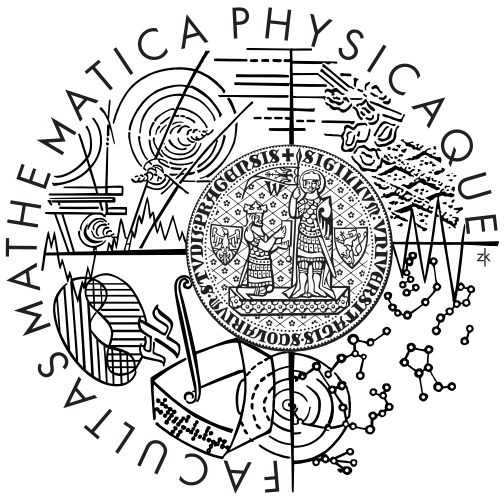
Charles University in Prague
Faculty of Mathematics and Physics

DIPLOMA THESIS



Martin Popel

Ways to Improve the Quality of English-Czech Machine Translation

Institute of Formal and Applied Linguistics

Supervisor: Ing. Zdeněk Žabokrtský, Ph.D.
Study programme: Computer Science
Study field: Mathematical Linguistics

Prague, 2009

I would like to thank my supervisor Zdeněk Žabokrtský for many inspiring ideas, encouragement, helpful comments on this thesis, for having patience with me; and above all for showing me the great world of Machine Translation where it is a joy to combine insightful linguistic knowledge with mighty statistical methods and machine learning.

I also thank my grandfather Frank for comments on English grammar and my dear Markétka for her care and love.

I certify that this diploma 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, July 23, 2009 Martin Popel

Contents

List of Tables

Title: Ways to Improve the Quality of English-Czech Machine Translation
Author: Martin Popel
Department: Institute of Formal and Applied Linguistics
Supervisor: Ing. Zdeněk Žabokrtský, Ph.D.
Supervisor's e-mail address: zabokrtsky@ufal.mff.cuni.cz

Abstract:
This thesis describes English-Czech Machine Translation as it is implemented in TectoMT system. The transfer uses deep-syntactic dependency (tectogrammatical) trees and exploits the annotation scheme of Prague Dependency Treebank.

The primary goal of the thesis is to improve the translation quality using both rule-base and statistical methods. First, we present a manual annotation of translation errors in 250 sentences and subsequent identification of frequent errors, their types and sources. The main part of the thesis describes the design and implementation of modifications in the three transfer phases: analysis, transfer and synthesis. The most prominent modification is a novel approach to the transfer phase based on Hidden Markov Tree Models (a tree modification of Hidden Markov Models). The improvements are evaluated in terms of BLEU and NIST scores.

Keywords: machine translation, tectogrammatical layer, TectoMT

Název práce: Možnosti zlepšení strojového překladu z angličtiny do češtiny
Autor: Martin Popel
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: Ing. Zdeněk Žabokrtský, Ph.D.
e-mail vedoucího: zabokrtsky@ufal.mff.cuni.cz

Abstrakt:
Tato diplomová práce popisuje strojový překlad z angličtiny do češtiny implementovaný v systému TectoMT. Překlad je založen na transferu přes tektogramatickou rovinu a využívá anotační schéma Pražského závislostního korpusu.

Prvotním cílem práce je zlepšení kvality překladu za pomoci pravidlového přístupu i statistických metod. Nejprve je popsána ruční anotace překladových chyb ve vzorku 250 vět a následná analýza častých typů chyb a jejich příčin. Hlavní část textu pak popisuje návrh a provedení úprav, které vedly k vylepšení tří fází překladu: analýzy, transferu a syntézy. Nejvýraznější inovací je využití stromové modifikace skrytých Markovových řetězců (Hidden Markov Tree Models) ve fázi transferu. Dosažené zlepšení je kvantitativně vyhodnoceno pomocí metrik BLEU a NIST.

Klíčová slova: strojový překlad, tektogramatická rovina, TectoMT

Chapter 1

Introduction

TectoMT je nyní experimentální systém,
který je překonán state-of-the-art MT systémy
otevřených zdrojových Mojžíšů.
TectoMT, 2009[1]

Machine translation (MT) is gaining more and more importance in the contemporary world. There are many approaches to MT, which are by tradition classified into two paradigms: rule-bases and statistical.

Classical rule-based MT systems make use of linguistic knowledge (grammars, dictionaries, rules written by human experts), but they use no information learned automatically from corpora. The translation usually comprises three phases: analysis, transfer and synthesis. MT systems can be further classified according to the level of language abstraction used for the transfer – some systems perform shallow analysis, some perform deep (or rich) analysis.[2] The advantage of deeper analysis is that the transfer should be easier and when building a system for translation between more than two languages, the analysis and synthesis can be shared for all language pairs with the given source or target language respectively.

Classical statistical MT systems make use of large-scale human-translated parallel corpora and monolingual corpora, but almost no linguistic knowledge. This has the advantage that the same system can be used for any pair of languages for which there are enough training data available.

In recent years, there is a tendency to exploit linguistic knowledge to a greater extend to improve the performance of statistical MT systems. On the other hand, rule-based or syntax-based MT systems incorporate more statistical methods (rules can be automatically learned from parallel corpora, stochastic taggers and parsers are used etc.). This results in a convergence of both the paradigms – it seems that modern high-quality MT systems will use statistical methods as well as linguistic knowledge, so the contemporary rivalry between statistical MT and rule-based MT will become irrelevant.

[1] This motto was translated to Czech by our system. The source English sentence is *"TectoMT is currently an experimental system, which is outperformed by state-of-the-art MT systems such as open source Moses"*. More translation samples can by found in Appendix E.

[2] Shallow syntax structure of a sentence can be represented either by constituency trees (e.g. Wang et al., 2007) or dependency trees (e.g. Quirk et al., 2005). For deep syntax structure it is more common to use dependency trees such as tectogrammatical trees from Functional Generative Description theory (Sgall, 1967), normalized trees in the ETAP-3 system (Boguslavsky et al., 2004), or logical form structures (Menezes and Richardson, 2001).

This thesis describes improvements of English-Czech translation system called TectoMT. TectoMT is one of the promising MT systems that combine statistical techniques and linguistic knowledge. It aims at transfer on the so-called tectogrammatical layer, which is a layer of deep syntactic dependency trees.

Currently, it is an experimental system, which is outperformed by state-of-the-art MT systems such as Google Translate[3] or open source Moses (Koehn et al., 2007). On the other hand, it has a potential to solve translation problems common to n-gram based systems and moreover, the whole process of translation is adequate also from the linguistic point of view.

1.1 Our goals

Our main goal was to improve the quality of translation in TectoMT. In order to do so, we designed these tasks:

- to investigate thoroughly the whole process of translation in TectoMT,
- to manually annotate errors in 250 sentences translated by the version of TectoMT, that participated in WMT 2009 Shared Task,[4]
- to identify the most prominent errors in translated output and their source,
- to design and implement methods that repair some of these errors.

1.2 Structure of the thesis

In Chapter 2, we describe the related work on machine translation based on tectogrammatics. We also briefly introduce key points of Prague Dependency Treebank annotation scheme and TectoMT framework.

In Chapter 3, we present the manual annotation of translation errors. We explain by examples how the annotations were done and we discuss the results of the analysis of translation errors.

Chapter 4 should be read before the following chapters, because it defines the baseline system and describes how to interpret the evaluation of our modifications.

The main part of this thesis comprises the description of modifications we have done in the analysis (Chapter 5), transfer (Chapter 6) and synthesis (Chapter 7) phases. A few improvements we have done, but that are not covered in the previous chapters, are summarized in Chapter 8

Finally, Chapter 9 concludes with a recapitulation of the achieved results.

[3] `http://translate.google.com`

[4] Fourth Workshop on Statistical Machine Translation, `http://www.statmt.org/wmt09/translation-task.html`

Chapter 2

MT Using Tectogrammatics

Ptáci v bederním hejnu spolu.
TectoMT, 2009[1]

2.1 Related work

There are several approaches to MT that make use of tectogrammatics.

Synchronous Tree Substitution Grammars were introduced by Hajič et al. (2002), formalized by Eisner (2003) and subsequently used for Czech-English MT (Čmejrek, 2006) and English-Czech MT (Bojar, 2008).

Our work is based on a different approach – tectogrammatical transfer blocks implemented in TectoMT framework (Žabokrtský et al., 2008; Bojar et al., 2009).

In this thesis, we will use terminology adopted from the Functional Generative Description theory (Sgall, 1967), which has been further elaborated and implemented in the Prague Dependency Treebank (PDT, Hajič et al. (2006)).

We give here only a very brief summary of the key points. There are three layers of annotation in PDT:

- **morphological layer (m-layer)**
  Each sentence is tokenized and each token is annotated with a lemma and morphological tag. For details see Zeman et al. (2005).

- **analytical layer (a-layer)**
  Each sentence is represented as a shallow-syntax dependency tree (a-tree). There is one-to-one correspondence between m-layer tokens and a-layer nodes (a-node). Each a-node is annotated with the so-called analytical function, which represents the type of dependency relation to its parent (i.e. its governing node). For details see Hajičová et al. (1999).

- **tectogrammatical layer (t-layer)**
  Each sentence is represented as a deep-syntax dependency tree (t-tree). Autosemantic (meaningful) words are represented as t-layer nodes (t-nodes). Information conveyed by functional words (such as auxiliary verbs, prepositions and subordinating conjunctions) is represented by attributes of t-nodes. Most important attributes of t-nodes are: tectogrammatical lemma,[2] functor and a set of grammatemes.

[1] *Birds of a feather flock together.*

[2] Tectogrammatical lemmas (t-lemmas) are usually identical to morphological lemmas (m-lemmas). However, some words have a special t-lemma, which has no counterpart among morphological lemmas, or they have a t-lemma that corresponds to the m-lemma of a different word. For example all personal pronouns have t-lemma #PersPron and deadjectival adverbs have t-lemma

Edges in t-trees represent a linguistic dependency except for several special cases, most notable of which are paratactic structures (coordinations). In these cases there is a difference between the *topological parent* of a node (i.e. the parent as it is saved in the tree) and the *effective parent* (i.e. the governing node in a linguistic sense). Analogously, there is a notion of *topological children* and *effective children*. For details see Mikulová et al. (2006).

2.2 TectoMT

TectoMT was developed by Zdeněk Žabokrtský and other members of the Institute of Formal and Applied Linguistics. It is a highly modular software framework for Natural Language Processing, implemented in Perl programming language under Linux.[3] We give here only a very brief summary of the key points, for details see TectoMT Developer's Guide.[4]

The basic units of code in TectoMT are called *blocks*. Each block should have a well-documented, meaningful, and (if possible) also linguistically interpretable functionality.

Blocks are used to process *documents* (which can be saved in `tmt` format). Documents consist of a sequence of *bundles*. Each bundle represents one sentence as a set of trees. The trees can be classified according to:

- level of language description (M=m-layer, A=a-layer, T=t-layer),
- language (English, Czech),
- indication whether the sentence was created by analysis (S=source) or by transfer/synthesis (T=target).

TectoMT trees are denoted by the three coordinates: for example, analytical layer representation of an English sentence acquired by analysis is denoted as `SEnlishA`. This naming convention is used on many places in TectoMT: for naming blocks, for node identifier generating, etc. In this thesis, we will for simplicity use mostly short names of blocks, so e.g. instead of full name `SEnglishA_to_SEnglishT::Assign_grammatemes` we will write only `Assign_grammatemes`.

A sequence of blocks is called *scenario*. Applications (end-to-end tasks) are processed by applying a scenario on a set of documents. In scenarios, we can specify also parameters for individual blocks. Using parameters we can define, for instance, which model should be used for parsing.

2.2.1 Formemes

The tectogrammatical layer used in TectoMT slightly differs from how it was defined in PDT. The differences are motivated pragmatically with regards to the needs of MT. One of most remarkable differences is addition of a new t-layer attribute called formeme.

of the corresponding adjective. For simplicity, we will use the term *lemma* to refer generally to both t-lemma and m-lemma, if there is no need to distinguish.

[3]Several tools in form of binary application or a Java application are integrated to TectoMT using Perl wrapper modules.

[4]`http://ufal.mff.cuni.cz/tectomt/guide/guidelines.html`

Formeme specifies in which morphosyntactic form a t-node was expressed in the surface sentence shape (or will be expressed, in case of synthesis). The set of formemes is generally language specific, but some formemes are applicable for both Czech and English. Instead of formal definition, we will give some examples of English formemes, as cited in the paper where they were introduced (Žabokrtský et al., 2008):

- n:subj – semantic noun in subject position,
- n:for+X – semantic noun with preposition *for*,
- n:X+ago – semantic noun with postposition *ago*,
- v:because+fin – semantic verb as a subordinating finite clause introduced by *because*,
- v:without+ger – semantic verb as a gerund after *without*,
- adj:attr – semantic adjective in attributive position,
- adj:compl – semantic adjective in complement position.

2.2.2 Translation scenario outline

Now, we briefly describe the whole process of English-Czech translation implemented in TectoMT. In Chapters 5 (Analysis), 6 (Transfer) and 7 (Synthesis) we give more details about each step.

Analysis from a raw text to m-layer

Four tasks have to be done to create the m-layer from a raw text: segmentation to sentences, tokenization, PoS (Part of Speech) tagging and lemmatization. The first task is not discussed in this thesis, because we have not changed the original TectoMT implementation of segmentation. Tokenization is discussed in Section 5.1, tagging in Section 5.2 and lemmatization in Section 5.3.

It should be noted that this is not the only possible order in which the beforementioned four tasks could be done. Some applications prefer tokenization before segmentation. In languages with rich morphology like Czech, taggers usually need full morphological analysis of tokens as input, which means that lemmatization is done before tagging and taggers chose from the set of possible lemma&tag pairs. Moreover, some tools do more tasks at once, for example `Lingua::EN::Tagger`[5] does tokenization and tagging in one step.

Analysis from m-layer to a-layer

For the difficult task of dependency parsing we use McDonald's Maximum Spanning Tree Parser, which builds a-layer trees (Section 5.4). However, nodes of these trees do not have filled the attribute `afun` (analytical function), so the task of assignment of analytical functions is realized afterwards in a separate block (Section 5.5).

[5] `http://search.cpan.org/perldoc?Lingua::EN::Tagger`

Analysis from a-layer to t-layer

Analytical trees are converted to tectogrammatical ones: Functional words (such as prepositions, subordinating conjunctions, articles etc.) are removed. The information conveyed in these words (and in word order and some morphological categories) is encoded into t-layer attributes (grammatemes, functor, semantic PoS, formeme and others). This step is described in Section 5.6.

