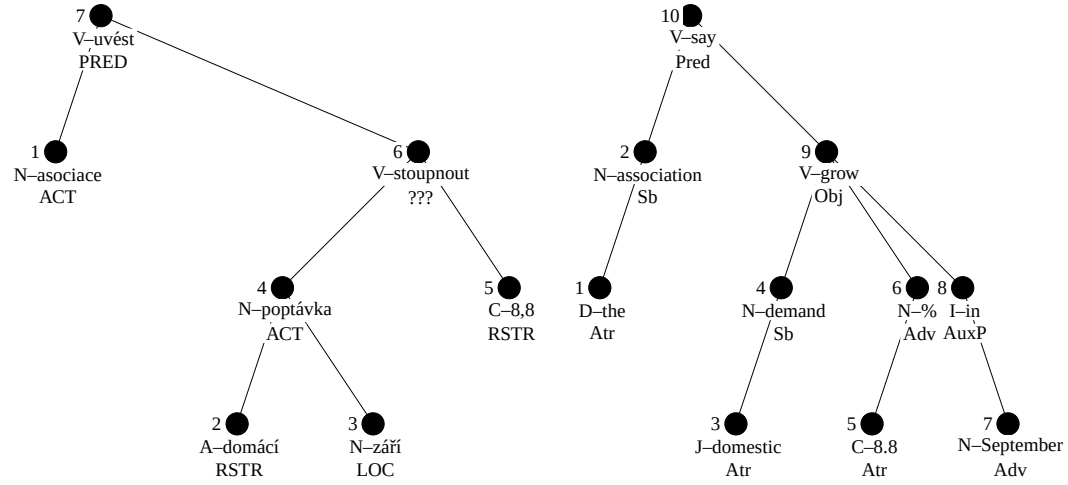


Figure A.3: A tree pair for Czech sentence *"Asociace uvedla, že domácí poptávka v září stoupla o 8,8 %."* and English sentence *"The association said domestic demand grew 8.8% in September."*

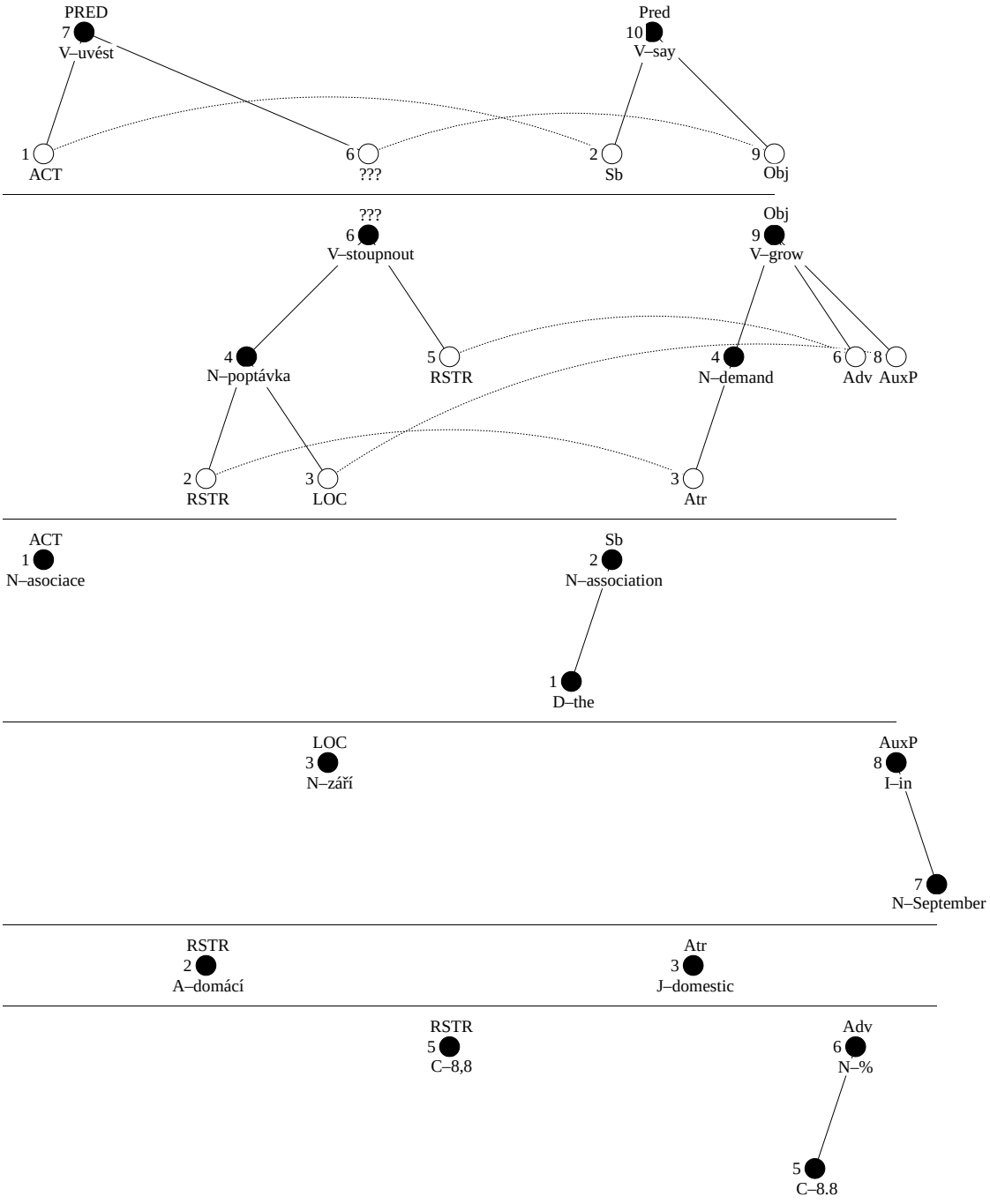


Figure A.4: Viterbi alignment of little trees for sentence pair "*Asociace uvedla, že domácí poptávka v září stoupla o 8,8 %.*" and "*The association said domestic demand grew 8.8% in September.*"