Transfer from English t-layer to Czech t-layer

English tectogrammatical trees are translated to Czech tectogrammatical trees. Probabilistic dictionaries provide n-best lists of lemmas and formemes. The combination that is optimal for the whole tree is selected using the HMTM (Hidden Tree Markov Models) transfer block. Additional rule-based blocks are used to translate other t-layer attributes. This step is described in Chapter 6.

Synthesis from Czech t-layer to a raw text

In this step Czech analytical trees are created from the tectogrammatical ones (auxiliary nodes are added), but the process of synthesis continuously goes on (morphological categories are filled, word forms are generated), so in the last block, the sentence if generated by simple flattening the tree and concatenating the word forms. This step is partly described in Chapter 7.

Chapter 3

Manual Annotation of MT Errors

Velcí řečníci jsou malí vrazi.
TectoMT, 2009[1]

The need for evaluation of MT output is obvious. The task in general is very difficult and there are many approaches with different goals and requirements.

Popular MT metrics such as BLEU (Papineni et al., 2002), NIST (Doddington, 2002) or WER can be considered as error analyses whose only output is a single number hopefully indicating the quality of translation. The only requirement for such automatic metrics is a set of reference translations. However, also the goals are limited. Automatic metrics are invaluable for frequent measuring of improvements within an MT system during the development process. They can be also used for comparison of different systems although the correlation with human judgements is controversial (e.g. Callison-Burch et al., 2008; Homola et al., 2009).

Manual analysis is expensive and time-demanding, but it can identify types and sources of errors. This knowledge is very helpful for developers of MT systems, that perform transfer on some level of abstraction which is higher than simple phrase-to-phrase.

3.1 Related work

There are many papers about manual evaluation of MT errors (e.g. Koehn and Monz, 2006), but mostly they are limited to scoring *fluency* and *adequacy*. Some papers (e.g. Hopkins and Kuhn, 2007) use manual analysis based on some form of *edit distance*, i.e. the number of editing steps (of various types) needed to transform the system output into an acceptable translation.

One of the most detailed manual analysis frameworks is the RWTH[2] Error Classification Scheme (Vilar et al., 2006), which classifies errors into a hierarchical structure depicted in Figure 3.1.

[1] *Great talkers are little doers.*
[2] Rheinisch-Westfälische Technische Hochschule Aachen

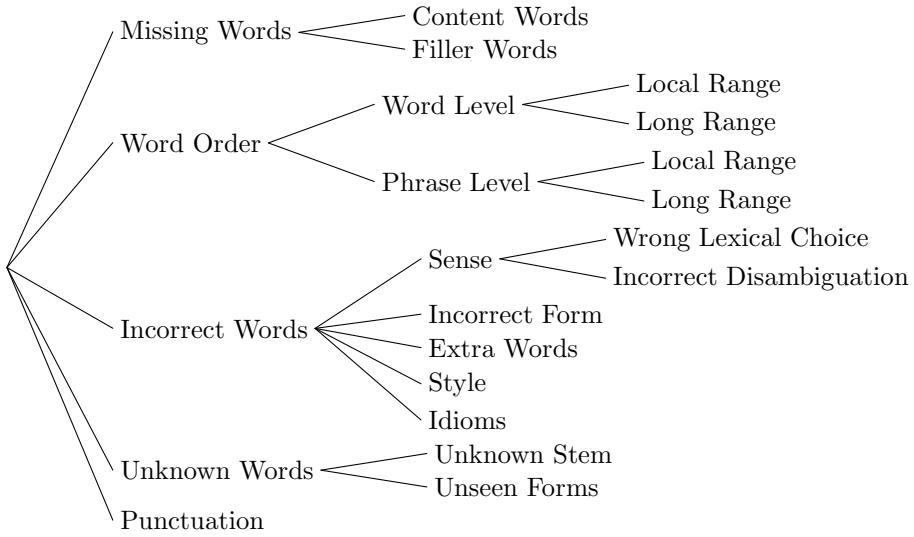


Figure 3.1: *Classification of translation errors, adopted from Vilar et al. (2006).*

3.2 Annotation framework

3.2.1 Overview

Our proposed error analysis framework is similar to that of Vilar et al. (2006), but instead of three hierarchical categories (*type, subtype* and *sub-subtype*) we have five categories: *seriousness, type, subtype, source* and *circumstances*.

Errors are marked in text by *error markers* which the annotator simply inserts in front of relevant words. If needed, one word can have more than one error marker. Every error marker describes all the five categories of an error. Possible values for these categories are summarized in Tables 3.1 – 3.3 and the main points of the framework are explained in the following examples.

The general idea of having each error marked in text and classified seems language and system independent. However, this does not hold for the actual values of classes and annotation guidelines.

Source		Description
Analysis	`tok`	tokenization errors
	`tagger`	PoS tagging errors
	`lem`	lemmatization errors
	`parser`	errors associated with parsing and related tasks (building a-layer from m-layer)
	`tecto`	tecto-analysis errors (building t-layer from a-layer)
Transfer	`x`	errors caused by the assumption of t-tree isomorphism (which is currently required in the TectoMT translation)
	`trans`	other errors associated with the transfer (translation of lemmas, formemes, grammatemes, noun gender assignment,...)
	`syn`	synthesis errors (generation of text from the target t-layer)
	`?`	source unknown

Table 3.1: *Possible values for sources of errors*

Circumstance	**Description** – errors associated with ...
`ne`	named entity
`num`	numbers
`coord`	coordination or apposition

Table 3.2: *Possible values for circumstances of errors*

Type	Subtype	Description	
lex		wrong lemma	serious errors by default
	asp	wrong aspect of a verb	
	se	wrong reflexivity, e.g. t-lemma *stát_se* instead of *stát* or vice versa	
	neT	named entity translated, but should remain unchanged	
	neU	named entity unchanged, but should be translated, because original form is not acceptable in the target language	
	neX	assumed named entity unchanged, but should be translated, because it is not a named entity actually (SRC: *Bill was approved.* REF: *Návrh zákona byl schválen.* TST: *Bill byl schválen*)	
	com	unchanged word due to an unprocessed compound word (e.g. *middle-aged*)	
	unk	unchanged (possibly out of dictionary) word other than neU, neX and com	
form		wrong formeme	
	ze	formeme v:že+fin instead of v:rc or v:fin	
gram		wrong grammateme and related errors	
	gender	wrong grammateme of gender (feminine, neuter, masculine animate, masculine inanimate)	
	person	wrong grammateme of person (first, second, third)	
	number	wrong grammateme of number (singular, plural) except cases classified as numberU (see below)	
	tense	wrong grammateme of tense (simultaneous, preceding, subsequent)	
	mod	wrong verbal, deontic, dispositional or sentence modality	
	deg	degree of comparison (positive, comparative, superlative)	
	neg	negation (affirmative, negative)	
	svuj	switched m-lemma *svůj* with *jeho*, *její*, ...	
	numberU	number unchanged, but should be changed e.g. *Ministry of Finance*(sg) → *Ministerstvo financí*(pl)	
phrase		phrases, idioms, deep syntactic structures that can not be translated node-to-node.	
miss		missing words that are not covered by the types above	
extra		superfluous words that are not covered by the types above	
punct		punctuation errors	minor
	brack	missing, superfluous or displaced brackets	
order		wrong word order (except cases classified as punct)	
case		switched upper/lower case	

Table 3.3: *Possible values for types and subtypes of errors*

3.2.2 Examples

In each example in the following paragraphs, there is an English source sentence (SRC), its reference translation made by professional human translator (REF) and the output of TectoMT (TST). In addition we will introduce also an *aimed translation* (AIM), which is a correct or at least acceptable translation but it is also theoretically achievable for TectoMT.[3] In general, aimed translations are more literal than the reference and may also be stylistically inferior.

Introduction of Types and Sources of errors

Example 1.
SRC: *The vote on it will take place at the beginning of next week.*
REF: *Hlasovat se o něm bude počátkem příštího týdne.*
AIM: *Hlasování o tom se bude konat na začátku dalšího týdne.*
TST: *Hlas o tom vezme místo na začátku dalšího týdne.*

In the reference translation, the English noun *vote* is translated as Czech verb infinitive *hlasovat*. However, this part of speech change is not necessary – it could have been translated also as Czech noun *hlasování*, as it is done in the aimed translation. Also, *počátkem* and *na začátku* are almost synonyms just as *příštího* and *dalšího* in this context.[4]

Nevertheless, in the TectoMT output, there is an unacceptable translation of *vote* (meaning polls) – *hlas* (meaning voice or suffrage). Although it has a common root with the correct translation *hlasování*, these are different lemmas.

The second error in the TectoMT output is the translation of the phrase *take place*. These two words (*take* being a governing node for *place*) cannot be translated independently, but this holds for many other word couples. The real problem here is that two t-nodes on the English t-layer should be translated as one t-node on the Czech t-layer.[5] This breaks the presumption of isomorphism between a source and target tree and cannot be translated correctly with the original version of TectoMT.

When marking the two errors in text we use the so-called *error markers* prefixed to the words in question.

TST: `lex::` *Hlas o tom* `phrase-x::` *vezme místo na začátku dalšího týdne.*

In markers we distinguish the type of error: `lex` means wrong lemma, `phrase` means wrong phrase (only the head of the phrase is marked). Also, we want to distinguish the source of error, i.e. look into the TectoMT internals and find the "culprit". Since it has been shown that the transfer step is the most common source of errors, we have decided to make annotations briefer: if there is no source specified in an error marker, it is the transfer by default. Both errors in the above example

[3] The aimed translation is either already in the search space of TectoMT (e.g. in the lemma n-best list) or we think it should be there in future. Of course, usually there are more aimed translations – we choose the one that is most similar to the current TectoMT output.

[4] *Příští* means rather *following*, whereas *další* preserves more meanings of *next* (*another*). Although in Example 1 it is appropriate to use more specific *příští* (*following*), generally we would like MT to preserve ambiguities if possible.

[5] T-node with lemma konat_se and with a grammateme indicating third person, singular, future tense, is synthesized into three words *se bude konat*.

come from the transfer phase. However, the source of the second error (*take place*) is of another kind than the source of the first error (*vote*). A source called `x` stands for errors caused by unrealized presumption of isomorphic t-trees.

Example 2.
SRC: *memory card*
REF: *paměťová karta*
TST: *karta* `lex::form::` *paměti*

In this context, the correct translation of *memory* with formeme `n:attr` is Czech adjectival lemma *paměťový* with formeme `adj:attr`, but TectoMT incorrectly choose noun lemma *paměť* with formeme `n:2`. So in this case also the formeme is wrong and we will mark it with the `form` marker.[6]

Introduction of Seriousness of errors

Example 3.
SRC: *That is, the members of congress have to complete some details of the agreement before they can make the final version of the law public and vote on it.*
REF: *Kongresmani totiž musejí dokončit některé detaily dohody, než budou moci zveřejnit finální podobu zákona a hlasovat o něm.*
AIM: *Tedy členové Kongresu musí dokončit některé podrobnosti dohody, než mohou zveřejnit konečnou verzi zákona a hlasovat o ní.*
TST: *To je, členy Kongresu musí dokončit některé podrobnosti o dohodě, mohou udělat konečnou verzi zákon veřejnosti a hlasu na tom.*

English phrase *that is* should not be translated literally in this context. However, even if it is translated literally (*to je*), it does not make the sentence unintelligible in Czech, moreover it could be considered grammatical. In other words, this error is less serious than other ones and if we decide to mark this error in text, we should also mark its *seriousness*. Although one can imagine quite a long scale ranging from almost correct constructions with minor stylistic slip-ups to fatal errors, we have introduced only two values: *serious* and *minor*. It is fully up to an annotator to choose depending on whether the error is essential for understanding the meaning or not. The least serious errors (such as *další* instead of *příští* in Example 1) are not marked at all. Since most of errors with type `lex` or `phrase` (and other types except `punct`, `order` and `case`) are serious, it is taken as a default value. Minor errors are marked in text by adding `0` to the type, e.g: *To* `phrase0-x::` *je*.

Next error is a rather unusual and tricky Czech speciality. Instead of correct plural form *členové*, there is used another form of the same lemma – *členy*. This may look like choosing accusative instead of nominative. In fact, it is only an inanimate form of nominative instead of animate. In TectoMT both the forms – animate and inanimate – share the same lemma, but can be distinguished by the value of grammateme *gender*. Therefore, this error is marked as `gram-gender`.

Phrases *podrobnosti o dohodě* (*details about the agreement*) and *podrobnosti dohody* (*details of the agreement*) differ only in formeme of the dependent word (`n:o+6`

[6] In early experiments, we tried to mark only the primary or the more serious error of `lex` and `form`. However, we did not succeed in specifying consistent rules for identifying which one is more serious or primary. Lemma and formeme are usually very closely related.

instead of n:2). If considered as an error at all, then this is definitely only a minor error, since the meaning is preserved.

Make X public is a phrase that should be translated as *zveřejnit X* – two English t-nodes correspond to one Czech t-node, so the error is marked with `phrase-x`.

Phrase *before they can* should be translated as *než mohou* or *než budou moci*. The conjunction *before* is not present on the t-layer as an independent node. It is embodied in the formeme v:before+fin, which should be translated to the formeme v:než+fin. Therefore, the absence of the conjunction *než* on the surface should not be marked with the `miss` type. Instead, the governing verb should be marked with `form`. The source of this error actually lies in the phase when English t-layer is built from a-layer – instead of v:before+fin there is incorrect v:fin. This source is called `tecto`.

The rest of the errors in Example 3 is caused by wrong part of speech tagging. Instead of *make the final version of the law public*$_{JJ}$ *and vote*$_{VBZ}$ *on it*, the tagger produced *make the final version of the law public*$_{NN}$ *and vote*$_{NN}$ *on it*. Due to these tagger errors, also the subsequent phases (parser, tecto-analysis and transfer) went wrong.

To conclude, the annotation of Example 3 is:

TST: *To* `phrase0-x::` *je,* `gram-gender::` *členy Kongresu musí dokončit některé podrobnosti o* `form0::` *dohodě, mohou* `form-tecto::` `phrase-x::` *udělat konečnou verzi* `form-tagger` *zákon veřejnosti a* `lex-tagger::` `form-tagger` *hlasu na* `form-tagger::` *tom.*

Introduction of Circumstances of errors

The chosen classification of error types is not the only one possible. For example some errors are associated with coordination phrases (which are hard to parse correctly), some are associated with numerals (Czech numerals have special rules for morphological cases). However, this alternative classification is orthogonal to the chosen one (`lex`, `form` etc.) – see Table 3.10. Therefore, we have introduced a category called *circumstances* to be able to annotate such alternative classifications.

Categories *type, subtype, seriousness* and *source* have always just one value[7] marked in an error marker. Circumstances can have more values (there can be an error that is associated with a coordination as well as a numeral). However, this happened only once in the analyzed sample.

[7] When no subtype is specified in an error marker, the subtype `other` is taken as the default one.

3.3 Analysis of the Annotated Material

The author of this thesis annotated 250 sentences. Tables 3.4 – 3.7 show numbers of occurrences of both serious and minor errors for each category (*source, type, subtype, circumstances*). Tables 3.8 – 3.10 show contingency tables for serious errors. In the following discussion we will also consider serious errors only, because they are more important and have more occurrences than minor errors.

3.3.1 Sources of errors

As expected, most errors lie in the transfer phase. Only 8% of errors are caused by the unfulfilled presumption of isomorphic t-trees, whereas 56% are other transfer errors that could be repaired within the node-to-node transfer paradigm.[8]

Another notable source of errors is parsing – 21%. As we can see in Table 3.9, about 39% of these parsing errors are associated with coordinations. Also other statistics indicate that the parsing of coordinations is a significant problem in TectoMT: There were 89 coordinations in the test data and more than half of them is parsed incorrectly which results in 1.13 serious errors per coordination on average.

3.3.2 Types and subtypes of errors

The most common type of error is a wrong choice of lemma (`lex` = 38%), followed by a wrong choice of formeme (`form` = 36%) and grammateme (`gram` = 11%).

Several subtypes of `lex` were classified (compound words, errors associated with named entities or reflexivity of lemmas), but most `lex` errors remain unclassified. We have not carried out any subclassification of `form` errors except registering problems with the Czech formeme *v:že+fin*. Among subtypes of `gram`, the most problematic one is the choice of correct gender[9] (26%) and number (23%).

[8] This finding is for us – TectoMT developers – very important. Of course, we are aware of the cases that are not possible to translate within the node-to-node paradigm and we plan to solve them in TectoMT in future. However, those 8% is a number small enough, that we primarily concentrate on the rest of errors.

[9] It is well known that when translating from English to Czech, gender must be sometimes guessed from context, since English does not indicate gender for verbs, but Czech does.

Source	#serious		#minor		#both	
trans	684	55.9%	161	67.4%	845	57.8%
parser	262	21.4%	38	15.9%	300	20.5%
x	95	7.8%	14	5.9%	109	7.5%
tecto	62	5.1%	6	2.5%	68	4.6%
tagger	37	3.0%	0	0.0%	37	2.5%
?	35	2.9%	10	4.2%	45	3.1%
syn	35	2.9%	7	2.9%	42	2.9%
tok	13	1.1%	3	1.3%	16	1.1%
lem	1	0.1%	0	0.0%	1	0.1%
total	**1224**	100%	**239**	100%	**1463**	100%

Table 3.4: *Distribution of translation errors with respect to their sources*

Type	#serious		#minor		#both	
lex	470	38.4%	74	31.0%	544	37.2%
form	443	36.2%	38	15.9%	481	32.9%
gram	137	11.2%	14	5.9%	151	10.3%
phrase	69	5.6%	12	5.0%	81	5.5%
extra	30	2.5%	6	2.5%	36	2.5%
order	29	2.4%	35	14.6%	64	4.4%
punct	27	2.2%	37	15.5%	64	4.4%
miss	18	1.5%	1	0.4%	19	1.3%
case	1	0.1%	22	9.2%	23	1.6%
total	**1224**	100%	**239**	100%	**1463**	100%

Table 3.5: *Distribution of translation errors with respect to their types*

Type	Subtype	#serious		#minor		#both	
lex							
	other	412	33.7%	69	28.9%	481	32.9%
	se	14	1.1%	1	0.4%	15	1.0%
	com	13	1.1%	0	0.0%	13	0.9%
	neT	10	0.8%	1	0.4%	11	0.8%
	neX	8	0.7%	0	0.0%	8	0.5%
	unk	6	0.5%	0	0.0%	6	0.4%
	neU	4	0.3%	0	0.0%	4	0.3%
	asp	3	0.2%	3	1.3%	6	0.4%
form							
	other	404	33.0%	38	15.9%	442	30.2%
	ze	39	3.2%	0	0.0%	39	2.7%
gram							
	gender	36	2.9%	5	2.1%	41	2.8%
	number	24	2.0%	2	0.8%	26	1.8%
	neg	19	1.6%	0	0.0%	19	1.3%
	svuj	15	1.2%	2	0.8%	17	1.2%
	mod	15	1.2%	3	1.3%	18	1.2%
	other	9	0.7%	1	0.4%	10	0.7%
	numberU	7	0.6%	1	0.4%	8	0.5%
	tense	5	0.4%	0	0.0%	5	0.3%
	deg	4	0.3%	0	0.0%	4	0.3%
	person	3	0.2%	0	0.0%	3	0.2%
punct							
	brack	17	1.4%	7	2.9%	24	1.6%
	other	10	0.8%	30	12.6%	40	2.7%

Table 3.6: *Distribution of translation errors with respect to their subtypes*

Circumstance	#serious		#minor		#both	
coord	101	8.3%	16	6.7%	117	8.0%
ne	82	6.7%	22	9.2%	104	7.1%
num	33	2.7%	7	2.9%	40	2.7%
none	1009	82.4%	194	81.2%	1203	82.2%
total	**1225**		**239**		**1464**	

Table 3.7: *Distribution of translation errors with respect to their circumstances*

	lex	form	gram	phrase	extra	order	punct	miss	case
trans	408	219	41		4	8		3	1
parser	26	145	41	2	9	8	26	5	
x	6	4	2	67	13			3	
tecto	6	23	30					3	
tagger	15	18	3			1			
?	1	23	5		2	1	1	2	
syn		5	15		2	11		2	
tok	7	6							
lem	1								

Table 3.8: *Distribution of serious translation errors with respect to their sources and types*

	coord	ne	num	none
trans		53	10	621
parser	101	12	1	149
x		5		90
tecto		5	1	56
tagger				37
?			17	18
syn			4	31
tok		7		6
lem				1

Table 3.9: *Distribution of serious translation errors with respect to their sources and circumstances*

	coord	ne	num	none
lex	5	46	1	418
form	61	29	31	323
gram	17	3		117
phrase	1			68
extra	5	2		23
order	5	1	1	22
punct	7			20
miss				18
case		1		

Table 3.10: *Distribution of serious translation errors with respect to their types and circumstances*

Chapter 4

Evaluation Methodology

Dobré je feťácké vejce jako činný pták.
TectoMT, 2009[1]

4.1 Baseline

To evaluate the effect of our modifications we need a baseline system – a version of TectoMT without the modifications. Since TectoMT is a team work, it is not always easy to separate our modifications from modifications done by other developers. Thanks to the version control system[2] it is possible to find easily various information about every committed modification in TectoMT (author, date, differences between versions etc.), but there is no single date or revision number that could be considered the baseline version. However, the version of TectoMT (with revision number 1156) whose results were submitted to WMT 2009 Shared Task in December 2008 (Bojar et al., 2009) is a point after which almost no modifications committed by other developers influenced the translation scenario presented in this thesis. On the other hand, only a minority of our modifications was committed before the date. Only one of those "pre-WMT09" modifications has a notable impact on the translation quality and can be easily switched with the original implementation – our re-implementation of English lemmatization.

Therefore, our baseline system is the TectoMT version submitted to WMT 2009 with the exception of lemmatization, which is from the revision 860. When evaluating our modifications in Sections 5 – 7, we will refer to this baseline system as "original implementation" or "original version of TectoMT".

4.2 Evaluate improvements or impairments?

Aside from evaluating the total difference of BLEU score between our new version of TectoMT translation and the original one, we want to evaluate also the effect of each modification separately. There are two possible ways how to measure the effect of one particular modification:

- Take the original version of TectoMT with the original scenario, and measure the *baseline BLEU score*. Then substitute one or more blocks in the original scenario with our new implementation equivalents and measure the difference of the new score and the baseline score. Hopefully, this difference will be positive, which should be interpreted as an improvement.

[1] *As good be an addled egg as an idle bird.*
[2] TectoMT SVN repository, `https://svn.ms.mff.cuni.cz/projects/tectomt_devel`

- Take our new version of TectoMT with the new scenario, which includes all our modifications. Measure the BLEU score – it is the best we can achieve so far, so we will call it *best score*. Then substitute one or more blocks with their old implementation equivalents and measure the difference of the best score and the new score. Again, we hope the difference will be positive, but this time we measured actually an impairment caused by the absence of the modification in question.

The first way is perhaps more intuitive, but it has a substantial drawback. To facilitate programming of new blocks, we have added some functionality also to TectoMT internals (e.g. several methods of `TectoMT::Node`, see Chapter 8). This means that our new blocks that use such new functions (and there are many blocks that do so), cannot be used in the original TectoMT framework.

In the result, with this first way of evaluating particular modifications, we would be able to evaluate only minority of the modifications we have done.

We have selected to use the second way, i.e. we measure an impairment caused by the absence of the modification in question. This value can be loosely interpreted as an improvement caused by the modification, but we must be careful, because there may be "interferences" between some blocks. Therefore, in all experiments presented in evaluation sections of Chapters 5 – 7, where the difference was greater than 0.001 BLEU or 0.01 NIST, we have manually checked also the differences in translated text to ensure that the improvement can be credited to the modification.

4.3 Our test set

We divided the evaluation data of WMT 2009 Shared Task (news-test2009) into two parts:

- First 250 sentences were used for the manual annotation of errors of the original implementation (as presented in Chapter 3).[3]
- The rest (2 777 sentences) is our test set. All tables in this thesis that summarize BLEU and NIST evaluation of our modifications (i.e. all tables whose caption start with *Modifications*) are evaluated on this test set.

4.4 Metrics used

4.4.1 Intrinsic and extrinsic evaluation

Suppose we want to evaluate the effect of substituting one block with another one in our translation scenario. In an extrinsic evaluation we measure performance of the whole translation scenario – in our case with BLEU and NIST scores. In an intrinsic evaluation we measure performance of the two blocks on the given task using some metric suitable for the task. For example, taggers are usually evaluated in the terms of accuracy of chosen PoS tags.

[3] We decided to use the sentences for manual annotation from the WMT 2009 evaluation data, because originally we had planned to compare our manual annotation results with human judgements that were done by volunteers.

Since our aim is to improve the quality of translation, we are primarily interested in the extrinsic evaluation. There can be some modifications with a significant positive effect according to an intrinsic evaluation, but a negligible or even negative effect according to an extrinsic evaluation.

4.4.2 BLEU and NIST scores

We use case-insensitive BLEU and NIST scores based on one reference translation. Although we have created our own implementation of the BLEU evaluation, which can be comfortably used as a TectoMT block (`Print::Bleu`), all results presented in this thesis are measured by the official `mteval-v11b` script.[4] There are already newer versions of the script (v12 and v13) that are able to tokenize Unicode text correctly, but the version v11b is used by WMT 2009 organizers and we want our results to be comparable with results of other MT systems and with last year's results.

Note on BLEU score reliability

Correct opening and closing quotation marks are in Czech „ and “. These symbols are produced by TectoMT as a translation of English “ and ”. However, reference translations in WMT09 training and test data use plain ASCII quotes (").
Statistical MT systems trained on such data produce of course also ASCII quotes. For the purpose of a fair comparison with those systems, we have created a simple block `Ascii_quotes` that converts correct Czech directional quotes to incorrect ASCII ones. We were surprised how great “improvement” can be achieved with this block on our test data – 0.0085 BLEU (0.1757 NIST, see Table 4.1). This fact only confirms that neither BLEU nor NIST can be used as the only measure for comparing two MT systems of different types.

For illustration of the impact see the following sentence. After applying the block `Ascii_quotes`, there are 6 new matching unigrams, 7 bigrams, 7 trigrams and 7 4-grams.[5]

SRC: "*The best years of my life,*" *he said,* "*were in places that were dark, damp and disgusting.*"
REF: "*Nejlepší roky mého života,*” *řekl,* "*byly na místech temných, vlhkých a odporných.*"
TST: *„Nejlepší roky mého života,“ řekl, „byly v místech, která byla temných, vlhkých a chutná.“*
TST: "*Nejlepší roky mého života,*" *řekl,* "*byly v místech, která byla temných, vlhkých a chutná.*"

[4] `http://www.itl.nist.gov/iad/mig/tools/`

[5] How can a change of four symbols result in six new matching unigrams? The official script `mteval-v11b` for measuring BLEU and NIST scores does not use Unicode classes for tokenization, so *„Nejlepší* is treated as one token, whereas "*Nejlepší* as two.

4.5 Tables with evaluation of modifications

Throughout Chapters 5 – 7, we will present tables with an extrinsic evaluation of the described modifications. These evaluations show only the differences in BLEU and NIST scores after substituting our new implementation with the original one. To compute the actual score achieved in the experiment we must subtract the presented difference from the *best score*, which is 0.0981 BLEU and 4.7157 NIST.

When more modifications are presented in one table, we include an additional experiment (called *all above together*) where we apply all the modifications at once. The differences for this experiment are not a summation of the individual differences above – sometimes it is higher (synergy effect), sometimes lower.

To set differences of individual modifications in context, we show in Table 4.1 overall results of the improvements achieved in the three translation phases: analysis, transfer and synthesis. The modification with conversion to ASCII quotes is presented separately.

Modification	diff (BLEU)	diff (NIST)
original analysis	0.0078	0.1363
original transfer	0.0171	0.4189
original synthesis	0.0031	0.0621
all above together	0.0263	0.5954
no `Ascii_quotes`	0.0085	0.1757
all above together	0.0322	0.7422

Table 4.1: *Modifications of analysis, transfer and synthesis*

Chapter 5

Analysis

Slečna palec je slečna miliónu.
TectoMT, 2009[1]

5.1 Tokenization

The goal of tokenization is to split the text into tokens. In most cases this is quite straightforward – splitting is done on whitespaces and punctuation symbols (commas, full-stops, brackets etc.). The hard task is to draw up guidelines that specify how to tokenize debatable cases like:

- numbers (*1 000, 1.000, 1/2,* $\frac{1}{2}$*, 1:2, 1-2, 1–2*),
- contractions (*don't, rock'n'roll, Paul's*),
- compound words (*man-at-arms, Greco-Roman, make-up, ill-treat, black&white*),
- abbreviations (*U. S., U.S., US*),
- dates (*1/1/1990 3pm, 1. 1. 1990 3 p.m., 10:40*),
- other named entities (*O'Doole, Tian'anmen, Sri Lanka*)
- and collocations (*according to, as well as, in the light of, a hell of a lot*).[2]

There are some NLP tasks that prefer "more split" tokens and other that prefer "less split" tokens. Even more diverse preferences can be encountered among linguists' notions of word boundaries. Inconsistencies between particular tokenization styles can be considered a technical detail – if both the styles are well defined, it should be theoretically possible to automatically convert data from one to another as needed. Unfortunately, this is not so easy in practice (especially after parsing is done) and tokenization inconsistencies give rise to severe problems. For example the accuracy of stochastic tools such as taggers and parsers is lower when test data have different tokenization than training data.