nodes		inside	outside	Viterbi	rule
0	1	-7.482	-5.011	-9.428	(NullSyntFunc→(FNullSyntFunc))(Atr→(ID–the))()
0	2	-7.557	-5.399	-9.504	(NullSyntFunc→(FNullSyntFunc))(Sb→((ID–the)IN–association))()
0	3	-7.467	-14.225	-9.413	(NullSyntFunc→(FNullSyntFunc))(Atr→(IJ–domestic))()
0	4	-7.733	-13.948	-9.679	(NullSyntFunc→(FNullSyntFunc))(Sb→((IJ–domestic)IN–demand))()
0	5	-7.413	-17.15	-9.359	(NullSyntFunc→(FNullSyntFunc))(Atr→(IC–8.8))()
0	6	-8.662	-13.847	-10.60	(NullSyntFunc→(FNullSyntFunc))(Adv→((IC–8.8)IN–%))()
0	7	-7.545	-12.203	-9.49	(NullSyntFunc→(FNullSyntFunc))(Adv→(IN–September))()
0	8	-7.513	-7.409	-9.459	(NullSyntFunc→(FNullSyntFunc))(AuxP→(II–in (IN–September)))()
0	9	-31.989	-25.478	-39.316	(NullSyntFunc→(FNullSyntFunc))(Obj→((FSb)IV–grow ((FAtr)IN–%)(FAuxP)))((0-3)(0-2)(0-1))
0	10	-41.61	-33.99	-50.069	(NullSyntFunc→(FNullSyntFunc))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-4)(0-3)(0-2)(0-1))
1	0	-7.129	-12.185	-9.432	(ACT→(IN–asociace))(NullSyntFunc→(FNullSyntFunc))()
1	1	-10.709	-5.400	-10.710	(ACT→(IN–asociace))(Atr→(ID–the))()
1	2	-2.704	-5.198	-2.704	**(ACT→(IN–asociace))(Sb→((ID–the)IN–association))()**
1	3	-10.733	-29.64	-10.733	(ACT→(IN–asociace))(Atr→(IJ–domestic))()
1	4	-10.812	-18.90	-10.812	(ACT→(IN–asociace))(Sb→((IJ–domestic)IN–demand))()
1	5	-10.7	-31.239	-10.734	(ACT→(IN–asociace))(Atr→(IC–8.8))()
1	6	-9.315	-20.459	-9.316	(ACT→(IN–asociace))(Adv→((IC–8.8)IN–%))()
1	7	-10.801	-23.024	-10.802	(ACT→(IN–asociace))(Adv→(IN–September))()
1	8	-9.57	-12.283	-9.578	(ACT→(IN–asociace))(AuxP→(II–in (IN–September)))()
1	9	-37.2	-24.403	-39.006	(ACT→(FACT))(Obj→(((FAtr)IN–demand)IV–grow (FAdv)(FAuxP)))((0-3)(0-2)(1-1))
1	10	-43.262	-32.49	-43.27	(ACT→(FACT))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-4)(0-3)(0-2)(1-1))
2	0	-7.000	-13.423	-9.303	(RSTR→(IA–domácí))(NullSyntFunc→(FNullSyntFunc))()
2	1	-9.430	-30.07	-9.430	(RSTR→(IA–domácí))(Atr→(ID–the))()
2	2	-9.428	-24.818	-9.428	(RSTR→(IA–domácí))(Sb→((ID–the)IN–association))()
2	3	-0.7243	-7.261	-0.7243	**(RSTR→(IA–domácí))(Atr→(IJ–domestic))()**
2	4	-0.7385	-9.522	-0.7385	(RSTR→(IA–domácí))(Sb→((IJ–domestic)IN–demand))()
2	5	-9.43	-19.427	-9.430	(RSTR→(IA–domácí))(Atr→(IC–8.8))()
2	6	-9.87	-16.098	-9.874	(RSTR→(IA–domácí))(Adv→((IC–8.8)IN–%))()
2	7	-10.80	-20.509	-10.80	(RSTR→(IA–domácí))(Adv→(IN–September))()
2	8	-9.205	-15.897	-9.205	(RSTR→(IA–domácí))(AuxP→(II–in (IN–September)))()
2	9	-28.370	-22.387	-30.37	(RSTR→(FRSTR))(Obj→((FSb)IV–grow ((FAtr)IN–%)(FAuxP)))((0-3)(0-2)(1-1))
2	10	-39.047	-32.38	-41.12	(RSTR→(FRSTR))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-4)(0-3)(0-1)(1-2))
3	0	-6.9072	-13.149	-9.209	(LOC→(IN–září))(NullSyntFunc→(FNullSyntFunc))()
3	1	-9.43	-30.074	-9.43	(LOC→(IN–září))(Atr→(ID–the))()
3	2	-10.8	-24.573	-10.814	(LOC→(IN–září))(Sb→((ID–the)IN–association))()
3	3	-9.430	-13.995	-9.430	(LOC→(IN–září))(Atr→(IJ–domestic))()
3	4	-10.813	-18.019	-10.814	(LOC→(IN–září))(Sb→((IJ–domestic)IN–demand))()
3	5	-9.430	-19.39	-9.43	(LOC→(IN–září))(Atr→(IC–8.8))()
3	6	-10.813	-16.06	-10.814	(LOC→(IN–září))(Adv→((IC–8.8)IN–%))()
3	7	-1.7230	-10.50	-1.7230	(LOC→(IN–září))(Adv→(IN–September))()
3	8	-0.7648	-7.136	-0.7648	**(LOC→(IN–září))(AuxP→(II–in (IN–September)))()**
3	9	-27.852	-22.387	-30.62	(LOC→(FLOC))(Obj→((FSb)IV–grow ((FAtr)IN–%)(FAuxP)))((0-3)(0-1)(1-2))
3	10	-38.07	-32.384	-41.374	(LOC→(FLOC))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-3)(0-2)(0-1)(1-4))
4	0	-14.774	-19.844	-20.027	(ACT→((IA–domácí)IN–poptávka (FLOC)))(NullSyntFunc→(FNullSyntFunc))((1-0))
4	1	-16.992	-30.071	-20.02	(ACT→((IA–domácí)IN–poptávka (FLOC)))(Atr→(ID–the))((1-0))
4	2	-16.942	-19.266	-20.02	(ACT→((IA–domácí)IN–poptávka (FLOC)))(Sb→((ID–the)IN–association))((1-0))
4	3	-8.955	-17.282	-11.33	(ACT→((IA–domácí)IN–poptávka (FLOC)))(Atr→(IJ–domestic))((1-0))
4	4	-2.4574	-12.45	-2.464	(ACT→((FRSTR)IN–poptávka (IN–září)))(Sb→((FAtr)IN–demand))((1-1))
4	5	-16.992	-24.1	-20.02	(ACT→((IA–domácí)IN–poptávka (FLOC)))(Atr→(IC–8.8))((1-0))
4	6	-16.956	-20.69	-20.02	(ACT→((IA–domácí)IN–poptávka (FLOC)))(Adv→((IC–8.8)IN–%))((1-0))
4	7	-9.213	-17.502	-11.552	(ACT→((FRSTR)IN–poptávka (IN–září)))(Adv→(IN–September))((1-0))
4	8	-9.082	-12.676	-11.507	(ACT→((FRSTR)IN–poptávka (IN–září)))(AuxP→(II–in (IN–September)))((1-0))
4	9	-12.448	-11.571	-13.837	(ACT→((FRSTR)IN–poptávka (FLOC)))(Obj→(((FAtr)IN–demand)IV–grow (FAdv)(FAuxP))))((0-1)(2-2)(1-3))
4	10	-29.341	-21.602	-32.430	(ACT→((FRSTR)IN–poptávka (FLOC)))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-3)(0-1)(2-4)(1-2))
5	0	-7.124	-15.55	-9.427	(RSTR→(IC–8,8))(NullSyntFunc→(FNullSyntFunc))()
5	1	-9.430	-21.35	-9.430	(RSTR→(IC–8,8))(Atr→(ID–the))()
5	2	-9.428	-20.34	-9.428	(RSTR→(IC–8,8))(Sb→((ID–the)IN–association))()
5	3	-9.430	-16.600	-9.430	(RSTR→(IC–8,8))(Atr→(IJ–domestic))()
5	4	-9.428	-18.759	-9.428	(RSTR→(IC–8,8))(Sb→((IJ–domestic)IN–demand))()
5	5	-0.21480	-10.741	-0.21480	(RSTR→(IC–8,8))(Atr→(IC–8.8))()
5	6	-0.5326	-7.402	-0.5326	**(RSTR→(IC–8,8))(Adv→((IC–8.8)IN–%))()**
5	7	-10.809	-19.853	-10.810	(RSTR→(IC–8,8))(Adv→(IN–September))()
5	8	-9.42	-15.367	-9.428	(RSTR→(IC–8,8))(AuxP→(II–in (IN–September)))()
5	9	-25.73	-19.251	-30.171	(RSTR→(FRSTR))(Obj→((FSb)IV–grow ((FAtr)IN–%)(FAuxP)))((0-2)(0-1)(1-3))
5	10	-35.164	-32.64	-39.993	(RSTR→(FRSTR))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-4)(0-2)(0-1)(1-3))
6	0	-23.286	-28.194	-30.845	(???→((FACT)IV–stoupnout (IC–8,8)))(NullSyntFunc→(FNullSyntFunc))((1-0))
6	1	-25.48	-33.336	-30.842	(???→((FACT)IV–stoupnout (IC–8,8)))(Atr→(FAtr))((1-1))
6	2	-25.381	-25.15	-30.839	(???→((FACT)IV–stoupnout (IC–8,8)))(Sb→((FAtr)IN–association))((1-1))
6	3	-19.766	-30.66	-22.15	(???→((FACT)IV–stoupnout (IC–8,8)))(Atr→(FAtr))((1-1))
6	4	-13.233	-20.64	-13.281	(???→((FACT)IV–stoupnout (IC–8,8)))(Sb→(FSb))((1-1))
6	5	-16.384	-29.733	-21.63	(???→((FACT)IV–stoupnout (IC–8,8)))(Atr→(IC–8.8))((1-0))
6	6	-16.384	-19.718	-21.638	(???→((FACT)IV–stoupnout (IC–8,8)))(Adv→((IC–8.8)IN–%))((1-0))
6	7	-20.025	-30.881	-22.370	(???→((FACT)IV–stoupnout (IC–8,8)))(Adv→(FAdv))((1-1))
6	8	-19.2	-20.863	-22.325	(???→((FACT)IV–stoupnout (IC–8,8)))(AuxP→(FAuxP))((1-1))
6	9	-3.6251	-4.263	-3.7231	**(???→(((FRSTR)IN–poptávka (FLOC))IV–stoupnout (FRSTR)))(Obj→(((FAtr)IN–demand)IV–grow (FAdv)(FAuxP)))((3-2)(2-3)(1**
6	10	-14.721	-10.81	-16.516	(???→(((FRSTR)IN–poptávka (FLOC))IV–stoupnout (FRSTR)))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-1)(3-4 2)(1-3))
7	0	-31.797	-42.07	-41.663	(PRED→((IN–asociace)IV–uvést (F???)))(NullSyntFunc→(FNullSyntFunc))((1-0))
7	1	-33.99	-42.74	-41.66	(PRED→((IN–asociace)IV–uvést (F???)))(Atr→(FAtr))((1-1))
7	2	-25.887	-34.726	-33.549	(PRED→((IN–asociace)IV–uvést (F???)))(Sb→((FAtr)IN–association))((1-1))
7	3	-30.523	-44.245	-32.97	(PRED→((IN–asociace)IV–uvést (F???)))(Atr→(FAtr))((1-1))
7	4	-24.01	-34.55	-24.099	(PRED→((IN–asociace)IV–uvést (F???)))(Sb→(FSb))((1-1))
7	5	-27.196	-43.31	-32.455	(PRED→((IN–asociace)IV–uvést (F???)))(Atr→(FAtr))((1-1))
7	6	-26.503	-33.62	-32.45	(PRED→((IN–asociace)IV–uvést (F???)))(Adv→((FAtr)IN–%))((1-1))
7	7	-30.774	-44.46	-33.18	(PRED→((IN–asociace)IV–uvést (F???)))(Adv→(FAdv))((1-1))
7	8	-29.663	-34.771	-33.14	(PRED→((IN–asociace)IV–uvést (F???)))(AuxP→(FAuxP))((1-1))
7	9	-14.327	-17.644	-14.540	(PRED→((IN–asociace)IV–uvést (F???)))(Obj→(FObj))((1-1))
7	10	-7.8876	0.0	-8.002	**(PRED→((FACT)IV–uvést (F???)))(Pred→((FSb)IV–say (FObj)))((2-2)(1-1))**

Table A.2: Computational chart with Viterbi probabilities for sentence pair *"Asociace uvedla, že domácí poptávka v září stoupla o 8,8 %."* and *"The association said domestic demand grew 8.8% in September."*

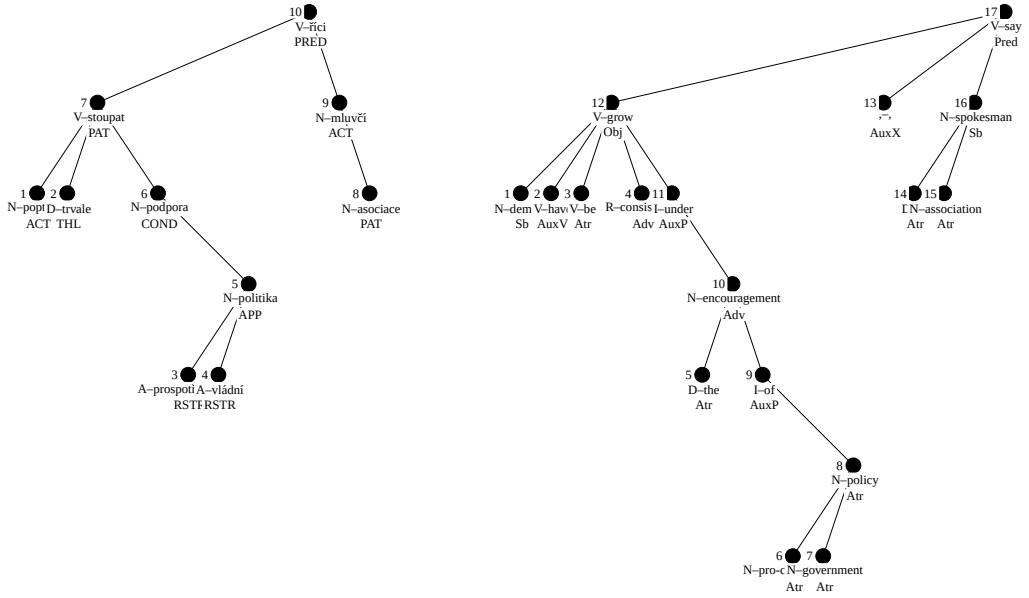


Figure A.5: A tree pair for Czech sentence *“Poptávka trvale stoupá za podpory prospotřebitelské vládní politiky, řekl mluvčí asociace.”* and English sentence *“Demand has been growing consistently under the encouragement of pro-consumption government policies, an association spokesman said.”*