The importance of consistent high quality tokenization is emphasized by the fact that it is the first step (after segmentation to sentences) in almost all scenarios and subsequent steps (TectoMT blocks) are highly dependent on it. Concerning scenarios with multiple languages involved (MT, word alignment), another requirement arises – tokenization should be consistent across all the languages, at least for common phenomena like numbers, dates and named entities.

[1] *A miss by an inch is a miss by a mile.*

[2] All cited collocations are treated as one token in British National Corpus, but not in TectoMT.

5.1.1 Original implementation

Block `Penn_style_tokenization` is based on Robert MacIntyre's `sed` script from PennTB.[3] It was adjusted by several TectoMT developers to handle some special characters like typographic apostrophe (’), en-dash (–) or non-breaking space; also few rules were added. The block consists only of simple regular expressions – there are no lists of exceptions (only twenty rules for contractions like *I'm* → *I 'm* or *gonna* → *gon na*).

Strictly speaking the block does not perform only tokenization (implemented as inserting spaces), but also some text normalization. Based on context, ASCII double quotes (") are changed to a pair of single forward or backward quotes (`` and ´´), which is a common computer encoding of opening and closing quotes (“ and ”). Bracket-like characters are converted to special placeholders, so the sequence ( ) [ ] { } becomes -LRB- -RRB- -RSB- -RSB- -LCB- -RCB-. The acronyms stand for Left/Right Round/Square/Curly Bracket. This conversion originates in PennTB, where data are saved in plain text format and bracket symbols are reserved for marking the parse structure.

Bracket placeholders used to cause errors in TectoMT, when some code was not programmed with this convention on mind. At one time they were even left untranslated in the output, but this was soon repaired.

5.1.2 New implementation

Not using bracket placeholders

This change did not influence translation results; its motivation was to improve consistency and maintainability of TectoMT.

We have pruned away the conversion of brackets in `Penn_style_tokenization`. In our opinion, bracket placeholders should be used only in wrapper modules of the tools that need it, hence we have added it to `Tagger::MxPost`. There is no reason for retaining these placeholders in the `form` attribute of m-nodes in TectoMT.

Block `Fix_tokenization`

The second change concerns abbreviations and was implemented only as a proof of concept. Originally, abbreviations like *U.S., a.m., e.g.* were split into four tokens. What is worse, those tokens were sometimes parsed into different phrases or even clauses. Afterwards, it was almost impossible to translate it correctly.

We have added a new block `Fix_tokenization` that merges the abbreviations back into one token, so that they can be translated correctly. Also ordinal number indicators (*st, nd, rd, th*) are merged with the preceding number. The advantage of this implementation compared to directly changing block `Penn_style_tokenization` is that authors of other TectoMT applications can decide whether to use the fixing block in their scenarios or not.

[3] `http://www.cis.upenn.edu/~treebank/tokenization.html`

5.1.3 Evaluation

Compared to other phases of translation, tokenization is one of the easiest tasks and does not cause many errors – only 1.1% according to Table 3.4.

Modification	diff (BLEU)	diff (NIST)
no `Fix_tokenization`	0.0008	0.0105

Table 5.1: *Modifications of the tokenization*
For explanation see Section 4.5.

We have concentrated only on a few tokenization issues, but there remain many others unsolved. For example, numbers with spaces as thousands separator are now split into more tokens as well as Internet domain names (*www.example.org* → *www . example . com*). After consensus on the exact tokenization guidelines for TectoMT, careful reimplementation will be needed.

5.2 Tagging

There are several third-party taggers available in TectoMT: MxPost (Ratnaparkhi, 1996), TnT (Brants, 2000) or Aaron Coburn's `Lingua::EN::Tagger`[4]. In an extrinsic evaluation, best BLEU scores were obtained with Morce tagger (Spoustová et al., 2007), so this is the tagger used in all experiments described in this thesis.

5.2.1 Original implementation

There are two block concerning tagging in the TectoMT translation scenario: `TagMorce` and `Fix_mtags`. The former can be substituted by other tagger blocks and the latter aims at fixing some specific tags by rules. The rules can be classified according to various criteria:

- Is the rule heuristic (so it can in some cases change a correct tag to a wrong one) or reliable?
- Is the rule specific for errors made by a particular parser or is it useful for more parsers?
- Is the rule correcting a real error or only a difference between two tagging guidelines?

To explain the last criterion, let's consider word *later* and quote PennTB PoS tagging guidelines (Santorini, 1990):

> *later* should be tagged as a simple adverb (RB) rather than as a comparative adverb (RBR), unless its meaning is clearly comparative. A useful diagnostic is that the comparative *later* can be preceded by *even* or *still*.
> EXAMPLES:
> I'll get it around sooner/RB or later/RB.
> We'll arrive (even) later/RBR than your mother.

However, this particular guideline goes against the spirit of PDT style m-layer annotation. From the morphological point of view, *later* is always a comparative (either adverb or adjective). The distinction proposed by PennTB guidelines, would be expressed in PDT style by t-layer grammateme gram/degcmp (`comp` is classical comparative and `acomp` is so called absolute comparative) in a more systematic way applicable also to other words than *later*.

Regardless of theoretical matters, tagging *later* as `RB` resulted in translation errors, because the absolute comparative in Czech should be also expressed by comparative forms. Therefore, the block `Fix_mtags` includes a rule that changes `RB` to `RBR` for words *later* and *sooner*.

[4] `http://search.cpan.org/perldoc?Lingua::EN::Tagger`

5.2.2 New implementation

Minor changes of `Fix_mtags`

We have decided to remove all heuristic rules specific for a particular tagger from the block `Fix_mtags` and add them to special blocks (`TagTnT_fix` in case of TnT tagger).

We have added a rule to tag all numbers written with digits as `CD`, because we had encountered taggers treating unknown numbers as some other open category PoS; for example, Morce can (though rarely) give tags `VBP` to some numbers.

According to PennTB PoS tagging guidelines (Santorini, 1990, p. 32), we have added a rule to tag *e.g.* (abbreviation of Latin *exempli gratia*) as `FW`.

Tagging after parsing

One of our aims in TectoMT is to repair errors and inconsistencies as soon as possible. We benefit from the layered design of FGD and PDT and we try to comply with specifications of the layers[5].

Unfortunately, some errors made by taggers can be automatically detected only after the parsing is done. For example, modal verbs in TectoMT a-trees must govern their main verbs. If they are governing a noun but no verb, it may be a tagger error – *We must show*$_{NN}$ *an example.*

We have created a new block `Fix_tags_after_parse` that tries to fix such errors. It uses a file with word forms that can have more than one PoS tag, e.g. *show* `NN VB VBP`. Before we change any tag, we always check this file, whether it is allowed.

Other changes implemented in `Fix_tags_after_parse` include:

- Clause heads are more likely to be verbs than nouns. For example, the sentence *The most expensive basket cost us 10 573 forints.* was correctly parsed with *cost* being the clause head, but incorrectly tagged as `NN`. According to our file with morphology analyses, word *cost* can have tags: `NN`, `VB`, `VBP`, `VBN` or `VBD`. If the right tag was `VB`, `VBP` or `VBN`, we hope the tagger would guess it correctly, so we change the tag to `VBD`. Similarly, if the clause head is tagged as `NNS`, we change it to `VBZ`.

- Mathematical operators (*plus, minus, times*) can be tagged as `CC` in PennTB, but only if there is a real coordination. So e.g. *It falls to minus*$_{CC}$*.* is corrected to *It falls to minus*$_{NN}$*.*

- Some phrasal verb particles (`RP`) are incorrectly tagged as `RB`. Unfortunately, we have not succeeded in specifying any general rules for identification of such cases except explicit listing of the verbs and particles (*shoot up*) and a condition that the particle immediately follows the verb.

[5]The specifications of m-layer, a-layer and t-layer in TectoMT are similar to those in PDT, but there are slight differences. We use the specifications of layers as an interface that ensures interoperability of various blocks. In theory, it is possible to substitute a sequence of blocks that creates m-layer from a raw text with another one and the rest of the scenario (parsing,...) will be still fully operational. In practice, there are small interferences – for example parser A may give best results with tagger X, but parser B with tagger Y.

5.2.3 Evaluation

As we can see in Table 5.2, both modification brought only negligible improvements. Morce is state-of-the-art tagger, so it would be surprising, if we could come up with only a few rules and get much better results. Another reason may be overfitting – rules included in `Fix_tags_after_parse` were constructed to fix problems found in our development test set, but there were probably not many such sentences in WMT09 test set.

Modification	diff (BLEU)	diff (NIST)
original `Fix_mtags`	0.0000	0.0003
no `Fix_tags_after_parse`	0.0000	0.0003
all above together = original tagging	0.0000	0.0007

Table 5.2: *Modifications of the tagging*
For explanation see Section 4.5.

5.3 Lemmatization

The task of lemmatization is to find a lemma for each word form.[6] By tradition, English taggers usually need only word forms and lemmatization is done afterwards. This means that lemmatizers can have also PoS tag on input, which is useful for example for distinguishing: *striking*$_{JJ}$ → striking, *striking*$_{VBG}$ → strike.

This practice (lemmatization after tagging) was adopted also in TectoMT, although it is not in line with the practice used for Czech analysis,[7] which could bring several advantages even for English. Taggers use context of a word to disambiguate possible tags proposed by a morphological analysis and this could be augmented also for disambiguation of possible lemmas. Usually this is already determined by choosing the tag (as with *striking*), but not always. For example, word *travels*$_{NNS}$ meaning *trips* should be lemmatized as travel, but if the meaning is *reports or books about traveling* the correct lemma is travels. Another example with disambiguation of *'s*$_{VBZ}$ → be/have is discussed on page 35.

5.3.1 Original implementation

The original implementation of English lemmatization in TectoMT was based on external tools `morpha` (Minnen et al., 2000) and `ispell`. It was wrapped in a Perl module `PEDT::MorphologyAnalysis` by Jiří Semecký. New process was launched for every analyzed token, so the lemmatization was quite slow even if accelerated by caching 10 000 most frequent word forms. More importantly, there were several other drawbacks of the original implementation which are described in the following paragraphs.

Adjectives and adverbs

Adjectives and adverbs were not handled at all by `morpha` – i.e. comparatives and superlatives were left unchanged. The original Perl wrapper module (`PEDT::MorphologyAnalysis`) implemented only a few heuristic rules, that suffered errors like *later*$_{RBR}$ → lat.

Negative prefixes

In PDT-style lemmatization, words with opposite meaning differing only in a negative prefix have the same lemma, e.g. *unclear* → clear, *disagreement* → agreement. As for comparatives and superlatives, this was not handled in `morpha`, but only in the original Perl wrapper and suffered errors of two types: Few words with negative prefixes were not handled correctly (e.g. *irrecoverable*), whereas quite many

[6] Actually, not only full-fledged words, but all tokens including punctuation symbols are processed in lemmatization. In the actual TectoMT implementation the lemma remains same as the form for punctuation tokens, but this may be improved in future. For example, if there are m-dashes(—) and n-dashes(–) in text, one may want to preserve this distinction in word forms, but assign the same lemma to both tokens. Similarly for English quotation marks (“ and ”) and French guillemets (≪ and ≫).

[7] The output of Czech morphology analyzers is a set of possible lemma&tag pairs for each token. Czech taggers use this analysis as input and chose one pair for each token. So the first part of lemmatization (generating the list of possible lemmas) is done before tagging and the second (disambiguation) is done during tagging.

words were incorrectly analyzed as negative and the assumed prefix was deleted (e.g. *immigration, incoming, Unix-based, intuition, dismissal*).

Tagset

The tagset assumed by `morpha` is CLAWS (Leech et al., 1994), but TectoMT uses Penn Treebank style tags (Santorini, 1990). This is not a major problem, because most rules care only about first one or two letters of a tag: `V` – verb, `NP` – proper noun, `N[^P]` – common noun; and these three assumptions holds for both CLAWS and PennTB tagsets. However, there are few rules where the chosen tagset matters:
CLAWS: lemma(*'d*$_{VHD}$) = have, lemma(*'d*$_{VM0}$) = would
PennTB: lemma(*'d*$_{VBD}$) = have, lemma(*'d*$_{MD}$) = would

Unused information from tags

English tags distinguish plural and singular nouns as well as past participles and infinitive/base verbs.[8] This information remains unused in `morpha`, which results in several errors. For example, *found* is lemmatized always as find, even if the tag was `VB` as in *Romulus wants to found the city of Rome.* Similarly for *rebound* → rebind or *bore* → bear. Errors caused by these homonymous verbs are not so frequent – more common are errors caused by changing singular nouns, especially in connection with foreign words described in the following paragraph.

Foreign words

`morpha` makes use of heuristic rules for handling some declensions of words of Latin origin. However, these rules cause a great number of errors (mostly for words that are not of Latin origin but even for some that are).

rule	correct example	incorrect example
ia→ium	*millennia*→millennium	*Serbia*→serbium
la→lum	*curricula*→curriculum	*papilla*→papillum
i→us	*fungi*→fungus	*Shanghai*→shanghaus
ata→a	*lemmata*→lemma	*Murata*→mura
ae→a	*formulae*→formula	*MAE*→ma

Table 5.3: *Heuristic rules for Latin declensions implemented in* `morpha`

The first two rules in Table 5.3 were overridden in the original wrapper, but only for proper nouns (so errors like *genitalia*$_{NNS}$ → genitalium were not fixed). The third rule caused far most of the errors, since there are many non-Latin words ending with *i* (*sci-fi, Mitsubishi, ASCII, Alexei, mini, chili,...*).

[8] This holds for both PennTB and CLAWS tagset. The only exception are proper nouns, which are tagged in CLAWS always as `NP0`, whereas PennTB distinguish singular proper nouns (`NNP`) and plural proper nouns (`NNPS`).

5.3.2 New implementation

Our new implementation is a pure Perl module called `EnglishMorpho::Lemmatizer`. It uses a set of rules (about one hundred regular expressions inspired by `morpha` (Minnen et al., 2000)) that handle words with regular paradigms.

Words with irregular lemmatization are saved in the file `exception.tsv` and loaded into a Perl hash during the module initialization. The second file needed is a list of words with negative prefixes (actually only beginnings of such words are saved in the file to keep it smaller and more robust). The lemmatization works as follows:

1. Look for the input word form and PoS tag in the hash of exceptions and if succeeded, return the lemma found.

2. In case of adjectives, adverbs and nouns, check whether the word form has a negative prefix and if yes, cut it. Negative prefixes are cut only for common nouns (`NN` and `NNS`), not for proper nouns (`NNP` and `NNPS`). Only prefixes *un, in, im, non-, dis, il, ir* are considered negative, see page 33 for a discussion.