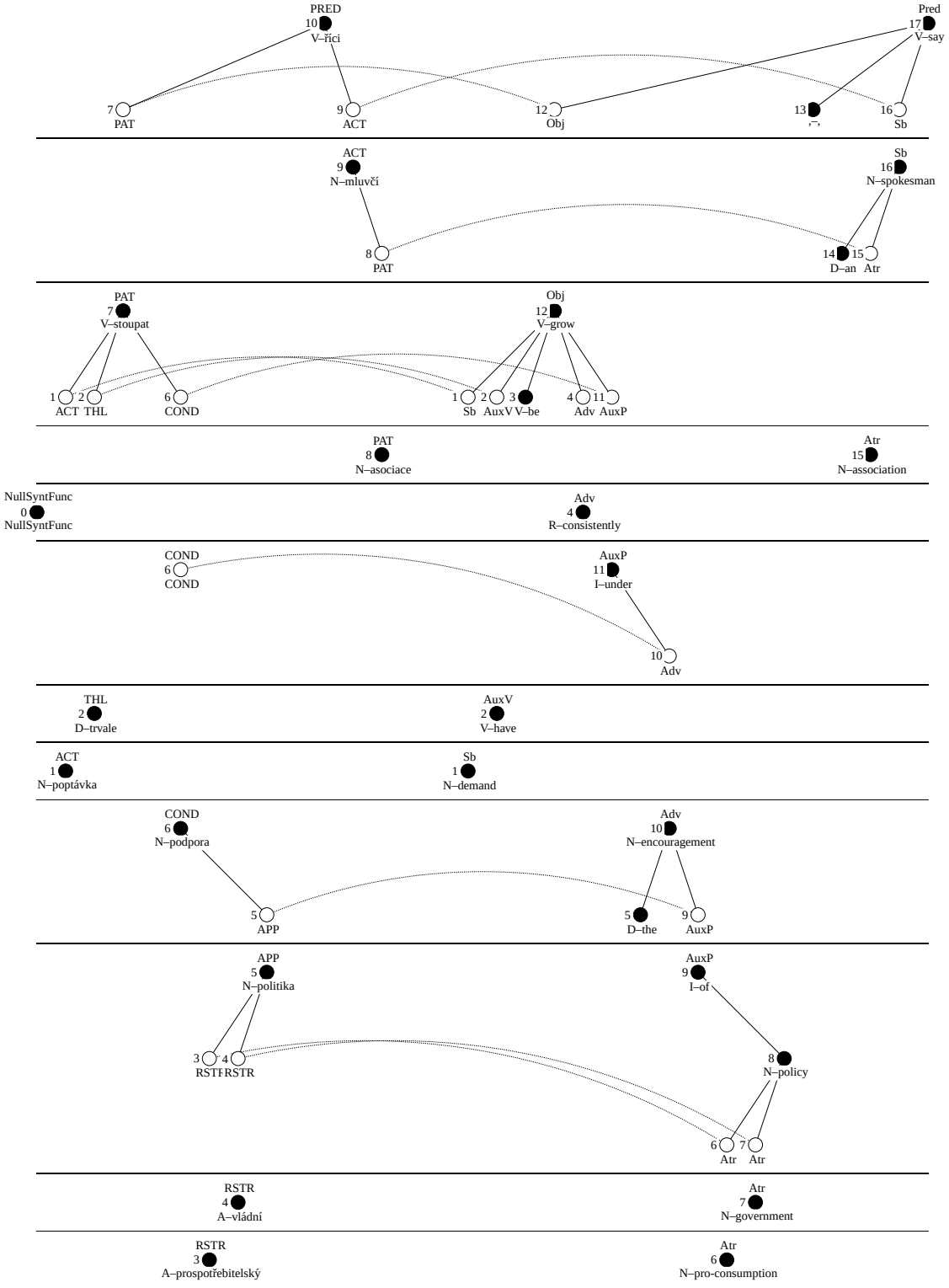


Figure A.6: Viterbi alignment of little trees for sentence pair "*Poptávka trvale stoupá za podpory prospotřebitelské vládní politiky, řekl mluvčí asociace.*" and "*Demand has been growing consistently under the encouragement of pro-consumption government policies, an association spokesman said.*"

0	1	-7.8105	-44.37	-10.113	(NullSyntFunc→(FNullSyntFunc))(Sb→(IN–demand))()
0	2	-7.129	-38.25	-9.432	(NullSyntFunc→(FNullSyntFunc))(AuxV→(IV–have))()
0	3	-7.133	-38.22	-9.436	(NullSyntFunc→(FNullSyntFunc))(Atr→(IV–be))()
0	4	-6.9695	-38.27	-9.272	**(NullSyntFunc→(FNullSyntFunc))(Adv→(IR–consistently))()**
0	5	-7.125	-37.96	-9.428	(NullSyntFunc→(FNullSyntFunc))(Atr→(ID–the))()
0	6	-7.002	-45.093	-9.305	(NullSyntFunc→(FNullSyntFunc))(Atr→(IN–pro-consumption))()
0	16	-15.481	-48.23	-20.12	(NullSyntFunc→(FNullSyntFunc))(Sb→((ID–an)(FAtr)IN–spokesman))((0-1))
0	17	-96.0	-51.42	-119.57	(NullSyntFunc→(FNullSyntFunc))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-2)(0-1))
1	0	-6.599	-49.253	-9.432	(ACT→(IN–poptávka))(NullSyntFunc→(FNullSyntFunc))()
1	1	-1.7390	-43.2	-1.7390	**(ACT→(IN–poptávka))(Sb→(IN–demand))()**
1	2	-9.440	-43.79	-9.44	(ACT→(IN–poptávka))(AuxV→(IV–have))()
1	3	-10.734	-43.76	-10.734	(ACT→(IN–poptávka))(Atr→(IV–be))()
1	4	-10.80	-43.83	-10.802	(ACT→(IN–poptávka))(Adv→(IR–consistently))()
1	17	-108.32	-45.888	-110.4	(ACT→(FACT))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-2)(1-1))
2	0	-6.595	-44.82	-9.429	(THL→(ID–trvale))(NullSyntFunc→(FNullSyntFunc))()
2	1	-10.81	-43.34	-10.816	(THL→(ID–trvale))(Sb→(IN–demand))()
2	2	-9.207	-38.0	-9.207	**(THL→(ID–trvale))(AuxV→(IV–have))()**
2	3	-9.432	-38.0	-9.433	(THL→(ID–trvale))(Atr→(IV–be))()
2	4	-10.816	-38.12	-10.816	(THL→(ID–trvale))(Adv→(IR–consistently))()
2	5	-9.432	-45.77	-9.433	(THL→(ID–trvale))(Atr→(ID–the))()
3	3	-9.430	-66.31	-9.430	(RSTR→(IA–prospotřebitelský))(Atr→(IV–be))()
3	4	-10.810	-66.45	-10.810	(RSTR→(IA–prospotřebitelský))(Adv→(IR–consistently))()
3	5	-9.430	-53.17	-9.430	(RSTR→(IA–prospotřebitelský))(Atr→(ID–the))()
3	6	-0.20037	-44.47	-0.20037	**(RSTR→(IA–prospotřebitelský))(Atr→(IN–pro-consumption))()**
3	7	-9.406	-51.14	-9.406	(RSTR→(IA–prospotřebitelský))(Atr→(IN–government))()
3	8	-9.553	-51.83	-10.915	(RSTR→(IA–prospotřebitelský))(Atr→((IN–pro-consumption)(FAtr)IN–policy))((0-1))
3	9	-18.94	-44.20	-20.32	(RSTR→(FRSTR))(AuxP→(II–of ((FAtr)(FAtr)IN–policy)))((0-2)(1-1))
4	4	-10.810	-66.45	-10.810	(RSTR→(IA–vládní))(Adv→(IR–consistently))()
4	5	-9.430	-53.07	-9.430	(RSTR→(IA–vládní))(Atr→(ID–the))()
4	6	-9.406	-51.143	-9.406	(RSTR→(IA–vládní))(Atr→(IN–pro-consumption))()
4	7	-0.5395	-44.13	-0.5395	**(RSTR→(IA–vládní))(Atr→(IN–government))()**
4	8	-9.664	-51.83	-11.254	(RSTR→(IA–vládní))(Atr→((FAtr)(IN–government)IN–policy))((0-1))
4	9	-19.01	-44.20	-20.662	(RSTR→(FRSTR))(AuxP→(II–of ((FAtr)(FAtr)IN–policy)))((0-1)(1-2))
4	10	-28.284	-38.019	-31.480	(RSTR→(FRSTR))(Adv→((ID–the)IN–encouragement (FAuxP)))((1-1))
5	6	-8.028	-59.54	-10.913	(APP→((IA–prospotřebitelský)(FRSTR)IN–politika))(Atr→(IN–pro-consumption))((1-0))
5	7	-8.3	-57.66	-11.252	(APP→((FRSTR)(IA–vládní)IN–politika))(Atr→(IN–government))((1-0))
5	8	-0.6765	-50.70	-1.4428	(APP→((FRSTR)(IA–vládní)IN–politika))(Atr→((FAtr)(IN–government)IN–policy))((1-1))
5	9	-2.660	-42.636	-2.661	**(APP→((FRSTR)(FRSTR)IN–politika))(AuxP→(II–of ((FAtr)(FAtr)IN–policy)))((2-2)(1-1))**
5	10	-13.474	-34.65	-13.479	(APP→(FAPP))(Adv→((ID–the)IN–encouragement (FAuxP)))((1-1))
5	11	-21.248	-27.218	-22.90	(APP→(FAPP))(AuxP→(II–under ((FAtr)IN–encouragement (FAuxP))))((0-1)(1-2))
6	7	-17.5	-60.31	-20.66	(COND→(IN–podpora ((FRSTR)(FRSTR)IN–politika)))(Atr→(FAtr))((2-1)(1-0))
6	8	-2.656	-50.877	-2.661	(COND→(IN–podpora ((FRSTR)(FRSTR)IN–politika)))(Atr→((FAtr)(FAtr)IN–policy))((2-2)(1-1))
6	9	-2.661	-42.78	-2.6616	(COND→(IN–podpora ((FRSTR)(FRSTR)IN–politika)))(AuxP→(II–of ((FAtr)(FAtr)IN–policy)))((2-2)(1-1))
6	10	-12.781	-33.074	-13.47	**(COND→(IN–podpora (FAPP)))(Adv→((ID–the)IN–encouragement (FAuxP)))((1-1))**
6	11	-20.214	-25.248	-22.689	**(COND→(FCOND))(AuxP→(II–under (FAdv)))((1-1))**
6	12	-52.22	-19.28	-60.778	(COND→(IN–podpora (FAPP)))(Obj→((FSb)(IV–have)(FAtr)IV–grow (FAdv)(FAuxP)))((0-3)(0-2)(0-1)(1-4))
6	13	-23.30	-67.7	-29.421	(COND→(IN–podpora ((FRSTR)(FRSTR)IN–politika)))(AuxX→(I,–,))((2-0)(1-0))
6	14	-23.302	-82.80	-29.421	(COND→(IN–podpora ((FRSTR)(FRSTR)IN–politika)))(Atr→(ID–an))((2-0)(1-0))
7	8	-19.373	-63.821	-22.90	(PAT→((IN–poptávka)(FTHL)IV–stoupat (FCOND)))(Atr→(FAtr))((2-1)(1-0))
7	9	-18.68	-55.141	-22.90	(PAT→((IN–poptávka)(FTHL)IV–stoupat (FCOND)))(AuxP→(II–of (FAtr)))((2-1)(1-0))
7	10	-11.796	-46.003	-14.711	(PAT→((FACT)(ID–trvale)IV–stoupat (FCOND)))(Adv→((ID–the)IN–encouragement (FAuxP)))((2-1)(1-0))
7	11	-15.805	-37.404	-15.989	(PAT→((FACT)(ID–trvale)IV–stoupat (FCOND)))(AuxP→(II–under ((FAtr)IN–encouragement (FAuxP))))((2-2)(1-1))
7	12	-39.471	-6.256	-47.264	**(PAT→((FACT)(FTHL)IV–stoupat (FCOND)))(Obj→((FSb)(FAuxV)(IV–be)IV–grow (FAdv)(FAuxP)))((0-3)(3-4)(2-2)(1-1))**
7	13	-36.66	-75.02	-49.668	(PAT→((IN–poptávka)(FTHL)IV–stoupat (FCOND)))(AuxX→(I,–,))((2-0)(1-0))
7	14	-36.69	-80.21	-49.668	(PAT→((IN–poptávka)(FTHL)IV–stoupat (FCOND)))(Atr→(ID–an))((2-0)(1-0))
7	15	-36.69	-82.84	-49.668	(PAT→((IN–poptávka)(FTHL)IV–stoupat (FCOND)))(Atr→(IN–association))((2-0)(1-0))
8	12	-71.85	-39.84	-88.6	(PAT→(IN–asociace))(Obj→((IN–demand)(FAuxV)(FAtr)IV–grow (FAdv)(FAuxP)))((0-4)(0-3)(0-2)(0-1))
8	13	-10.813	-54.552	-10.814	(PAT→(IN–asociace))(AuxX→(I,–,))()
8	14	-10.815	-43.05	-10.816	(PAT→(IN–asociace))(Atr→(ID–an))()
8	15	-2.707	-43.053	-2.7073	**(PAT→(IN–asociace))(Atr→(IN–association))()**
8	16	-10.025	-41.130	-12.135	(PAT→(IN–asociace))(Sb→((FAtr)(IN–association)IN–spokesman))((0-1))
8	17	-92.80	-44.7	-111.5	(PAT→(FPAT))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-1)(1-2))
9	0	-7.983	-56.43	-10.817	(ACT→(IN–mluvčí (IN–asociace)))(NullSyntFunc→(FNullSyntFunc))()
9	13	-10.638	-55.02	-10.639	(ACT→(IN–mluvčí (IN–asociace)))(AuxX→(I,–,))()
9	14	-10.511	-51.903	-10.512	(ACT→(IN–mluvčí (IN–asociace)))(Atr→(ID–an))()
9	15	-2.4056	-51.90	-2.405	(ACT→(IN–mluvčí (IN–asociace)))(Atr→(IN–association))()
9	16	-4.602	-41.13	-4.631	**(ACT→(IN–mluvčí (FPAT)))(Sb→((ID–an)(FAtr)IN–spokesman))((1-1))**
9	17	-86.39	-43.981	-104.08	(ACT→(FACT))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-1)(1-2))
10	0	-48.714	-95.70	-69.92	(PRED→(((FACT)(FTHL)IV–stoupat (FCOND))IV–říci (FACT)))(NullSyntFunc→(FNullSyntFunc))((4-0)(3-0)(2-0)(1-0))
10	1	-46.516	-89.47	-62.07	(PRED→((FPAT)IV–říci (IN–mluvčí (FPAT))))(Sb→(FSb))((2-0)(1-1))
10	13	-51.38	-94.39	-69.74	(PRED→(((FACT)(FTHL)IV–stoupat (FCOND))IV–říci (FACT)))(AuxX→(FAuxX))((4-0)(3-0)(2-0)(1-1))
10	14	-51.39	-91.74	-69.61	(PRED→(((FACT)(FTHL)IV–stoupat (FCOND))IV–říci (FACT)))(Atr→(FAtr))((4-0)(3-0)(2-0)(1-1))
10	15	-46.097	-91.74	-61.5	(PRED→(((FACT)(FTHL)IV–stoupat (FCOND))IV–říci (FACT)))(Atr→(FAtr))((4-0)(3-0)(2-0)(1-1))
10	16	-38.53	-80.96	-54.29	(PRED→((FPAT)IV–říci (IN–mluvčí (FPAT))))(Sb→((FAtr)(FAtr)IN–spokesman))((2-2)(1-1))
10	17	-45.728	0.0	-53.55	**(PRED→((FPAT)IV–říci (FACT)))(Pred→((FObj)(I,–,)(FSb)IV–say))((2-2)(1-1))**

Table A.3: Computational chart with Viterbi probabilities for sentence pair "*Poptávka trvale stoupá za podpory prospotřebitelské vládní politiky, řekl mluvčí asociace.*" and "*Demand has been growing consistently under the encouragement of pro-consumption government policies, an association spokesman said.*"

A.3 Tree-to-Tree Alignment between Analytical Representations of Czech and English

Results of the automatic alignment between Czech and English analytical trees are presented here. The tree structures are results of an automatic annotation procedures, thus may contain errors. For each sentence pair the presentation contains the original tree pair, the Viterbi alignment and the computational chart.

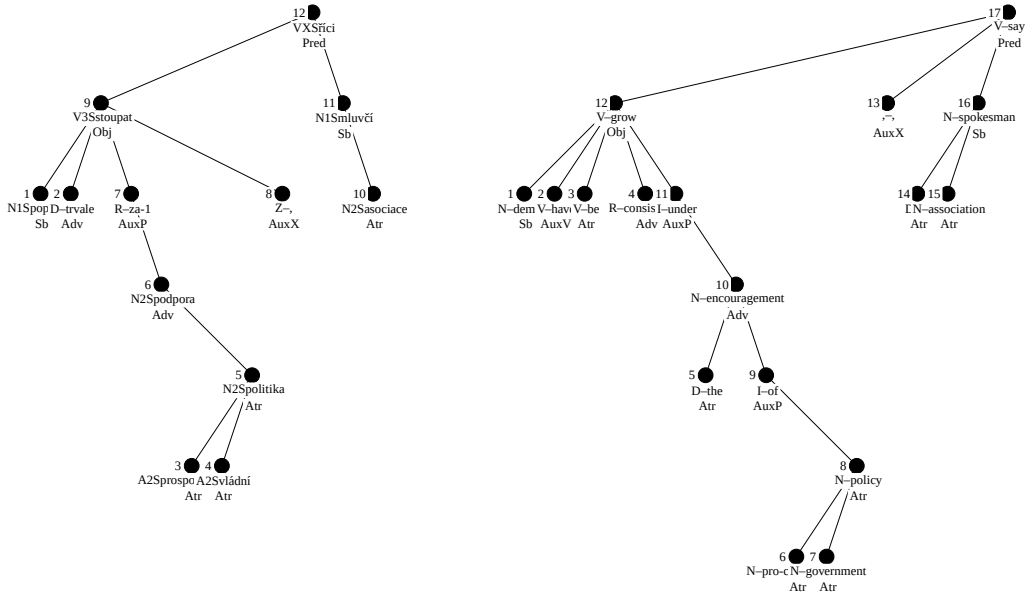


Figure A.7: A tree pair for Czech sentence *"Asociace uvedla, že domácí poptávka v září stoupla o 8,8 %."* and English sentence *"The association said domestic demand grew 8.8% in September."*

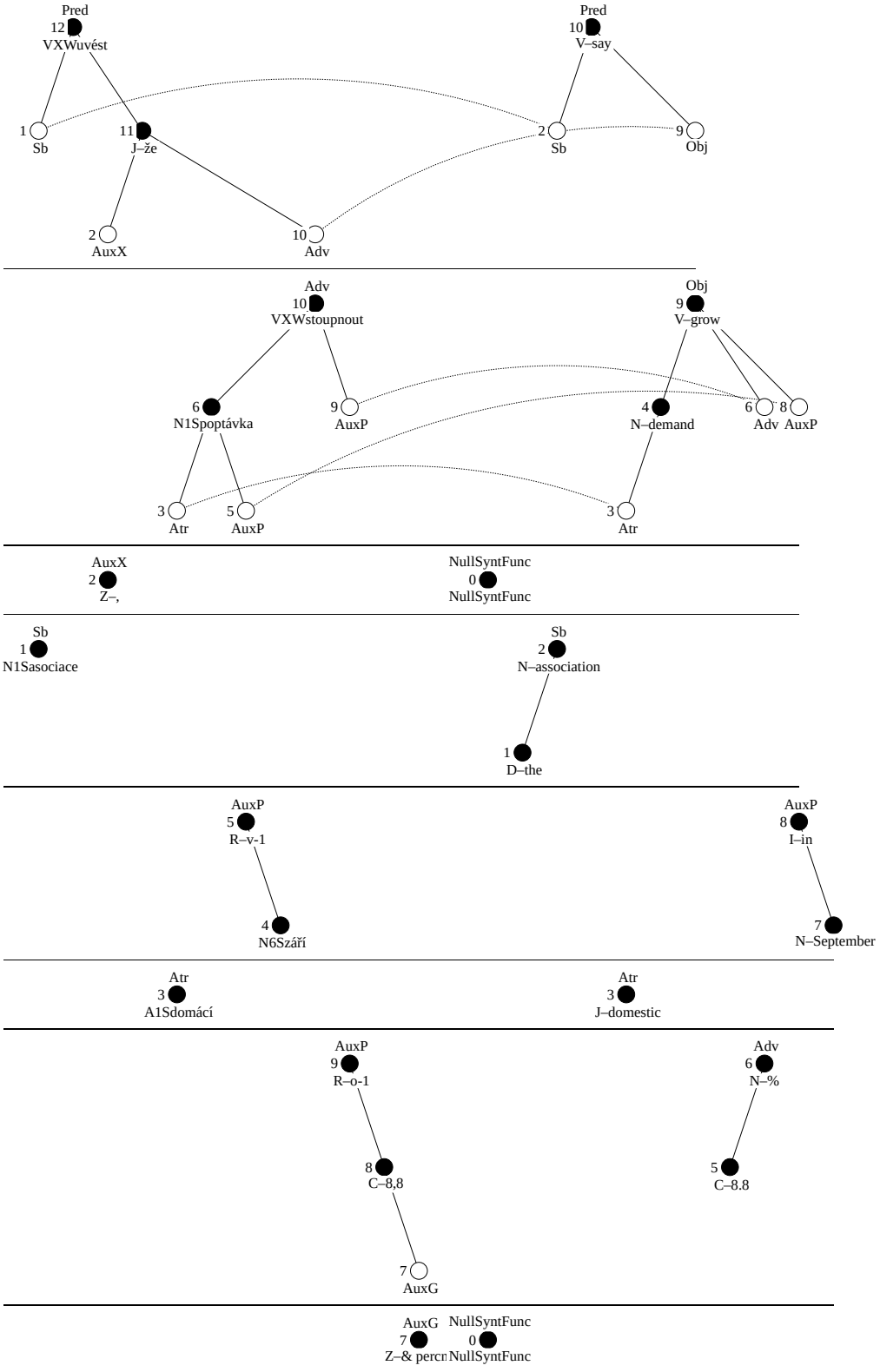


Figure A.8: Viterbi alignment of little trees for sentence pair *"Asociace uvedla, že domácí poptávka v září stoupla o 8,8 %."* and *"The association said domestic demand grew 8.8% in September."*