3. Apply rules for regular paradigms for the given PoS tag.

Rule or exception?

In order to keep the file with exceptions reasonably small, we tried to cover by rules as many morphological patterns as possible.[9] We have also included a few lexical rules that are frequent in many derived words. For example, $*men_{NNS} \rightarrow$ *man – there are over 600 such plural nouns in BNC with 88 497 occurrences in total (e.g. *firemen, policewomen, life-boatmen, ex-salesmen*).

List of exceptions

Each exception has a form of tabulator separated columns: word form, PoS tag and lemma. It was obvious that the list of exceptions will be eventually extended over time. To facilitate its creation, coverage checks and maintenance of the list we have distributed the exceptions to several files according to their type. The final file `exceptions.tsv` is then compiled from these source files.

[9] However, we did not include patterns that match less than about ten words. For example, `morpha` includes patterns like **ctoring* → **ctor*, but there are only 7 such verbs in BNC with 122 occurrences in total (e.g. *doctoring, revectoring*). Such patterns can be easily substituted by including these words in the exception list if needed.

The distribution to several source files has three other advantages:

- The system is more modular and configurable – it is possible to substitute one source file by another (for possible usage see the paragraph about British and American consonant doubling below).
- The exception lists can be used also for other purposes than lemmatization – at the moment some source files are also used for the opposite process (generation of word forms from lemmas) and some for `Morce` tagger.[10]
- Some source lists can be saved in a more compact and neat way than simple tab separated word form, PoS tag and lemma for each exception (see the paragraph about irregular verbs for an example).

Counts of exceptions for every source file are presented in Table 5.4. Note that although some files contain only few exceptions, these may be very frequent words (e.g. adjectives *bad, worse, worst*).

source file (type of exception)	**#exceptions**		**example**
`adjectives`	10	0.2%	$best_{JJS}$→good
`contractions`	11	0.2%	$n't_{RB}$→not
`nouns_invariant_ending_with_s`	263	4.9%	$ethics_{NNS}$→ethics
`nouns_irregular`	37	0.7%	$children_{NNS}$→child
`nouns_latin`	50	0.9%	$theses_{NNS}$→thesis
`nouns_plural_es`	4	0.1%	$gases_{NNS}$→gas
`nouns_plural_s`	28	0.5%	$canoes_{NNS}$→canoe
`verbs_cked`	18	0.3%	$mimicked_{VBD}$→mimic
`verbs_doubling`	3438	64.2%	$begging_{VBG}$→beg
`verbs_ending_with_e`	420	7.8%	$tuned_{VBN}$→tune
`verbs_irregular`	971	18.1%	was_{VBD}→be
`verbs_not_ending_with_e`	92	1.7%	$bathed_{VBN}$→bath
`verbs_other`	16	0.3%	am_{VBP}→be
total	5358	100%	

Table 5.4: *Types of lemmatization exceptions*
For every file we have created, we present the total number of exceptions it contains and one of those exceptions as an example.

Types of exceptions

Some source files in Table 5.4 contain words that are exceptions also from the linguistic point of view – irregular verbs, nouns or adjectives. Other source files contain only those words that would not be handled correctly by rules. The typical question

[10]To enable uniform manipulation with all source files, the following technical implementation was chosen. All source files are actually Perl scripts that print the format demanded for `exceptions.tsv` when invoked with option `-a` (like analysis). The format demanded for generation of word forms is printed with option `-g`. Full data included in a source file are printed with option `-d`.

when lemmatizing plural nouns ending with *es* is, whether to cut *es* or just *s* to form a lemma. There is a rule *[vowel]*ses* → *[vowel]se, which works well for *cases, increases, phrases...*, but not for *gases*. So the exception $gases_{NNS}$ → gas must be included in `nouns_plural_es`.

Consonant doubling

The final consonant of some verb lemmas is doubled when forming past tense (`VBD`) and past or present participles (`VBN, VBG`). However, the set of such verb lemmas is different for British and American English. For example, past tense of *travel* is *travelled* in British English and *traveled* in American English. Actually, this makes problems only in generation of word forms from lemmas, not in lemmatization. Nevertheless, there are also opposite cases like British *fulfil* and American *fulfill*. As there are many more verbs that do not allow consonant doubling, the file `verbs_doubling` contains only lemmas that do. If ever needed for generation, this file may have two versions in future – British and American.

Irregular verbs

A special effort was taken to complex elaboration of irregular verbs. Usual lists of irregular verbs in textbooks contain about up to two hundred of them, digital Wiktionary list[11] contains about three hundred. However, these lists are not complete, because new irregular verbs can be created by prefixes (e.g. *quick-freeze, quick-froze, quick-frozen*). To facilitate easy extensions of the list as well as its universal exploitation, we have saved the list in a compact form of a Perl script, whose main part is a hash with all the data. Few lines of the script are shown in Figure 5.1.

```
'burst' => {ps=>'burst',      pp=>'burst',  reg=>2},
'bust'  => {ps=>'bust',       pp=>'bust',   reg=>1},
'buy'   => {ps=>'bought',     pp=>'bought', pref=>'over|under'},
'cast'  => {ps=>'cast',       pp=>'cast',
            pref=>'broad|for|fore|mis|over|re|rebroad|rough|sand-|tele|type'},
'catch' => {ps=>'caught',     pp=>'caught'},
'choose'=> {ps=>'chose',      pp=>'chosen'},
'cleave'=> {ps=>'clove|cleft',pp=>'cloven|cleft',reg=>2,
            com=>'irregular forms means "split apart",
                  regular means "adhere/cling to"'},
```

`ps` means past simple or preterit
`pp` means past participle
`com` means comment
`reg=>1` means that most common are regular forms of this verb (e.g. *bust, busted, busted*)
`reg=>2` means that regular forms of this verb are possible, but not the most common (*bursted* is less frequent than *burst*)

Figure 5.1: *Fragment of the source code* `irregular_verbs.pl` *and a legend*

[11] `http://en.wiktionary.org/wiki/Appendix:Irregular_verbs:English`

We can see that prefix derivations or alternative forms can be added in an easy and intuitive way. The information about preference of a regular vs. irregular form is useful mainly for the generation of forms.[12]

Negative prefixes

As already stated, the detection of words with a negative prefix is not easy, because there are many words that start with *un, in, im, non-, dis, il, ir* without being negative. Actually, we have found that there are more such words than words with a real negative prefix.[13] The file `negation` contains stems of words with a real negative prefix, so the size is reasonably small (1171 stems, 11 KB) while achieving high coverage. For example, there are only 7 stems for the prefix *il*, but the number of covered derived words is much higher (stem *illegal* covers words *illegal, illegals, illegally, illegality, illegalities, illegalize, illegalization, illegalise, illegalisation*).

There are also several open theoretical questions concerning negative prefixes and lemmatization:

- Should also some verbs be handled?
  At the moment, no prefixes are being cut from verbs, even if the meaning is strictly opposite like *disagree*. The reason is that there are only few verbs with a negative prefix in English – negation is usually expressed analytically using forms of *not* (or *doesn't, won't...*).

- Should also other prefixes be handled?
  At the moment, prefixes *anti-, an, a* are left within lemmas, even for words like *anti-war, anastigmatic, asexual*.[14] The argument for not cutting these prefixes in lemmatization is, that the meaning is not always strictly opposite. Similarly problematic is the prefix *dis* which can mean either strict opposite (*disloyal*) or some kind of reverse action (*disconnected*). Moreover, there are words with different meanings, but the probabilistic distribution of the meanings is not the same as for the negated word (*uneasy* is more likely to mean *uncomfortable* than *difficult*). Sometimes the non-negated form is not used (*indiscriminately*).

Foreign words

Unlike `morpha`, our Lemmatizer does not have any heuristic rules for declensions of words with Latin origin. Instead, there is an exception list `nouns_latin` with some widely known English words with Latin origin. Therefore, our Lemmatizer does not produce errors presented in Table 5.3 any more. Although we may miss some less frequent Latin words (not included in the exceptions), this should not

[12] It should be noted that the preference of a regular vs. irregular form for a given verb is dependent on the dialect of English.

[13] In the original implementation, an external spellchecker (`ispell`) was used to decide whether the word formed by deleting an assumed negative prefix is correct. It slowed down the process but the list of exception could be smaller and contain only words like *inflammable, dismissal, union* etc. Indeed, no space was saved, because the external dictionary was much bigger than our list of stems with a negative prefix. What is worse, there were apparently some problems with capitalization and hyphen words (so *Unix-based* was lemmatized as ix-based).

[14] There are some fallback routines in TectoMT that can recognize prefixes like *anti-* in the transfer phase.

be considered a major disadvantage, because those words are usually specialized technical terms, where it is questionable what the lemma should be. For example, biological terms like *Spheniscidae* – for the purposes of MT these terms are better to be left unchanged by lemmatization, because they have usually the same form in the target language.

Lowercase lemmas

The purpose of a lemma is to serve as a common label for a set of related word forms. What string of characters is chosen for the label is a matter of convention – usually it is some kind of base form, by linguistic tradition: singular (nominative) for nouns, infinitive of verbs etc. Finally, lemmas are lowercase by default.

The only problem is with proper nouns, because their capitalization is lexically determined. We have decided to preserve capitalization of proper nouns in lemmas, so we can distinguish for example lemmas Bill (William) and bill (payment or proposed law).[15]

5.3.3 Evaluation

Speed

One of initial motivations for reimplementing lemmatization in TectoMT was slowness of the original implementation. Our new implementation is more than 70 times faster. According to `Devel::DProf`[16] tests it can lemmatize about 6 200 tokens per second on 1.6 GHz CPU.

Intrinsic evaluation

After creating the first version of files `exceptions.tsv` and `negation`, we have compared our implementation of lemmatization with the original one. We have gathered a list of 240 000 word types and tags from portions of BNC and WSJ corpora. There were 8 155 word types that obtained different lemma. Out of this 2 402 word types were errors of the old implementation, 97 were errors of the new implementation and the rest was not manually evaluated. All errors found in the new implementation were subsequently added to appropriate exception files.

In the second experiment we have converted BNC corpus from CLAWS style tags to PennTB style using mapping printed in (Manning and Schütze, 1999, pp. 141–142). In order to compare our lemmatization with the one included in BNC we have done the following steps:

- All multi-word tokens were deleted (phrases like *according to* or *in terms of* are considered one token in BNC).

[15] In PDT style, lemmas can have numbers for distinguishing homonyms and technical suffixes for comments and other information. For example, Bill_;Y (Y stands for first name), bill-1_-^payment and bill-2_^proposed_law. However, this convention was not applied for English lemmas in TectoMT, because many existing tools assume that lemmas are just base forms without any technical suffixes.

[16] `http://search.cpan.org/perldoc?Devel::DProf`

- All pronouns, adjectives and adverbs were deleted, because comparatives and superlatives of adjectives and adverbs are not handled in the BNC lemmatization. On the other hand, our Lemmatizer does not change pronouns at the moment, but BNC does – e.g. *him* → he.[17]

- Both lemmas were converted to lowercase before comparing, because BNC uses different rules for capitalization (for example, *BBC* is lemmatized Bbc).

After these steps we have obtained test set A and measured accuracy of our lemmatization against the BNC lemmatization. The results are summarized in Table 5.5.

We have discovered that more than half of "errors" is caused by two reasons:

- BNC lemmatization does not cut negative prefixes

- BNC lemmatization changes also singular nouns (*MS-DOS* → ms-do)

Therefore, we have created a test set B from the set A by deleting all words beginning with a negative prefix (*un, in, im, non-, dis, il, ir*) and all words with tags NN and NNP that have different lemma and form.

	set A	**set B**
type accuracy	97.52%	98.94%
token accuracy	99.73%	99.88%
#types	740 421	712 896
#tokens	91 214 101	87 710 551

Table 5.5: *Lemmatization evaluation on BNC*

After analyzing the results of test set B, we have found that out of 109 thousand differing tokens, 28 thousand errors are caused by switching lemmas have and be when lemmatizing the contraction *'s*.[18] Although there are many other Lemmatizer errors, these have much less occurrences. 15 thousand errors are caused by the word *used* with CLAWS tag VM0 that was translated to PennTB tag MD, but in PennTB it is not considered a modal verb and would be normally tagged as VBD or VBN, so these errors would not occur. Many errors are caused by homonymous nouns that can be both invariant and regular. For example, *links* is lemmatized as links in BNC, whereas Lemmatizer chooses link. Some errors are caused by the tagger used in BNC – for example, *Alexei* is tagged as plural and lemmatized as Alexeus in BNC.

[17] However, this may change in future to be in line with the PDT style lemmatization.

[18] Lemmas have and be are tagged differently in CLAWS tagset (VHZ and VBZ), so the hard task is done by taggers – *It's*$_{VBZ}$ *easy now after it's*$_{VHZ}$ *been disambiguated by a tagger.* In PennTB both lemmas have the same tag (VBZ) and with no context information available, the best decision is to guess be as it is more frequent than have. And this is what our Lemmatizer does.

Extrinsic evaluation

Modification	diff (BLEU)	diff (NIST)
original lemmatization	0.0006	0.0294

Table 5.6: *Modifications of the lemmatization*
For explanation see Section 4.5.

Although the intrinsic evaluation revealed several areas where our new implementation (`EnglishMorpho::Lemmatizer`) outperforms the old one (`PEDT::Morpho-Analysis`), BLEU score difference is small. One reason for this can be seen in relatively high accuracies of both implementation – the differences are mainly concerning less frequent words.

Lemmatizer was implemented already before TectoMT participated in WMT09 Shared Task, so we can use results of the error analysis presented in Table 3.4. In 250 sentences analyzed, there was only one error out of 1463 that was caused by lemmatization – word *unsellable* was not included in the file `negation`, so the negative prefix was not cut and the word was left untranslated. All errors of the original `morpha` based lemmatization described in Section 5.3.1 are fixed in Lemmatizer.

5.4 Parsing

When building a-layer from m-layer, the first step to be done is parsing. A-layer uses dependency trees, but traditional English parsers create constituency trees. The first translation scenarios implemented in TectoMT have been using a constituency parser (Collins, 1999) with Ken William's Perl interface `Lingua::CollinsParser`[19]. In this way a so called p-layer (phrase-tree layer) was created and afterwards converted to dependency a-layer.

This side-step is no more needed since English dependency parsers are available. Newer translation scenarios use Ryan McDonald's Maximum Spanning Tree Parser (McDonald et al., 2005) and achieve better results than the old scenarios with Collins' Parser.

McDonald's parser is trained on CoNLL[20] data and assigns a `deprel` (dependency relation) label to every edge (technically it is stored in the attribute `conll_deprell` of the dependent node), but a-layer uses analytical functions (attribute `afun`) to label edges. The task of assignment of analytical functions will be described in Section 5.5.