0	1	-6.947	-21.671	-9.43	(NullSyntFunc→(FNullSyntFunc))(Atr→(ID–the))()
0	2	-8.301	-23.347	-10.787	(NullSyntFunc→(FNullSyntFunc))(Sb→((ID–the)IN–association))()
0	8	-7.627	-21.597	-10.112	(NullSyntFunc→(FNullSyntFunc))(AuxP→(II–in (IN–September)))()
0	9	-31.390	-37.38	-41.02	(NullSyntFunc→(FNullSyntFunc))(Obj→((FSb)IV–grow ((FAtr)IN–%)(FAuxP)))((0-3)(0-2)(0-1))
0	10	-42.3	-51.668	-53.329	(NullSyntFunc→(FNullSyntFunc))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP)))((0-4)(0-3)(0-2)(0-1))
1	0	-7.137	-21.850	-9.440	(Sb→(IN1Sasociace))(NullSyntFunc→(FNullSyntFunc))()
1	1	-9.793	-19.725	-9.793	(Sb→(IN1Sasociace))(Atr→(ID–the))()
1	2	-2.266	-19.743	-2.2663	**(Sb→(IN1Sasociace))(Sb→((ID–the)IN–association))()**
1	3	-10.815	-55.79	-10.815	(Sb→(IN1Sasociace))(Atr→(IJ–domestic))()
1	4	-10.814	-44.97	-10.814	(Sb→(IN1Sasociace))(Sb→((IJ–domestic)IN–demand))()
1	5	-10.815	-53.73	-10.815	(Sb→(IN1Sasociace))(Atr→(IC–8.8))()
1	6	-10.81	-42.921	-10.814	(Sb→(IN1Sasociace))(Adv→((IC–8.8)IN–%))()
1	7	-10.816	-54.8	-10.816	(Sb→(IN1Sasociace))(Adv→(IN–September))()
1	8	-10.814	-44.04	-10.814	(Sb→(IN1Sasociace))(AuxP→(II–in (IN–September)))()
1	9	-39.04	-37.644	-41.02	(Sb→(IN1Sasociace))(Obj→((FSb)IV–grow ((FAtr)IN–%)(FAuxP)))((0-3)(0-2)(0-1))
1	10	-44.806	-49.99	-44.80	(Sb→(FSb))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP)))((0-4)(0-3)(0-2)(1-1))
2	0	-6.906	-15.12	-9.209	**(AuxX→(IZ–,))(NullSyntFunc→(FNullSyntFunc))()**
2	1	-9.207	-19.974	-9.207	(AuxX→(IZ–,))(Atr→(ID–the))()
2	2	-10.76	-19.99	-10.768	(AuxX→(IZ–,))(Sb→((ID–the)IN–association))()
3	0	-7.382	-25.28	-9.685	(Atr→(IA1Sdomácí))(NullSyntFunc→(FNullSyntFunc))()
3	1	-9.513	-53.59	-9.513	(Atr→(IA1Sdomácí))(Atr→(ID–the))()
3	2	-10.805	-42.86	-10.805	(Atr→(IA1Sdomácí))(Sb→((ID–the)IN–association))()
3	3	-0.3364	-21.768	-0.3364	**(Atr→(IA1Sdomácí))(Atr→(IJ–domestic))()**
3	4	-1.6303	-22.764	-1.630	(Atr→(IA1Sdomácí))(Sb→((IJ–domestic)IN–demand))()
3	5	-9.513	-32.75	-9.513	(Atr→(IA1Sdomácí))(Atr→(IC–8.8))()
3	6	-10.489	-30.60	-10.490	(Atr→(IA1Sdomácí))(Adv→((IC–8.8)IN–%))()
3	7	-10.815	-30.802	-10.815	(Atr→(IA1Sdomácí))(Adv→(IN–September))()
5	3	-9.438	-23.028	-9.439	(AuxP→(IR–v-1 (IN6Září)))(Atr→(IJ–domestic))()
5	4	-9.51	-31.475	-9.514	(AuxP→(IR–v-1 (IN6Září)))(Sb→((IJ–domestic)IN–demand))()
5	5	-9.448	-33.374	-9.448	(AuxP→(IR–v-1 (IN6Září)))(Atr→(IC–8.8))()
5	6	-10.80	-30.085	-10.810	(AuxP→(IR–v-1 (IN6Září)))(Adv→((IC–8.8)IN–%))()
5	7	-1.615	-24.2	-1.6155	(AuxP→(IR–v-1 (IN6Září)))(Adv→(IN–September))()
5	8	-1.071	-20.959	-1.1625	**(AuxP→(IR–v-1 (IN6Září)))(AuxP→(II–in (IN–September)))()**
5	9	-22.171	-40.82	-25.398	(AuxP→(IR–v-1 (FAtr)))(Obj→((FSb)IV–grow (FAdv)(II–in (FAdv))))((0-2)(0-1)(1-3))
5	10	-38.81	-51.644	-44.37	(AuxP→(FAuxP))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-3)(0-2)(0-1)(1-4))
6	8	-10.595	-21.557	-11.98	(Sb→((IA1Sdomácí)IN1Spoptávka (FAuxP)))(AuxP→(FAuxP))((1-1))
6	9	-11.71	-30.044	-14.13	(Sb→((FAtr)IN1Spoptávka (FAuxP)))(Obj→(((FAtr)IN–demand)IV–grow (FAdv)(FAuxP)))((0-1)(2-2)(1-3))
6	10	-29.66	-40.86	-34.38	(Sb→(FSb))(Pred→(((FAtr)IN–association)IV–say (FObj)))((0-2)(1-1))
7	0	-6.906	-15.275	-9.209	**(AuxG→(IZ–& percnt;))(NullSyntFunc→(FNullSyntFunc))()**
7	1	-9.430	-42.781	-9.431	(AuxG→(IZ–& percnt;))(Atr→(ID–the))()
7	2	-10.813	-43.45	-10.814	(AuxG→(IZ–& percnt;))(Sb→((ID–the)IN–association))()
7	3	-9.207	-34.24	-9.207	(AuxG→(IZ–& percnt;))(Atr→(IJ–domestic))()
7	4	-10.813	-35.17	-10.814	(AuxG→(IZ–& percnt;))(Sb→((IJ–domestic)IN–demand))()
7	5	-9.430	-24.50	-9.431	(AuxG→(IZ–& percnt;))(Atr→(IC–8.8))()
7	6	-0.01035	-24.54	-0.010485	(AuxG→(IZ–& percnt;))(Adv→((IC–8.8)IN–%))()
9	5	-8.373	-17.002	-10.730	(AuxP→(IR–o-1 (IC–8,8 (FAuxG))))(Atr→(IC–8.8))((1-0))
9	6	-8.316	-13.727	-10.797	**(AuxP→(IR–o-1 (IC–8,8 (FAuxG))))(Adv→((IC–8.8)IN–%))((1-0))**
9	7	-17.50	-25.54	-20.02	(AuxP→(IR–o-1 (IC–8,8 (FAuxG))))(Adv→(IN–September))((1-0))
9	8	-17.447	-22.99	-20.019	(AuxP→(IR–o-1 (IC–8,8 (FAuxG))))(AuxP→(II–in (IN–September)))((1-0))
9	9	-20.27	-35.26	-25.632	(AuxP→(IR–o-1 (IC–8,8 (FAuxG))))(Obj→(((FAtr)IN–demand)IV–grow (FAdv)(FAuxP)))((0-3)(0-2)(1-1))
9	10	-30.854	-43.897	-37.78	(AuxP→(IR–o-1 (IC–8,8 (FAuxG))))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP)))))((0-4)(0-2)(0-1)(1-3))
10	5	-26.765	-29.817	-31.374	(Adv→((FSb)IVXWstoupnout (IR–o-1 (FAdv))))(Atr→(FAtr))((2-1)(1-0))
10	6	-26.253	-21.008	-31.371	(Adv→((FSb)IVXWstoupnout (IR–o-1 (FAdv))))(Adv→((FAtr)IN–%))((2-1)(1-0))
10	7	-31.66	-30.116	-34.060	(Adv→((FSb)IVXWstoupnout (IR–o-1 (FAdv))))(Adv→(FAdv))((2-0)(1-1))
10	8	-29.746	-22.2	-33.607	(Adv→((FSb)IVXWstoupnout (IR–o-1 (FAdv))))(AuxP→(FAuxP))((2-0)(1-1))
10	9	-11.283	-10.72	-13.954	**(Adv→(((FAtr)IN1Spoptávka (FAuxP))IVXWstoupnout (FAuxP)))(Obj→(((FAtr)IN–demand)IV–grow (FAdv)(FAuxP)))((3-2)(2-3)(1-1))**
10	10	-23.070	-21.59	-27.956	(Adv→(((FAtr)IN1Spoptávka (FAuxP))IVXWstoupnout (FAuxP)))(Pred→((FSb)IV–say ((FSb)IV–grow (FAdv)(FAuxP))))((0-1)(3-4)(2-2)(1-3))
11	0	-40.962	-28.130	-52.48	(AuxC→((IZ–,)IJ–že (FAdv)))(NullSyntFunc→(FNullSyntFunc))((1-0))
11	1	-36.577	-32.65	-45.80	(AuxC→((IZ–,)IJ–že (FAdv)))(Atr→(ID–the))((1-0))
11	2	-36.577	-25.866	-45.80	(AuxC→((IZ–,)IJ–že (FAdv)))(Sb→((ID–the)IN–association))((1-0))
11	3	-38.62	-30.12	-43.311	(AuxC→((IZ–,)IJ–že (FAdv)))(Atr→(FAtr))((1-1))
12	6	-46.14	-35.05	-51.53	(Pred→((IN1Sasociace)IVXWuvést (FAuxC)))(Adv→(FAdv))((1-1))
12	7	-51.013	-44.80	-54.222	(Pred→((IN1Sasociace)IVXWuvést (FAuxC)))(Adv→(FAdv))((1-1))
12	8	-49.851	-35.75	-54.21	(Pred→((IN1Sasociace)IVXWuvést (FAuxC)))(AuxP→(II–in (FAdv)))((1-1))
12	9	-32.53	-17.535	-35.23	(Pred→((IN1Sasociace)IVXWuvést (FAuxC)))(Obj→(FObj))((1-1))
12	10	-22.007	0.0	-27.005	**(Pred→((FSb)IVXWuvést ((FAuxX)IJ–že (FAdv))))(Pred→((FSb)IV–say (FObj)))((3-2)(2-0)(1-1))**

Table A.4: Computational chart with Viterbi probabilities for sentence pair *"Asociace uvedla, že domácí poptávka v září stoupla o 8,8 %."* and *"The association said domestic demand grew 8.8% in September."*

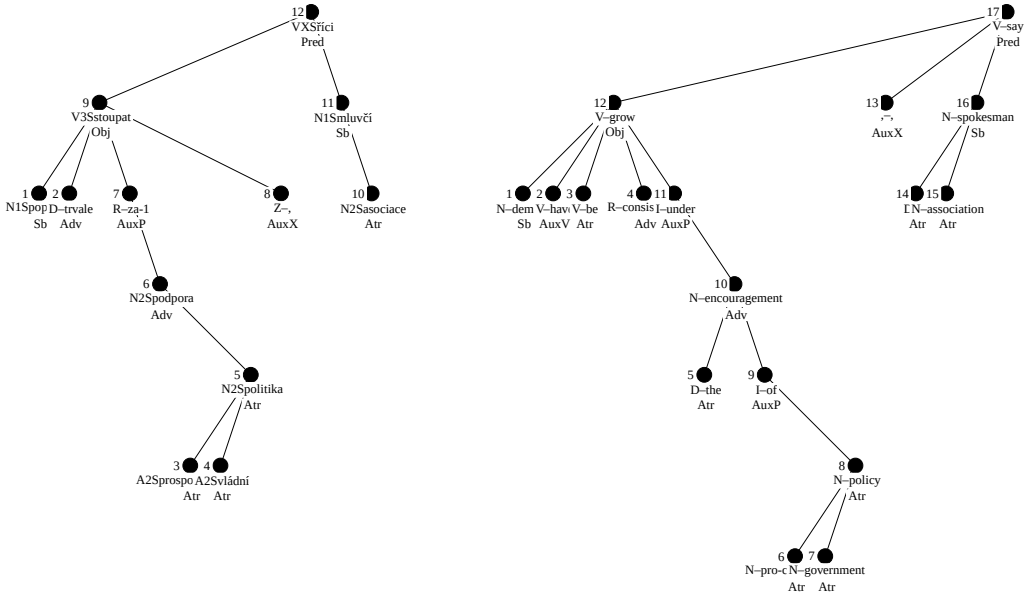


Figure A.9: A tree pair for Czech sentence "*Poptávka trvale stoupá za podpory prospotřebitelské vládní politiky, řekl mluvčí asociace.*" and English sentence "*Demand has been growing consistently under the encouragement of pro-consumption government policies, an association spokesman said.*"

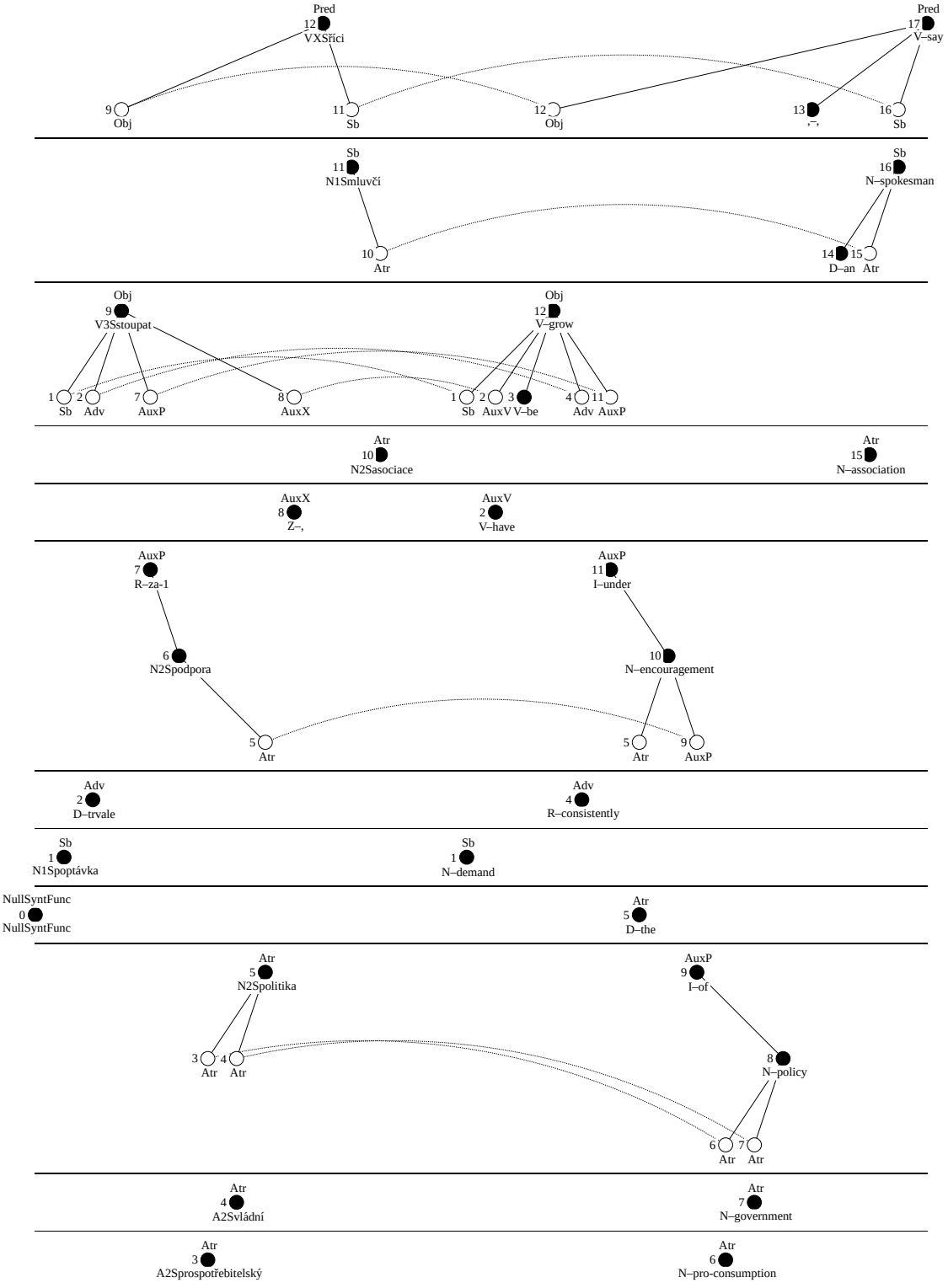


Figure A.10: Viterbi alignment of little trees for sentence pair *"Poptávka trvale stoupá za podpory prospotřebitelské vládní politiky, řekl mluvčí asociace."* and English sentence *"Demand has been growing consistently under the encouragement of pro-consumption government policies, an association spokesman said."*

0	1	-6.732	-43.78	-9.217	(NullSyntFunc→(FNullSyntFunc))(Sb→(IN–demand))()
0	2	-6.947	-37.10	-9.432	(NullSyntFunc→(FNullSyntFunc))(AuxV→(IV–have))()
0	3	-6.947	-36.863	-9.4	(NullSyntFunc→(FNullSyntFunc))(Atr→(IV–be))()
0	4	-6.724	-37.200	-9.209	(NullSyntFunc→(FNullSyntFunc))(Adv→(IR–consistently))()
0	5	-6.947	-35.78	-9.43	**(NullSyntFunc→(FNullSyntFunc))(Atr→(ID–the))()**
0	6	-6.848	-42.92	-9.333	(NullSyntFunc→(FNullSyntFunc))(Atr→(IN–pro-consumption))()
0	7	-6.848	-42.434	-9.333	(NullSyntFunc→(FNullSyntFunc))(Atr→(IN–government))()
0	17	-93.3	-58.88	-119.38	(NullSyntFunc→(FNullSyntFunc))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-2)(0-1))
1	0	-6.607	-47.39	-9.440	(Sb→(IN1Spoptávka))(NullSyntFunc→(FNullSyntFunc))()
1	1	-1.7701	-41.20	-1.7701	**(Sb→(IN1Spoptávka))(Sb→(IN–demand))()**
1	2	-9.916	-41.30	-9.916	(Sb→(IN1Spoptávka))(AuxV→(IV–have))()
1	3	-10.815	-41.09	-10.815	(Sb→(IN1Spoptávka))(Atr→(IV–be))()
2	2	-9.45	-35.19	-9.45	(Adv→(ID–trvale))(AuxV→(IV–have))()
2	3	-10.816	-35.17	-10.816	(Adv→(ID–trvale))(Atr→(IV–be))()
2	4	-9.207	-35.31	-9.207	**(Adv→(ID–trvale))(Adv→(IR–consistently))()**
2	5	-10.816	-60.683	-10.816	(Adv→(ID–trvale))(Atr→(ID–the))()
2	6	-3.2746	-74.58	-3.2746	(Adv→(ID–trvale))(Atr→(IN–pro-consumption))()
2	7	-10.81	-74.58	-10.816	(Adv→(ID–trvale))(Atr→(IN–government))()
3	3	-9.51	-74.11	-9.513	(Atr→(IA2Sprospotřebitelský))(Atr→(IV–be))()
3	4	-10.815	-74.37	-10.815	(Atr→(IA2Sprospotřebitelský))(Adv→(IR–consistently))()
3	5	-9.51	-53.20	-9.513	(Atr→(IA2Sprospotřebitelský))(Atr→(ID–the))()
3	6	-0.3073	-42.239	-0.3073	**(Atr→(IA2Sprospotřebitelský))(Atr→(IN–pro-consumption))()**
3	7	-9.513	-46.61	-9.513	(Atr→(IA2Sprospotřebitelský))(Atr→(IN–government))()
3	8	-9.599	-50.45	-10.943	(Atr→(IA2Sprospotřebitelský))(Atr→((IN–pro-consumption)(FAtr)IN–policy))((0-1))
3	9	-19.06	-44.325	-20.458	(Atr→(FAtr))(AuxP→(II–of ((FAtr)(FAtr)IN–policy)))((0-2)(1-1))
4	4	-10.815	-74.37	-10.815	(Atr→(IA2Svládní))(Adv→(IR–consistently))()
4	5	-9.513	-52.96	-9.513	(Atr→(IA2Svládní))(Atr→(ID–the))()
4	6	-5.30	-48.53	-5.304	(Atr→(IA2Svládní))(Atr→(IN–pro-consumption))()
4	7	-0.7889	-41.758	-0.788	**(Atr→(IA2Svládní))(Atr→(IN–government))()**
4	8	-9.838	-50.42	-11.42	(Atr→(IA2Svládní))(Atr→((FAtr)(IN–government)IN–policy))((0-1))
4	9	-19.258	-44.32	-20.940	(Atr→(FAtr))(AuxP→(II–of ((FAtr)(FAtr)IN–policy)))((0-1)(1-2))
4	10	-28.47	-43.814	-31.67	(Atr→(FAtr))(Adv→((FAtr)IN–encouragement (II–of (FAtr))))((0-1)(1-2))
5	6	-8.388	-56.6	-11.125	(Atr→((FAtr)(IA2Svládní)IN2Spolitika))(Atr→(FAtr))((1-1))
5	7	-8.876	-53.94	-11.606	(Atr→((IA2Sprospotřebitelský)(FAtr)IN2Spolitika))(Atr→(FAtr))((1-1))
5	8	-0.8113	-48.05	-1.5279	(Atr→((FAtr)(IA2Svládní)IN2Spolitika))(Atr→((FAtr)(IN–government)IN–policy))((1-1))
5	9	-2.856	-40.37	-2.8586	**(Atr→((FAtr)(FAtr)IN2Spolitika))(AuxP→(II–of ((FAtr)(FAtr)IN–policy)))((2-2)(1-1))**
5	10	-13.671	-33.55	-13.67	(Atr→(FAtr))(Adv→((ID–the)IN–encouragement (FAuxP)))((1-1))
5	11	-21.448	-34.901	-23.10	(Atr→(FAtr))(AuxP→(II–under ((FAtr)IN–encouragement (FAuxP))))((0-1)(1-2))
5	12	-53.78	-29.839	-61.78	(Atr→(FAtr))(Obj→((FSb)(FAuxV)(IV–be)IV–grow (FAdv)(FAuxP)))((0-3)(0-2)(0-1)(1-4))
7	8	-11.476	-49.34	-12.345	(AuxP→(IR–za-1 (IN2Spodpora (FAtr))))(Atr→(FAtr))((1-1))
7	9	-10.324	-41.43	-11.269	(AuxP→(IR–za-1 (IN2Spodpora (FAtr))))(AuxP→(II–of (FAtr)))((1-1))
7	10	-12.96	-31.631	-13.66	(AuxP→(IR–za-1 (IN2Spodpora (FAtr))))(Adv→((ID–the)IN–encouragement (FAuxP)))((1-1))
7	11	-19.089	-24.119	-21.803	**(AuxP→(IR–za-1 (IN2Spodpora (FAtr))))(AuxP→(II–under ((FAtr)IN–encouragement (FAuxP))))((0-1)(1-2))**
7	12	-50.06	-19.02	-60.478	(AuxP→(IR–za-1 (FAdv)))(Obj→((FSb)(FAuxV)(FAtr)IV–grow (FAdv)(II–under (FAdv))))((0-4)(0-3)(0-2)(0-1)(1-5))
7	13	-24.891	-69.21	-31.069	(AuxP→(IR–za-1 (IN2Spodpora (FAtr))))(AuxX→(FAuxX))((1-1))
7	17	-76.89	-38.01	-91.44	(AuxP→(FAuxP))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-2)(1-1))
8	0	-6.376	-41.70	-9.209	(AuxX→(IZ–,))(NullSyntFunc→(FNullSyntFunc))()
8	1	-9.439	-41.725	-9.43	(AuxX→(IZ–,))(Sb→(IN–demand))()
8	2	-9.225	-35.24	-9.225	**(AuxX→(IZ–,))(AuxV→(IV–have))()**
8	3	-9.430	-35.19	-9.430	(AuxX→(IZ–,))(Atr→(IV–be))()
8	4	-9.265	-35.38	-9.266	(AuxX→(IZ–,))(Adv→(IR–consistently))()
8	5	-9.20	-60.673	-9.207	(AuxX→(IZ–,))(Atr→(ID–the))()
8	6	-9.43	-74.68	-9.430	(AuxX→(IZ–,))(Atr→(IN–pro-consumption))()
9	10	-32.04	-43.691	-40.466	(Obj→((FSb)(FAdv)IV3Sstoupat (IR–za-1 (FAdv))(FAuxX)))(Adv→((FAtr)IN–encouragement (II–of (FAtr))))((4-2)(3-1)(2-0)(1-0))
9	11	-33.79	-35.51	-40.23	(Obj→((FSb)(FAdv)IV3Sstoupat (IR–za-1 (FAdv))(FAuxX)))(AuxP→(II–under ((FAtr)IN–encouragement (FAuxP))))((4-2)(3-1)(2-0)(1-0))
9	12	-37.704	-5.595	-44.97	**(Obj→((FSb)(FAdv)IV3Sstoupat (FAuxP)(FAuxX)))(Obj→((FSb)(FAuxV)(IV–be)IV–grow (FAdv)(FAuxP)))((4-2)(3-4)(2-3)(1-1))**
9	13	-37.08	-71.93	-51.839	(Obj→((IN1Spoptávka)(FAdv)IV3Sstoupat (FAuxP)(FAuxX)))(AuxX→(FAuxX))((3-1)(2-0)(1-0))
9	14	-42.62	-77.43	-55.77	(Obj→((IN1Spoptávka)(FAdv)IV3Sstoupat (FAuxP)(FAuxX)))(Atr→(FAtr))((3-0)(2-1)(1-0))
9	15	-44.636	-80.93	-60.39	(Obj→((IN1Spoptávka)(FAdv)IV3Sstoupat (FAuxP)(FAuxX)))(Atr→(FAtr))((3-0)(2-1)(1-0))
10	12	-69.47	-44.6	-88.40	(Atr→(IN2Sasociace))(Obj→((FSb)(FAuxV)(IV–be)IV–grow (FAdv)(FAuxP)))((0-4)(0-3)(0-2)(0-1))
10	13	-9.433	-52.62	-9.433	(Atr→(IN2Sasociace))(AuxX→(I,–,))()
10	14	-9.513	-41.541	-9.513	(Atr→(IN2Sasociace))(Atr→(ID–an))()
10	15	-1.727	-41.583	-1.7273	**(Atr→(IN2Sasociace))(Atr→(IN–association))()**
10	16	-10.071	-39.7	-12.462	(Atr→(IN2Sasociace))(Sb→((FAtr)(IN–association)IN–spokesman))((0-1))
10	17	-90.28	-52.73	-111.69	(Atr→(FAtr))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-1)(1-2))
11	0	-7.978	-54.73	-10.812	(Sb→(IN1Smluvčí (IN2Sasociace)))(NullSyntFunc→(FNullSyntFunc))()
11	13	-10.119	-53.28	-10.1	(Sb→(IN1Smluvčí (IN2Sasociace)))(AuxX→(I,–,))()
11	14	-10.790	-50.48	-10.793	(Sb→(IN1Smluvčí (IN2Sasociace)))(Atr→(ID–an))()
11	15	-3.0090	-50.49	-3.0091	(Sb→(IN1Smluvčí (IN2Sasociace)))(Atr→(IN–association))()
11	16	-3.5656	-39.73	-3.578	**(Sb→(IN1Smluvčí (FAtr)))(Sb→((ID–an)(FAtr)IN–spokesman))((1-1))**
11	17	-82.	-51.80	-102.80	(Sb→(FSb))(Pred→((FObj)(I,–,)(FSb)IV–say))((0-1)(1-2))
12	0	-56.15	-93.21	-80.81	(Pred→(((FSb)(FAdv)IV3Sstoupat (FAuxP)(FAuxX))IVXSříci (FSb)))(NullSyntFunc→(FNullSyntFunc))((5-0)(4-0)(3-0)(2-0)(1-0))
12	1	-53.83	-87.91	-72.06	(Pred→((FObj)IVXSříci (IN1Smluvčí (FAtr))))(Sb→(FSb))((2-0)(1-1))
12	14	-57.76	-89.46	-76.03	(Pred→(((FSb)(FAdv)IV3Sstoupat (FAuxP)(FAuxX))IVXSříci (FSb)))(Atr→(FAtr))((5-0)(4-1)(3-0)(2-0)(1-0))
12	15	-53.92	-89.47	-73.01	(Pred→(((FSb)(FAdv)IV3Sstoupat (FAuxP)(FAuxX))IVXSříci (FSb)))(Atr→(FAtr))((5-0)(4-0)(3-0)(2-0)(1-1))
12	16	-45.031	-78.71	-59.340	(Pred→((FObj)IVXSříci (IN1Smluvčí (FAtr))))(Sb→((FAtr)(FAtr)IN–spokesman))((2-2)(1-1))
12	17	-43.29	0.0	-50.58	**(Pred→((FObj)IVXSříci (FSb)))(Pred→((FObj)(I,–,)(FSb)IV–say))((2-2)(1-1))**