5.4.1 Original implementation

The parser itself was wrapped in the block `SEnglishM_to_SEnglishA::McD_parser`[21] by Václav Novák. The feature set of McDonald's parser was modified by adding some feature templates to improve the parsing accuracy and by pruning features with lowest weights to lower the model size and speed up the parsing (Novák and Žabokrtský, 2007). After processing this block, some modifications of the a-tree were done in the block `SEnglishM_to_SEnglishA::Fix_McD_Tree`:

- Attribute `is_member` was set to 1 for nodes that had `conll_deprell = COORD`.
- Attribute `afun` was set to `Coord` for nodes with some "`is_member` children"
- Infinitive marker *to* was "switched" to govern the verb.
- Auxiliary verbs *be, have, will* were switched to depend on the main verb.

The first two rules handle coordinations. The last modification was motivated by the fact, that also in Czech a-trees auxiliary verbs depend on the main verb.

The motivation of rehanging infinitive markers (word *to* not serving as a preposition) is questionable. There are no annotation guidelines for English a-layer yet and there is no real analogy in Czech. By convention, preposition are hanged above nouns and auxiliary verbs under main verbs, but there is no infinitive marker in Czech.

[19] `http://search.cpan.org/perldoc?Lingua::CollinsParser`

[20] `http://ifarm.nl/signll/conll/`

[21] Initially, the block had been named `SEnglishM_to_SEnglishA::McD_parser_local` but it was renamed.

5.4.2 New implementation

Modularity

In the spirit of TectoMT goals, we tried to design blocks in a modular way, so they can be reused in various scenarios. The original block `Fix_McD_Tree` has served three different purposes: to fill `is_member` attributes, to fill some `afun` attributes and to modify the topology of a-trees. We have isolated these task into separate blocks: `Fill_is_member_from_deprel`, `Fill_afun_AuxCP_Coord`[22] and `Fix_McD_topology`. Moreover, we have improved the last two blocks and added some new blocks, which will be now described. Substantial changes were also made in the block `McD_parser` to support re-parsing of sentences with repaired PoS tags (see page 40) and to improve parsing of sentences with parentheses (see page 40).

Block `Fix_McD_topology`

As we can see in Table 3.4, parsing is the second biggest source of errors in our translation scenarios. After thorough inspection of these errors, we have discovered that some of them are not caused by the parser itself, but by subsequent modifications made in `Fix_McD_Tree`. For example, the rule inteded to switch auxiliary verb *be* to depend on the main verb in sentences like *What are you doing?* or *It was done.*, was applied also in sentences like *According to him, it is bad.* or *Given that fact, it is bad.* This has resulted in an incorrect (flat) structure of the a-tree and serious translation errors.

Therefore, we have refined these rules to apply only when appropriate. We have also added a rule for recognition of auxiliary verb *do*, so it is hanged under the main verb in sentences like *It did not help.*

We have decided to remove the original rule that switched infinitive markers to govern verbs. This was motivated pragmatically – when building the t-layer from a-layer, auxiliary nodes (prepositions, subordinating conjunctions, auxiliary verbs, infinitive marker *to* etc.) are collapsed to appropriate autosemantic nodes. When infinitive markers are under verbs, they can be handled in the same way as auxiliary verbs and the implementation is more lucid. However, we are aware of the fact that this particular question is polemic – both conventions have their pros and cons.

Block `Fix_is_member`

Coordination heads with only one coordination member (i.e. only one child with the attribute `is_member` set to 1) are suspicious. There may be rare cases when the one member of coordination is elided or pronounced in another sentence – *And I love her.* However, in most cases it is one of the following parser errors:

- There is another child of the coordination head that is actually also a member of the coordination, but it did not get `conll_deprel = COORD`, so it was not recognized as `is_member`.
- There should be another member of the coordination, but it was attached below a wrong parent.

[22] Aside from filling analytical functions for coordination conjunctions (Coord), `Fill_afun_AuxCP_Coord` fills also analytical functions for subordinating conjunctions (AuxC) and prepositions (AuxP), see page 45.

- There is actually no coordination; there should be no `conll_deprel = COORD` (*everybody but me.*[23]).

Block `Fix_coordinations`

According to our survey (see Section 3.3.1), parsing errors associated with coordinations are frequent and represent 39% of all parsing errors. However, the block `Fix_is_member` changes only the `is_member` attribute, so it can fix only small amount of these errors.

The McDonald's parser uses a set of features and weights to score all possible edges between all nodes and then it chooses the best combination that forms a tree – the maximum spanning tree. With this algorithm (McDonald et al., 2005) it is not possible to model (probabilistic) dependencies between sibling nodes – score of an edge is the same regardless of whether another edge with the same upper node is selected for the tree.[24]

As a result, the parser often creates coordinations whose members are not compatible, e.g. one member is a verb and the other noun.

We have implemented rules that can detect such incompatibilities with high precision – almost all detected cases were real errors. Unfortunately, we have failed to specify rules that could fix those errors. Either we got very low recall (the rule was too specific), or very low precision (the rule made parsing even worse than it was). Therefore, we have not included this block to the final translation scenario.

Block `Fix_atree`

This block includes several rules for fixing parser errors (not associated with coordinations) that were frequently found in the data, for example:

- Terminal punctuation is hanged on the technical root to be in line with PDT style.
- Phrases in form *much more/less* `RB/JJ` are parsed incorrectly by McDonald's parser.
- Rhematizer and similar words that should not have children (*only, just, too, almost*), but actually have some, are switched with the first one.
- WH-pronouns should not have children. If there are some, all are hanged to the parent of the WH-pronoun.

[23]The part of speech of *but* in its various senses is a notorious problem. According to PennTB Guidelines, it is a preposition (`IN`) when meaning *except* and adverb (`RB`) when meaning *only* – *We can but try.* There are other theories; some argue it is always a conjunction, some distinguish whether it is at the and of a clause or not. We adhere to PennTB style, because it helps us to distinguish the two (or more) meanings and translate it correctly. *None but the brave deserves the fair.*

[24]This limitation of the original McDonald's parser can be partly overcome by using second-order maximum spanning tree parser (McDonald and Pereira, 2006), where the score is computed for adjacent edge pairs that are "on the same side of parent". For example, the score of coordination *dogs #comma cats and rats* (with correct parsing) is computed as S(*and, dogs, #comma*) + S(*and, #comma, cats*) + S(*and, cats, –*) + S(*and, –, rats*), where S(x, y, z) is the score of creating a pair of adjacent edges from word x to words y and z. Nevertheless, it is still not possible to model the probability of *cats* and *rats* being siblings.

- In phrases like *go from here to there* or *sleep from 9 p.m. until 9 a.m.*, words *to* and *until* should not be children of *from*, but siblings.
- Numerals should not be siblings unless coordinated (*over ten thousand* – here *ten* should be a child of *thousand*)

There are also two rules that change both the parsing and some PoS tags. The rules concern:

- Words *about, around* and *nearly* when used to mean *approximately*
- Article *a* serving as a preposition *per* or *for* (*eight days a week*, *$5 a day*)

Block `Fix_multiword_prep_and_conj`

This block was already present in TectoMT, but it was used only in the old scenarios with Collins' parser. Its goal is to detect multiword prepositions (*because of, in accordance with, up to, on to,...*) and conjunctions (*as well as, even though, provided that,...*) and normalize the way how they are parsed – the first token becomes the head and the other become its children.

We have added this block to the new translation scenario and improved it slightly by adding some more prepositions and conjunctions to the respective lists. We have also added a rule to skip cases when phrasal verb particle (`RP`) may be confused with a preposition (*heat up to toxic levels* or *He moved on to do his own work.*).

Re-parsing

The block `SEnglishM_to_SEnglishA::Fix_tags_after_parse` (described in 5.2.2) fixes some tags using the knowledge of a parse. It is possible that the wrong tag caused also some parsing errors.[25]

To resolve this issue, we decided to run the parser once again. Since parsing is one of the most time-demanding blocks, we re-parse only the sentences influenced by the block `Fix_tags_after_parse`. Even for those sentences it is not necessary to build new a-nodes and fill all the attributes; it suffices to change the topology and edge labels (`conll_deprel`) of old nodes.

When writing scenarios, we must specify by block parameters a model to be used by McDonald's parser and whether it should run in re-parse mode. For illustration see Figure 5.2.

The block for filling `is_member` attributes must be also run for the second time, because it is a prerequisite for `Fix_tags_after_parse`, but after the re-parsing, some `is_member` attributes might have changed. The block is very fast, so it can be applied on all sentences without any noticeable slowdown.

Parsing sentences with parentheses

First of all, one terminological note: We will use term "bracket" for any type of brackets: round, square and curly (there are, however, almost no square and curly

[25] We must be careful, because we are near a vicious circle. The error in tag was discovered based on the parsing, so either it is a reliable parsing that does not contain many errors, or we cannot be sure that the block `Fix_tags_after_parse` works well.

```
SEnglishM_to_SEnglishA::McD_parser
  TMT_PARAM_MCD_EN_MODEL=conll_mcd_order2_0.01.model
SEnglishM_to_SEnglishA::Fill_is_member_from_deprel
SEnglishM_to_SEnglishA::Fix_tags_after_parse
SEnglishM_to_SEnglishA::McD_parser
  TMT_PARAM_MCD_EN_MODEL=conll_mcd_order2_0.01.model REPARSE=1
SEnglishM_to_SEnglishA::Fill_is_member_from_deprel
SEnglishM_to_SEnglishA::Fix_McD_topology
SEnglishM_to_SEnglishA::Fix_is_member
SEnglishM_to_SEnglishA::Fix_atree
SEnglishM_to_SEnglishA::Fix_multiword_prep_and_conj
SEnglishM_to_SEnglishA::Fill_afun_AuxCP_Coord
SEnglishM_to_SEnglishA::Fill_afun
```

Figure 5.2: *Current scenario for building English a-layer from m-layer*

brackets in the data we use for training and evaluation). We will use term "parenthesis" for a sequence of words that does not belong to the basic level of the text and is separated from it by brackets.[26]

We have noticed that translation of parentheses was a surprisingly big problem in the original TectoMT. Although 1.4% of serious errors and 2.9% of minor errors (see Table 3.6) can be considered a small number compared to errors caused by wrong lexical choice (38% of serious errors), we must reflect the fact that out of 250 analyzed sentences there were only 22 sentences with parentheses. Moreover, only two of those 22 sentences contained no error of type `punct-brack` (missing, superfluous or displaced brackets).

The most striking error was wrong word order of brackets, e.g. instead of surrounding one word they were surrounding whole phrase or clause. Sometimes they were completely missing in the output and conversely, sometimes there were superfluous brackets. The reason for this strange behaviour is that brackets are not kept as self-standing nodes on the t-layer; instead there is an attribute `is_parenthesis`.[27] For correct translation it is essential that both left and right bracket is parsed correctly (i.e. they have the same parent).

After more detailed inspection of a-trees made by McDonald's parser, we have found that the problem with parentheses is even more complicated:

- Sometimes only one of the brackets is parsed incorrectly and everything else is as it should be.
- Sometimes the parenthesis is incorrectly divided and each part hanged to another parent.

[26]Parenthesis can be separated also with dashes or commas, but these cases are not so common – moreover, there is no clear distinction from coordination and apposition, for example see this sentence. In the current implementation, we recognize only parentheses that are separated with round brackets.

[27]In PDT, the attribute `is_parenthesis` is set to 1 for all nodes that are part of a parenthesis. However, in TectoMT it is set only for the root of the subtree. This is almost equivalent solution except for rare sentences with so-called discontinuous parenthesis like *Peter (and Paul) came.* For more information see T-layer manual for PDT (Mikulová et al., 2006).

- Sometimes there are parse errors also in the rest of the sentence, but these errors disappear, when we try to parse the sentence without the parenthesis. We have not done any further research involving changes in McDonald's parser internals, so we can only guess what is the reason. Maybe it is just an easier task with fewer distracting words, but there may be also some features that give better scores to "shorter" edges.

To prevent all described problems, we have enhanced the `McD_parser` block. Every sentence is segmented into chunks according to matching brackets. If there are any irregularities like non-matching brackets or nested brackets, we rather treat the whole sentence as one chunk. Each chunk is then parsed separately. For example in the sentence *Up to 700 billion dollars (nearly 12 billion Czech crowns) was to be invested.* we call the parser twice: first with *Up to 700 billion dollars was to be invested.* and then with *nearly 12 billion Czech crowns.*

This way it is ensured that every parenthesis will be parsed in its own subtree with left and right bracket depending on the root node. In the current implementation, we always hang the parenthesis subtree to the immediately preceding word. This is not always the correct solution, but even if it is wrong, it does not affect negatively the translation quality, since the dependency of parenthesis on the rest is usually "weak".

5.4.3 Evaluation

Modification	diff (BLEU)	diff (NIST)
`Fix_McD_Tree` instead of `Fix_McD_topology`	0.0016	0.0661
no `Fix_is_member`	0.0001	0.0017
no `Fix_atree`	0.0015	0.1780
no `Fix_multiword_prep_and_conj`	0.0006	0.0276
no re-parsing	0.0001	0.0004
old `McD_parser` (no parentheses handling)	0.0031	0.0317
all above together = original parsing	0.0072	0.3006

Table 5.7: *Modifications of the parsing*
For explanation see Section 4.5.

We have achieved a significant improvement in parsing without actually changing any internals of McDonald's parser. According to BLEU scores in Table 5.7, the most helpful improvement was the concept of parsing parentheses separately from the rest of a sentence. Surprisingly, this is not in accordance with NIST scores, where the most helpful improvement is the block `Fix_atree`.

Re-parsing did not help, because there were only few sentences to be re-parsed. This is in consonance with the results in Table 5.2, where we can see that also the block `Fix_tags_after_parse` does not help. However, we think that if there were more changes of tags after parsing, re-parsing would become useful.

Coordinations remain the biggest source of parsing error – we have not succeeded to find any rule-based solution. After this experience, we will consider applying either some machine learning techniques for parsing postprocessing or a combination with another parser.

5.5 Assignment of analytical functions

Analytical function (afun for short) is the key attribute of the a-layer. It specifies the type of dependency relation of a node to its governing node. Although the list of analytical functions is language dependent, most analytical functions are quite universal and can be used in different languages. For example, subject (Sb), object (Obj), predicate (Pred), adverbial (Adv), attribute (Atr), preposition (AuxP), subordinating conjunction (AuxC) or coordination head (Coord).[28]

Detailed guidelines for annotation of the a-layer and analytical functions are available only for Czech (Hajičová et al., 1999). There is no such source for English yet, so we had to decide several questions before implementing automatic assignment of analytic functions. We tried to be in line with the Czech system implemented in PDT as much as possible, but there are some English phenomena without any Czech equivalent.