Table A.5: Computational chart with Viterbi probabilities for sentence pair *"Poptávka trvale stoupá za podpory prospotřebitelské vládní politiky, řekl mluvčí asociace."* and English sentence *"Demand has been growing consistently under the encouragement of pro-consumption government policies, an association spokesman said."*

Appendix B

Implementation details

The framework for experiments with dependency tree structure has been implemented in Java and is available as a library. This appendix briefly summarizes the most important features of the framework.

B.1 A Java Framework for Tree Transformations

The Java framework supports operations with many types of trees: *analytical*, *tectogrammatical*, *phrase structures of Penn Treebank*, *packed-tree representaion of tectogrammatical trees*, *Collins' trees*[1], and *Charniak's trees*.[2]

B.1.1 Installation

The framework can be installed from the CVS located at UFAL to the current directory `...myDir` using:

```
...myDir>cvs -d/home/CVSROOT/cmejrek checkout CETGTranslation
```

And built using:

```
...myDir>cd CETGTranslation
...myDir/CETGTranslation>ant jar
```

The resulting jar is then created as:

```
...myDir/CETGTranslation/dist/CETGTranslation.jar
```

B.1.2 Basic Tree Operations

Basic tree operations are: loading and serialization of trees from and to the `csts` format, accessing child nodes, as well as iterations in various orderings, such as *prefix*, *postfix*, *breadth-first*, or *depth-first*. This functionality is accessible through the *NodeInterface.java*.

[1] Resulting from the Collins' parser

[2] Resulting from the Charniak's parser.

B.1.3 Implementation of PDT-Specific Trees

PDT-specific features are accessible through the following interfaces.

Analytical Trees

The functionality specific to the analytical trees is accessible through the *ARNodeInterface.java*. The interface enables manipulation with values specific to the analytical representation, such as *form*, *lemma*, *morphological tag*, *afun*, *word order* (r), *word order of the parent node*(g), etc.

Tectogrammatical Trees

The functionality specific to the tectogrammatical trees is accessible through the *TRNodeInterface.java*. The interface enables manipulation with tectogrammatical attributes, such as *trlemma*, *functor*, *tectogrammatical morphoogical tag*, *topic-focus articulation*, *deep word order*, etc.

Packed Tectogrammatical Trees

The functionality specific to the packed-tree representation used for transferring tectogrammatical trees is accessible through the *TransferNodeInterface.java*. The packed-tree representation anables to store variants of the treestructures.

B.2 Penn Treebank Trees

The functionality specific to Penn Treebank trees is accessible through *WSJNodeInterface.java*. The inteface enables manipulation with specific features of the phrase-structure used in Penn Treebank, such as *form*, *lemma*, *WSJ POS tag*, or *nonterminal*.

B.2.1 Custom Tree-Convertors

Many convertors from various formats into PDT style of annotation have been implemented.

Penn Treebank to Analytical Trees

Class *WSJToATSTreeConverter.java* implements a convertor from the Penn Treebank style of annotation into analytical trees.

Integration of Charniak's Parser

Class *ConvertEugeneCharniakTreeToCSTS.java* converts the output format of the Charniak's parset o Czech into the analytical representation.

Integration of Collins' Parser

Class *CollinsTreeToATSTreeConverter.java* converts output format of the Collins' parser for Czech into analytical representation.

B.2.2 DBMT system: Rule-based MT

The rule-based MT system described in Chapter 3 is implemented in class *RANLP03GenerationFromTGTTransfer.java*.

B.2.3 Implementation of Tree-to-Tree Transducer

The framework for STSG modeling has been implemented as class *TTengine.java*. The engine can be run with the following parameters:

- `-I` *CETGTPathPrefix*
  Prefix to the working directory, where all model files and other working file will be stored.
- `-properties` *property_file*
- `-f` *list_of_files*
  List of pairs of files with parallel trees. The files should contain the output of the *nsgmls* parser run on `csts` trees.
- `-l` *log_file*
  The name of the main log file.
- `-d` *debugLevel*
  Debug level of the mai logger.
- `-L` *logger_name log_file log_level*
  Enables to define different files and levels of logging for different types of logging. Known loggers are: `latexLT` for LaTeXlogs of synchronous rules, and `modelSimple` for textual representation of the probabilistic model of the STSG.
- `-D` *dictionary*
  Probabilistic translation dictionary to be imported.

- `-modelFN` *modelPrefix*
  Prefix of the model file.
- `-iteration` *iteration*
  Number of the current iteration.
- `-startFromSentence` *first_sentence*
- `-sentences` *last_sentence*
- `-countNonSynchronous`
  Turns on counting the non-synchronous rules.
- `-observeSynchronous`
  Turns on observing synchronous rules. This is the training part.
- `-createTrDict`
  Creates translation dictionary from the PDT links between analytical and tectogrammatical nodes.
- `-bm` *backoffModel lambda*
  Sets λ for the selected backoff model.
- `-sumInsideProbs`
  Turns on computing the sum of inside probabilities (for evaluating different Λ).
- `-saveNewModel`
  The resulting trained model will be saved.
- `-prepareGIZATrainingData`
  Prepares the parallel corpus of plain text for GIZA++ training.
- `-generateLatex`
  Turns on additional logging of little trees into LaTeX.y
- `-traceViterbi`
  Turns on tracing Viterbi alignments during the computation of inside probabilities.