- **Articles**
  We have established a new analytic function AuxA for articles (in English definite *the* and indefinite *a, an*). There used to be an analytic function named Det in TectoMT, but we do not use it, because other determiners (*this, each, any, some, every,...*) have Czech equivalents and these are considered attributes (Atr). Moreover, it is a convention that names of analytic functions for auxiliary nodes (i.e. those that are not present on the t-layer) start with Aux.

- **Negation**
  We have established a new analytic function Neg for the word *no*. In PDT-style t-layer, this word should be represented by a t-node with t-lemma #Neg, so the name of the analytic function should not start with Aux. In TectoMT-style t-layer, the negation is represented by a special grammateme, but this should not affect the name of the analytic function.

- **Auxiliary verbs**
  In Czech, there is only one auxiliary verb *být* (and all its forms) – *to be*. Just this verb gets analytic function AuxV (but only when used as auxiliary). In English, there are also other auxiliary verbs (apart from *be* and *will*): *do* (*What do you want? It did not help.*) and *have* (*Have you been there? He had written.*). We have decided to label these auxiliaries also with AuxV.

  Although modal verbs (*must, may, can, ought,...*) or so-called quasi-auxiliaries (*used to, be going to,...*) are considered auxiliary verbs in some grammars, these are not marked as AuxV, according to the same convention in Czech.[29]

- **Phrasal verb particles**
  Words with PoS tag RP are part of phrasal verbs, e.g. *put off*$_{RP}$*, make up*$_{RP}$.

[28] Other universal analytic functions include technical root of the tree (AuxS), comma not serving as a coordinating conjunction (AuxX), terminal punctuation of a sentence (AuxK) and other graphic symbols (AuxG). On the other hand, analytic functions like reflexive tantum (AuxT) or passive reflexive (AuxR) are specific for Czech (and other Slavic languages) and remain unused in English.

[29] In the Czech a-layer, modal verbs govern main verbs, so the analytic function of the modal verb in a main clause is Pred.

On the t-layer, phrasal verbs are represented by one node with a "compound t-lemma", e.g. put_off, make_up. We have decided to label these particles also with the analytic function AuxV.

- **Infinitive marker *to***
  The particle *to* is not a verb, so it cannot be called an auxiliary verb. On the other hand, it shares some properties with auxiliary verbs; namely, it is a part of compound verbs, e.g. *want to go, have to go.* We have decided to label the infinitive marker with the analytic function AuxV. For a discussion about how to parse infinitive markers see page 38.

5.5.1 Original implementation

In Czech analysis scenarios, analytic functions are heavily used, especially when building the t-layer. For example, it is very helpful to be able to distinguish between subjects and objects of verbs. Since there are manually annotated corpora with analytic functions available for Czech (PDT 2.0 (Hajič et al., 2006), CAC 2.0 (Hladká et al., 2008)), there are also statistical parsers that can assign analytic functions, e.g. the Czech version of Morce (Spoustová et al., 2007).

Unfortunately, there is no such parser for English yet, so analytic functions must be assigned in a separate block. In TectoMT, there was a block `Fill_afun_after_McD` that applied several simple rules to guess some `afun` values based on PoS tags and `conll_deprel` attributes. However, this block was never used in translation scenarios. Instead, the blocks for building t-layer from a-layer were using heuristic rules to find out the information that would be otherwise contained in analytic functions.

5.5.2 New implementation

Our motivation for introducing analytic functions to the English analysis was the following (ordered by its importance):

- Translation errors caused by tectogrammatical analysis were for us hard to repair, because the original implementation was very complicated. After adding a block for analytic function assignment, we were able to rewrite the tectogrammatical analysis in a more straightforward way (see Section 5.6).
- Our block for analytic function assignment may be substituted by a better one in future, which should also improve the accuracy of the new tectogrammatical analysis based on analytic functions. That is the purpose of a modular design.
- The block for analytic function assignment could be reused also in other applications than translation. (It was recently used in a scenario for preprocessing of Prague English Dependency Treebank.[30])
- We have noticed some code duplication caused by the absence of analytic functions.
- Without analytic functions, it would not be accurate to call our trees "analytical" and use the term a-layer.

[30] `http://ufal.mff.cuni.cz/pedt/`

Should we use conll_deprel?

Our first attempt at introducing analytic functions into the translation was to improve the block `Fill_afun_after_McD`. It exploits `conll_deprel` attributes which are assigned to nodes by the McDonald's parser. The problem with this method is that it precludes future switching to another parser that does not assign `conll_deprel` attributes. Analytical trees without analytic functions and `conll_deprel` attributes can also be obtained from constituency trees – either automatically parsed (e.g. by Collins' parser) or manually parsed (e.g. in PennTB). We consider the inability to analyze such trees up to t-layer a serious disadvantage. Therefore, we have decided to create a new block that would not depend on `conll_deprel` attributes.

Block `Fill_afun_AuxCP_Coord`

Soon after we started programming the new block for assignment of analytic functions, we have noticed that it is useful to divide the task into two parts. The block `Fill_afun_AuxCP_Coord` fills only analytic functions for coordination conjunctions (`Coord`), subordinating conjunctions (`AuxC`) and prepositions (`AuxP`). After applying this block, it is possible to use standard TectoMT methods for finding effective children (see Section 2.1) and effective parents. These methods are used in the block `Fill_afun`, which fills the rest of analytic functions.

Coordinations are detected according to the attribute `is_member`.[31] Subordinating conjunctions and prepositions are detected according to PoS tags. In general, both have tag `IN`, but prepositions should have a noun child, whereas subordinating conjunction a verb child. There are, however, some special cases:

- Word *to* is tagged always as `TO` in PennTB, regardless of whether it is actually a preposition or an infinitive marker.
- When talking about a noun child of a preposition, we must consider not only PoS tags starting with `NN`, but also `PRP|CD|WP|WDT|DT`.
- The second word of a multiword preposition or conjunction has no child. Similarly for a possible third (fourth,...) word.
- Postposition *ago* should also have analytic function `AuxP`, but it is usually tagged as `RB` in PennTB.
- The assignment of analytical functions should be robust and choose the best value even if the parsing is not fully correct.

Block `Fill_afun`

In our early experiments, we had some problems with distinguishing subjects and objects of a verb. The problems were varying: reversed word order in sentences with direct speech, relative clauses, copula constructions, modal verbs etc. Therefore, we have decided to traverse each a-tree twice.

[31] The attribute `is_member` is set to 1 for members of a coordination or apposition. Since appositions are not detected yet in TectoMT, every node with some "`is_member` children" is said to be a coordination head. There is a slight difference between PDT and TectoMT style in treating `is_member` attributes of preposition, but this goes behind the scope of this thesis.

In the first traversal, subjects are identified for each finite verb. Except from coordinations, each finite verb can have just one subject, so we choose the "best" child according to its tag and word order.

In the second traversal, we assign an analytic function to every node (except for nodes that have analytic function already filled). We use about twenty rules and if no one matches, we fill the attribute `afun` with value `NR` (not recognized).

5.5.3 Evaluation

Although there are no manually annotated data for a real intrinsic evaluation, we could benefit from the existence of the older implementation – `Fill_afun_after_McD`. We have periodically compared the results of both blocks during the development to improve our implementation. More than 95% of the differing instances are now guessed correctly by the new implementation (`Fill_afun`). However, this could not be considered an evaluation, because there are many cases when both blocks give the same wrong result – and these were not checked in the manual comparison.

Modification	diff (BLEU)	diff (NIST)
`Fill_afun_after_McD` instead of `Fill_afun`	*0.0240	*0.6676
no analytic functions & old tectogrammatical analysis	0.0056	0.1084

Table 5.8: *Modifications of the assignment of analytical functions*
For explanation see Section 4.5.

In the extrinsic evaluation presented in Table 5.8, it was not possible to precisely evaluate the improvements achieved by our new analytic function assignment block on its own. Our new tectogrammatical analysis (described in the following Section) needs analytic functions; if we delete the block `Fill_afun` from the scenario, we must also substitute the new tecto-analysis with the old one. This results in a remarkable difference in BLEU scores (0.0056). For examples of changed sentences and a discussion see Section 5.6.4.

We have also carried out an experiment, where we substituted the block `Fill_afun` with the older implementation – `Fill_afun_after_McD`. Results of this experiment are marked with an asterisk, because they cannot be interpreted as if the block `Fill_afun` had improved the translation quality by more than 0.02 BLEU. It just shows that the older implementation (`Fill_afun_after_McD`) fails to fill all analytic functions needed by the new tecto-analysis.

5.6 From a-layer to t-layer

Building the t-layer in TectoMT can be divided into three phases:

1. **Mark auxiliary nodes and edges to be collapsed**
   Auxiliary nodes are marked in a-trees. These nodes will not be present on the t-layer as self-standing t-nodes. Every auxiliary node will be saved in the attribute `a/aux.rf` of some non-auxiliary (i.e. autosemantic) node, so we can say that these a-nodes will collapse into one t-node. Some types of auxiliary nodes (prepositions, subordinating conjunction, modal verbs) govern "their" autosemantic nodes, whereas other types (articles, auxiliary verbs,...) depend on their autosemantic nodes.[32] Hence apart from marking auxiliary nodes, we must also mark whether to collapse them along the edge going up to the parent or along the edge going down to some child.

2. **Build t-layer structure**
   The structure of the t-layer is built using an algorithm that collapses edges marked in the previous phase. At the same time, attributes `a/aux.rf` and `a/lex.rf` are filled.

3. **Fill t-layer attributes**
   All other needed attributes are filled. Some of these attributes are a standard part of the t-layer also in PDT – `nodetype`, `functor`, grammatemes, `is_member` etc. Some are added in TectoMT for various purposes – `formeme`, `named_entity`, `is_clause_head` etc.

Before we describe modifications we have done to improve these phases, we should point out that the t-layer used in TectoMT for translation purposes differs in some aspect from the t-layer used in PEDT (Hajič et al., 2009). The most noticeable differences are:

- No generated nodes are added.
- Formemes are assigned to t-nodes.
- Negation is represented by the grammateme negation also for verbs.[33]

[32] The situation is actually somewhat more complicated. For example, prepositions have normally just one child (typically a noun), so they are classified as "aux-to-child". However, there are multi-word prepositions (e.g. *up to(parent=up) a point(parent=up)*), where the second word (preposition *to*) has no child. We call this case "aux-to-parent" even if the autosemantic node is not a parent, but a sibling.

[33] Verbal negation is represented by a t-node with lemma #Neg in the PDT style, which is used also in PEDT. There is no information structure annotated in PEDT, so the scope of negation cannot be determined from the attribute `deepord`. The PEDT-style representation of negation is equivalent to the TectoMT-style. Both representations cause the same problems with modal verbs: On the t-layer, most modal verbs are not represented by separate t-nodes; instead, the modality is expressed by the value of the deontic modality grammateme (deontmod). However, there must be an exception for cases with a negated infinitive of the full verb, e.g. *I can't not obey her*, where both the modal verb and the main verb have their own t-node in order to be able to distinguish the two types of negation. See (Cinková et al., 2006, p. 89).

5.6.1 Mark auxiliary nodes and edges to be collapsed

Original implementation

The original block `Mark_auxiliary_nodes` uses three a-node attributes to mark auxiliary nodes and edges to be collapsed: `is_aux_to_child`, `parent_is_aux` and `is_aux_to_parent`. The first attribute is used in cases when an auxiliary node (e.g. preposition) has only one child and the edge to that child should be collapsed. This attribute is redundant, because the same effect can be achieved by marking the child with the attribute `parent_is_aux`.

The block is not designed to make use of analytic functions.[34] As a result, many exceptions must be solved and the whole block is quite complicated.

There is also a block `Mark_negator_as_aux` that marks word *not* under verbs as auxiliary. This block is needed in the translation scenarios, but not in preparing data for annotators of PEDT, so it must be kept separately from `Mark_auxiliary_nodes`.

New implementation

We have decided to create a completely new block named `Mark_edges_to_collapse`. Instead of the three before-mentioned attributes, we use only two attributes: `is_auxiliary` and `edge_to_collapse`.[35] The latter is set to 1 for nodes that should be "joined" with their parents.

Apart from this rather technical modification, our rules exploit analytic functions and deal with special cases that were not solved properly in the original implementation. We have aimed at a robust implementation that can handle also some cases with inaccurate parsing, e.g. preposition under noun. To make sure that our new implementation does not make errors where the original one was correct, we collected 300 sentences and manually checked all differences of resulting t-trees.

Block `Mark_edges_to_collapse_neg` is an equivalent to `Mark_negator_as_aux` except it uses the new-style attributes.

5.6.2 Build t-layer structure

Block `Build_ttree`

The original block `SEnglishA_to_SEnglishT::Build_ttree` contains many rules for special cases including some rules specific to English.

We have implemented our new block `SxxA_to_SxxT::Build_ttree` in a language independent way, so it can be used also for other languages.

The algorithm is quite straightforward – it creates new t-trees by a recursive traversal through a-trees and collapsing marked edges. References to auxiliary a-nodes are saved in the attribute `a/aux.rf` and non-auxiliary (also called lexical) in `a/lex.rf`. Every t-node must have exactly one lexical a-node counterpart and zero or more auxiliary a-nodes. This requirement can be broken in two cases:

[34] There are rules involving p-layer attributes, as an ersatz for analytic functions. Since p-layer is not present in the new translation scenarios, some of these rules were patched to work also without p-layer attributes.

[35] These attributes were introduced by David Mareček in his work on converting SynTagRus Treebank (Boguslavsky et al., 2000) into tectogrammatical trees.

- There is no `a/lex.rf`
  In our implementation of `Mark_edges_to_collapse` this case cannot occur, because with every edge to be collapsed we mark just one node as auxiliary (either the dependent or the governing one) and no other auxiliary nodes are marked.

- There are more than one `a/lex.rf` for one t-node
  This case may rarely happen when the sentence was not parsed correctly. The attribute `a/lex.rf` can bear only one reference to a lexical a-node, so the situation must be solved within the block `Build_ttree`. Our first implementation simply removed the original lexical a-node from `a/lex.rf` to `a/aux.rf` and reported a warning. Afterwards, we have discovered that better results are achieved by creating a new t-node as a sibling of the original t-node.

Block `Move_aux_from_coord_to_members`

Prepositions and subordinating conjunctions should not be saved in the `aux.rf` attribute of a head of coordination. Instead, they should be saved in the `aux.rf` attribute of members of the coordination.

For example, let's consider the phrase *in Prague and London* in Figure 5.3 – the preposition *in* is moved from the coordination head (*and*) to the members (*Prague* and *London*) as if the phrase was *in Prague and in London*.