Bibliography

[Al-Onaizan et al., 1999] Al-Onaizan, Y., Cuřín, J., Jahr, M., Knight, K., Lafferty, J., Melamed, D., Och, F.-J., Purdy, D., Smith, N. A., and Yarowsky, D. (1999). The Statistical Machine Translation. Technical report. NLP WS'99 Final Report.

[Berger et al., 1994] Berger, A., Brown, P., Della-Pietra, S., Della-Pietra, V., Gillett, J., Lafferty, J., Mercer, R., Printz, H., and Ureš, L. (1994). The Candide System for Machine Translation. In *Proceedings of the ARPA Human Language Technology Workshop*.

[Böhmová, 2001] Böhmová, A. (2001). Automatic Procedures in Tectogrammatical Tagging. *The Prague Bulletin of Mathematical Linguistics*, 76.

[Brown et al., 1992] Brown, P. F., Pietra, V. J. D., deSouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

[Brown et al., 1993] Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2).

[Charniak, 1999] Charniak, E. (1999). A Maximum-Entropy-Inspired Parser. Technical Report CS-99-12.

[Cikhart et al., 2001] Cikhart, O., Čmejrek, M., and Cuřín, J. (2001). Povídejte si se svým počítačem. *Softwarové noviny*, XII(5):26–37.

[Čmejrek, 1998] Čmejrek, M. (1998). Automatická extrakce dvojjazyčného pravděpodobnostního slovníku z paralelních textů. Master's thesis, Charles University, Prague. In Czech.

[Čmejrek et al., 2005] Čmejrek, M., Cuřín, J., Hajič, J., and Havelka, J. (2005). Prague Czech-English Dependency Treebank: Resource for Structure-based MT. In Hutchins, J., Kis, B., and Prószéky, G., editors, *Proceedings of the 10th EAMT Conference*, pages 73–78, Budapest, Hungary. European Association for Machine Translation. ISBN 963 9206 04 0.

[Čmejrek et al., 2003a] Čmejrek, M., Cuřín, J., and Havelka, J. (2003a). Czech-English Dependency-based Machine Translation. In *Proceedings of the 10th Conference of The European Chapter of the Association for Computational Linguistics*, pages 83–90, Budapest, Hungary.

[Čmejrek et al., 2003b] Čmejrek, M., Cuřín, J., and Havelka, J. (2003b). Treebanks in Machine Translation. In *Proceedings of The Second Workshop on Treebanks and Linguistic Theories*, pages 209–212, Vaxjo, Sweden.

[Čmejrek et al., 2004a] Čmejrek, M., Cuřín, J., and Havelka, J. (2004a). Prague Czech-English Dependency Treebank: Any Hopes for a Common Annotation Scheme? In *Frontiers in Corpus Annotation. Proceedings of the Workshop of the HLT/NAACL Conference*, pages 47–54. MSM113200006, LN00A063.

[Čmejrek et al., 2004b] Čmejrek, M., Cuřín, J., Havelka, J., Hajič, J., and Kuboň, V. (2004b). Prague Czech-English Dependency Treebank. Syntactically Annotated Resources for Machine Translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, volume V, pages 1597–1600, Lisboa, Portugal. European Language Resources Association. ISBN 2-9517408-1-6.

[Collins et al., 1999] Collins, M., Hajič, J., Ramshaw, L., and Tillmann, C. (1999). A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland.

[Cuřín and Čmejrek, 1999a] Cuřín, J. and Čmejrek, M. (1999a). Automatic Extraction of Terminological Translation Lexicon from Czech-English Parallel Texts. In *Proceedings of 4th TELRI-II Seminar*, Bratislava, Slovakia.

[Cuřín and Čmejrek, 1999b] Cuřín, J. and Čmejrek, M. (1999b). Automatic Translation Lexicon Extraction from Czech-English Parallel Texts. *The Prague Bulletin of Mathematical Linguistics*, 71:47–57.

[Cuřín and Čmejrek, 2001] Cuřín, J. and Čmejrek, M. (2001). Automatic Extraction of Terminological Translation Lexicon from Czech-English Parallel Texts. *International Journal of Corpus Linguistics*, 6(Special Issue):1–12.

[Cuřín et al., 2002] Cuřín, J., Čmejrek, M., and Havelka, J. (2002). Czech-English Dependency-based Machine Translation: Data Preparation for the Starting up Experiments. *The Prague Bulletin of Mathematical Linguistics*, 78:103–116.

[Cuřín et al., 2004a] Cuřín, J., Čmejrek, M., Havelka, J., Hajič, J., Kuboň, V., and Žabokrtský, Z. (2004a). Prague Czech-English Dependency Treebank, version 1.0. Linguistic Data Consortium (LDC).

[Cuřín et al., 2004b] Cuřín, J., Čmejrek, M., Havelka, J., and Kuboň, V. (2004b). Building parallel bilingual syntactically annontated corpus. In *Proceedings of The First International Joint Conference on Natural Language Processing*, pages 141–146, Hainan Island, China.

[Cuřín et al., 2005] Cuřín, J., Čmejrek, M., Havelka, J., and Kuboň, V. (2005). Building a parallel bilingual syntactically annontated corpus. In Keh-Yih Su, Jun'ichi Tsujii, J.-H. L. e. a., editor, *Natural Language Processing - IJCNLP 2004: First International Joint Conference, Hainan Island, China, March 22-24, 2004, Revised Selected Papers*, volume 3248 of *LNAI*, pages 168–176. ISBN: 3-540-24475-1.

[Eisner, 2001] Eisner, J. (2001). *Smoothing a Probabilistic Lexicon via Syntactic Transformations*. PhD thesis, University of Pennsylvania. 318 pages.

[Eisner, 2003] Eisner, J. (2003). Learning Non-Isomorphic Tree Mappings for Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (Companion Volume)*, Sapporo.

[Germann et al., 2001] Germann, U., Jahr, M., Knight, K., Marcu, D., and Yamada, K. (2001). Fast Decoding and Optimal Decoding for Machine Translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 228–235.

[Hajič et al., 2003] Hajič, J., Homola, P., and Kuboň, V. (2003). A Simple Multilingual Machine Translation System. In Hovy, Eduard//Macklovitch, E., editor, *Proceedings of Machine Translation Summit IX*, pages 157—164, New Orleans, USA.

[Hajič, 1987] Hajič, J. (1987). Ruslan: An MT System between Closely Related Languages. In *Proc. of the 3rd EACL*, pages 113–117, Copenhagen, Denmark.

[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. Technical Report Research Note 37, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD.

[Hajič and Hladká, 1998] Hajič, J. and Hladká, B. (1998). Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured

Tagset. In *Proceedings of COLING-ACL Conference*, pages 483–490, Montreal, Canada.

[Hajič et al., 2002] Hajič, J., Čmejrek, M., Dorr, B., Ding, Y., Eisner, J., Gildea, D., Koo, T., Parton, K., Penn, G., Radev, D., and Rambow, O. (2002). Natural Language Generation in the Context of Machine Translation. Technical report. NLP WS'02 Final Report.

[Jelinek, 1985] Jelinek, F. (1985). Markov Source Modeling of Text Generation. In *Impact of Processing Techniques on Communications*, pages 569—598. NATO Advanced Study Institute.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical Phrase-Based Translation. In *HLT-NAACL*, pages 127–133, Edmonton, Canada.

[Kučerová and Žabokrtský, 2002] Kučerová, I. and Žabokrtský, Z. (2002). Transforming Penn Treebank Phrase Trees into (Praguian) Tectogrammatical Dependency Trees. *Prague Bulletin of Mathematical Linguistics*, 78:77–94.

[Langkilde, 2000] Langkilde, I. (2000). Forest-Based Statistical Sentence Generation. In *Proceedings of NAACL'00*, Seattle, WA.

[Linguistic Data Consortium, 1995] Linguistic Data Consortium (1995). North American News Text Corpus. LDC95T21.

[Linguistic Data Consortium, 1999] Linguistic Data Consortium (1999). Penn Treebank 3. LDC99T42.

[Linguistic Data Consortium, 2001] Linguistic Data Consortium (2001). Prague Dependency Treebank 1. LDC2001T10.

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

[Melamed, 1996] Melamed, I. D. (1996). A Geometric Approach to Mapping Bitext Correspondence. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing*.

[Minnen et al., 2001] Minnen, G., Carroll, J., and Pearce, D. (2001). Applied Morphological Processing of English. *Natural Language Engineering*, 7(3):207–223.

[Nagao, 1984] Nagao, M. (1984). A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In *Proceedings of the international NATO symposium on Artificial and human intelligence*, pages 173–180, New York, NY, USA. Elsevier North-Holland, Inc.

[Och, 2002] Och, F. J. (2002). *Statistical Machine Translation: From Sigle-Word Models to Alignment Templates*. PhD thesis, RWTH, Aachen, Germany.

[Och and Ney, 2000] Och, F. J. and Ney, H. (2000). Improved Statistical Alignment Models. In *Proc. of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hongkong, China.

[Och and Ney, 2003] Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

[Panevová et al., 2000] Panevová, J., Cuřín, J., Čmejrek, M., Kuboň, V., Peterek, N., Ribarov, K., Vidová-Hladká, B., and Zeman, D. (2000). Počítačová lingvistika ve vztahu k informatice. *Pokroky matematiky, fyziky a astronomie*, 45(3):207–218.

[Papineni et al., 2001] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). Bleu: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176, IBM.

[Ratnaparkhi, 1996] Ratnaparkhi, A. (1996). A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, University of Pennsylvania. ACL.

[Sgall et al., 1986] Sgall, P., Hajičová, E., and Panevová, J. (1986). *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands.

[Stolcke, 2002] Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit. In *Proceeding of 7th International Conference on Spoken Language Processing*, Denver, Colorado.

[Vauquois, 1975] Vauquois, B. (1975). La Traduction Automatique 'a Grenoble.

[Žabokrtský et al., 2002] Žabokrtský, Z., Sgall, P., and Džeroski, S. (2002). Machine Learning Approach to Automatic Functor Assignment in the Prague Dependency Treebank. In *Proceedings of LREC 2002*, volume V, pages 1513–1520, Las Palmas de Gran Canaria, Spain.

[Wu, 1997] Wu, D. (1997). Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Comput. Linguist.*, 23(3):377–403.

[Wu and Wong, 1998] Wu, D. and Wong, H. (1998). Machine Translation with a Stochastic Grammatical Channel. In *COLING-ACL*, pages 1408–1415.

[Xia and Palmer, 2001] Xia, F. and Palmer, M. (2001). Converting Dependency Structures to Phrase Structures. In *Proceedings of HLT 2001, First International Conference on Human Language Technology Research*, San Francisco.

[Yamada and Knight, 2001] Yamada, K. and Knight, K. (2001). A Syntax-based Statistical Translation Model. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530, Morristown, NJ, USA. Association for Computational Linguistics.

[Yamada and Knight, 2002] Yamada, K. and Knight, K. (2002). A Decoder for Syntax-based Statistical MT. In *ACL*, pages 303–310.