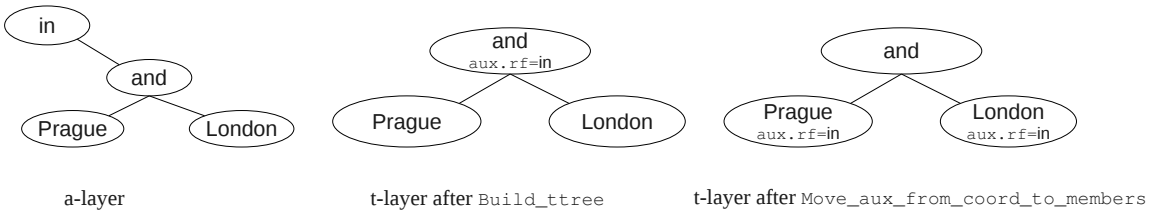


Figure 5.3: *Moving auxiliary nodes from a coordination head to members*

The original block for this task[36] suffered several errors. For example, it moved also punctuation symbols including brackets, which gave rise to superfluous brackets surrounding every coordination member.

[36]The original block was actually named `Distrib_coord_aux`, but for simplicity we will call it "old `Move_aux_from_coord_to_members`" in the table with results (Table 5.10).

5.6.3 Fill t-layer attributes

Block `Assign_grammatemes`

We have rewritten this block and repaired several errors. We have also added rules for recognizing interrogative and imperative sentences (grammateme sentmod), modal verbs *have to, ought to, be able to* (grammateme deontmod), and correct number for words like *hundred, thousand,...* (grammateme number)[37].

Block `Detect_formeme`

We have rewritten this block and removed a code for detecting semantic part of speech (attribute `sempos`). The attribute `sempos` must be detected already in the block `Assign_grammatemes`, so it is unnecessary (and inappropriate) to duplicate that code. In some rules, we have substituted the usage of a topological parent by an effective parent. Substantial improvements were achieved by introducing analytic functions to distinguish subject and object (formemes n:subj and n:obj). We have established a new formeme value n:adv for semantic nouns used as adverbials (e.g. *This year(*`formeme=n:adv`*), it was ...*).

Blocks `Mark_clause_head` **and** `Mark_relclause_head`

Heads of finite verb clauses are marked by setting the attribute `is_clause_head` to the value 1. Similarly, heads of relative clauses are marked with `is_relclause_head` (in addition to `is_clause_head`). We have rewritten both the blocks and refined some rules.

For example, in the original implementation, there was a rule for marking relative clause heads, that marked just those clause heads that had some wh-pronoun (PoS tag starts with `W`) as a child. However, there are also constructions like *licenses, the validity of which will expire*, where the wh-pronoun (*which*) is not a child, but a grandchild of the verb (*expire*). In our implementation also these constructions are marked as relative clauses.

5.6.4 Evaluation

Transformation based tecto-analysis

In TectoMT, there are blocks for transformation-based tectogrammatical analysis (TBLa2t) of English (Klimeš, 2007). These blocks are intended to build the PDT-style (or PEDT-style) t-layer with generated nodes and attributes like functors and valency frames. In an intrinsic evaluation presented in Klimeš (2007), TBLa2t outperformed the original TectoMT rule-based implementation in almost all subtasks of tecto-analysis, most significantly in the assignment of functors and valency frames. Although these attributes are not used in TectoMT translation scenarios, TBLa2t had slightly better results also in determining the structure of t-trees, which could help to improve the translation quality.

[37] There are many nouns that have the same form for singular and plural (*fish, aircraft, blues, athletics,...*), but they are disambiguated by a tagger (`NN` or `NNS`). However, there is only one tag for numerals (`CD`).

We have added TBLa2t blocks to our translation scenario to check this hypothesis. English version of TBLa2t needs in its first block (`TBLa2t_phase0`) constituency trees to construct features, but we do not have any p-layer in our new scenarios. Therefore, we have skipped the first block, initialized the t-layer using TectoMT rule-based blocks (our best version) and afterwards applied blocks `TBLa2t_phase1` – `TBLa2t_phase4` to improve the structure.

	BLEU
our scenario	0.1013
with TBLa2t	0.0920

Table 5.9: *Results of adding TBLa2t tecto-analysis to our translation scenario*

We should be careful when interpreting the results in Table 5.9. It simply shows that our experiment with utilization of TBLa2t was not successful, but it does not mean that TBLa2t cannot help in future after some adjustments.[38]

Our modifications

	Modification	diff (BLEU)	diff (NIST)
a)	old construction of t-layer structure	0.0039	0.0804
b)	old `Move_aux_from_coord_to_members`	0.0036	0.0360
	a – b	**0.0040**	**0.0753**
c)	old `Mark_clause_head`	0.0011	0.0247
d)	old `Mark_relclause_head`	0.0000	0.0005
e)	old `Assign_grammatemes`	0.0010	0.0135
f)	old `Detect_formeme`	0.0010	0.0309
	c – f	**0.0028**	**0.0686**
	a – f = original tecto-analysis	0.0053	0.1024

Table 5.10: *Modifications of the tecto-analysis*
For explanation see Section 4.5.

In Table 5.10 we can see that we have achieved a substantial improvement in the phase of marking auxiliary nodes and edges to be collapsed and the phase of building t-layer structure (BLEU difference 0.0040 for the experiment a–b) as well as in the phase of filling t-layer attributes (BLEU difference 0.0028 for the experiment c–f).

[38]There are at least three areas that should be adjusted for successful cooperation of TBLa2t and TectoMT translation. First, the a-layer should be arranged to a form that is acceptable as the input for TBLa2t. It seems that TBLa2t uses the attribute `afun`, but assumes some ad hoc values different from PDT-style analytic functions. There are also some technical problems with the fnTBL tool, which crashed on some inputs, so we were not able to translate our whole test data, but only its subset – 850 sentences. Second, TBLa2t was trained on manually annotated data from PennTB converted to dependency a-trees, which may be different than a-trees created by McDonald's parser, so probably a re-training would be needed. Third, also the transfer phase should be adjusted to cope with the t-layer produced by TBLa2t. Our transfer does not need functors and valency frames, but their presence should not be a problem. However, some problems could be caused by generated t-nodes, not using the grammateme for negation for verbs etc.

Errors in coordination lead to great differences in BLEU, especially when there are superfluous tokens (brackets) between the members of the coordination or missing commas. The reason is that even if the members are translated correctly, they are counted only as matching unigrams instead of matching bigrams, trigrams etc.

We should note that punctuation errors are usually treated as less serious by human judges compared to the BLEU score. This shortcoming of the BLEU score is partly overcome in the NIST score (Doddington, 2002), which uses the so-called information weight to score frequent tokens (or n-grams in general) with lower weight. This may be the cause, why the NIST differences in the construction of t-layer structure are much greater then in `Move_aux_from_coord_to_members`, whereas the BLEU differences are almost the same.

Similar discrepancy between BLEU scores and NIST scores can be seen also in the phase of filling attributes. According to the NIST score, most helpful were the improvements of `Detect_formeme`. Among those improvements, most helpful was the correct recognition of subjects and objects in sentences with reversed word order, like the direct speech in the following example.

SRC:	*"The chaos and disruption on the daily lives of Americans will be immense," declared Republican Judd Gregg.*
REF:	*„Trauma, chaos a narušení každodenního života Američanů budou obrovské," prohlásil republikán Judd Gregg.*
TST–OLD:	*„Trauma, chaos a porucha na denních životech Američanů budou obrovské," prohlásil**i** republikán**a** Judd**a** Gregg**a**.*
TST–NEW:	*„Trauma, chaos a porucha na denních životech Američanů budou obrovské," prohlásil republikán Judd Gregg.*

The difference between the output produced with the old implementation of `Detect_formeme` (TST-OLD) and with the new one (TST-NEW) is emphasized by a bold font. Although it is just a change of inflections (indicating number and case), the meaning of TST-OLD is obscured.

Chapter 6

Transfer

Sem čelist ještě na své milé.
Poetic TectoMT, 2008[1]

6.1 Transfer strategy

The task of the transfer phase in our translation scenarios is to build Czech t-trees from English t-trees. Since this task is very complex it is divided into four subtasks:

1. clone t-tree
   The topology of target-side t-trees is copied from source-side t-trees. Also all attributes of t-nodes are copied without any change.[2] This subtask is just a technical step implemented in the block `Clone_ttree`.

2. translate lemmas and formemes
   Translation of lemmas and formemes can be viewed as two channels of transfer factorization. Although these channels are not fully independent, the factorization seems to be useful, because it helps to build robust models and fight data sparseness. Probabilistic dictionaries provide n-best list of lemmas and n-best list of formemes for every t-node. The question how to find the best combination is discussed in Section 6.2.1.

3. translate grammatemes
   Translation of grammatemes can be considered a third channel of factorization. It is much easier than the first two channels, because most grammatemes remain unchanged in transfer – singular is usually translated as singular, past tense as past tense, interrogative mood as interrogative mood etc. Therefore, the blocks for translation of grammatemes can use much less complex methods – typically, they are rule-based.

4. fix topology and word order
   There are cases when target t-trees cannot be isomorphic with the source t-trees. Some of these cases can by solved using rule-based blocks, for example, to translate *in 1990* as *v roce 1990* a new t-node for the word *roce* (*year*) must be added. There are also blocks for changing word order, for example nodes with genitive formeme are moved to postposition – *police chief* → *šéf policie*.

[1] This motto was translated to Czech by a modified version of our system, that reverses the n-best list of lemmas. The source English sentence is *I'll come a bit later on my own.*

[2] The only exception are co-reference attributes (e.g. `coref_gram.rf`) and the identifier attribute (`id`), which have to be updated, e.g. `SEnglishT-s1w2` changes to `TCzechT-s1w2`. Every target-side t-node has also the attribute `source/head.rf`, which points to the source-side t-node.

Most modifications were done in the second subtask, which is apparently the most difficult one. We have suggested a new strategy for translation of lemmas and formemes, which will be described in the rest of this Section. One of the motivations for the new strategy was to facilitate the addition of Hidden Markov Tree Model based transfer block, which will be discussed in Section 6.2 and Section 6.3.

6.1.1 Original implementation

The subtask of translation of lemmas and formemes was realized in two blocks: `Baseline_formeme_translation` and `Baseline_tlemma_translation`.[3]

Block `Baseline_formeme_translation`

For every t-node, one formeme was chosen using so-called valency formeme translation dictionary, which models the probability of target formeme given source formeme and source parent's lemma. For example, in the sentence *What do you mean?*, word *you* has formeme n:subj (semantic noun in subject position) and its parent has lemma mean. According to the valency formeme dictionary, the probabilistic distribution $P(Y|\text{n:subj}, \text{mean})$ is n:4 = 0.37, n:1 = 0.32, n:7 = 0.19, adv: = 0.07 etc. Only the first translation from this n-best list was chosen – the Czech formeme n:4 (semantic noun in accusative case).[4]

If there was no entry for the given formeme and parent's lemma in the valency dictionary, simple formeme-to-formeme dictionary was used as fallback. For example, in the phrase *question without an answer*, word *answer* has formeme n:without+X and its parent has lemma question. There is no entry for (n:without+X, question) in the valency formeme dictionary, because no such pair was present in the training corpus. According to the formeme-to-formeme dictionary, the distribution $P(Y|\text{n:without+X})$ is n:bez+2 = 0.72, v:aniž+fin = 0.06, adv: = 0.06 etc.

Both the formeme dictionaries were created by Zdeněk Žabokrtský using 10 000 sentence pairs from the parallel text distributed during the Shared Task of Workshop in Statistical Machine Translation[5] (Žabokrtský et al., 2008).

The block `Baseline_formeme_translation` contained also a few rules to fix some formeme translations from the dictionaries. For example, the Czech formeme v:fin is applicable only for main clause verbs, heads of parentheses and heads of direct speeches; in other cases it was changed to v:že+fin.

Block `Baseline_tlemma_translation`

The lemma of every t-node was translated as its most probable target-language counter-part compatible with the already chosen formeme. The compatibility was

[3] The very first implementation of blocks `Baseline_formeme_translation` and `Baseline_tlemma_translation` was quite simple and straightforward (therefore it was called "baseline"). However, over time it was extended by many modifications and both the blocks became quite complex as they served for several goals at once. Actually, these "baseline" blocks were the best what was available at the time of WMT09 Shared Task submission.

[4] This particular choice is actually incorrect. The correct formeme is n:1 (semantic noun in nominative case), because subjects are represented in Czech by nominative. Since Czech is a pro-drop language, there is a block in the synthesis phase that drops nominative personal pronouns.

[5] `http://www.statmt.org/wmt08/`

ensured by a set of rules. For example, formemes beginning with `n` (semantic noun) can be combined only with lemmas whose PoS is noun or pronoun.

The probabilistic dictionary used was created by Jan Rouš from the parallel corpus CzEng (Bojar et al., 2008) and other sources as a replacement for the PCEDT dictionary (Cuřín et al., 2004).

There were also rules for translation of some types of lemmas (e.g. numerals) that would not be translated correctly with the dictionary.

6.1.2 New implementation

Our new design of the translation of lemmas and formemes is more modular. We have created 10 new blocks which can be combined in various translation scenarios. All the new blocks indicate in their names whether they serve to translate lemmas (`L`), formemes (`F`) or both (`LF`).

We have added almost all rules from the original blocks `Baseline_formeme_translation` and `Baseline_tlemma_translation` also to our new blocks. Apart from the modularity, the most noticeable difference from the old implementation is that we save all translation variants proposed by the dictionaries to the attribute `translation_model`. This structured attribute comprises of two n-best lists: `t_lemma_variants` and `formeme_variants`. The variants are ordered by their probability and the first one is always saved also in the attribute `t_lemma` or `formeme`, respectively.

Overview of new translation blocks

- `Translate_F_try_rules` – rules to be applied before querying the dictionaries.
- `Translate_F_add_variants` – formeme translation variants from the formeme-to-formeme dictionary are filled.
- `Translate_F_rerank` – formeme translation variants are reranked with the valency formeme dictionary.
- `Translate_F_fix_by_rules` – rules for fixing translation of formemes that are unacceptable in Czech. E.g. formeme `adj:attr` as a translation of `n:attr` is unacceptable for a name of person (attribute `is_name_of_person`=1).
- `Translate_L_try_rules` – rules to be applied before querying the dictionary, e.g. ordinal numerals (*1st, 32nd, 999th*) can be translated by a simple rule (to *1., 32., 999.*).
- `Translate_L_add_variants` – lemma translation variants from the dictionary `TranslationDict::EN2CS` are filled.
- `Translate_L_filter_aspect` – verbal lemmas whose aspect is incompatible with the given context are removed from translation variants.
- `Translate_LF_numerals_by_rules` – chooses the best translation variant of numerals by a rule and deletes the other variants.
- `Translate_LF_phrases` – see page 65.
- `Translate_LF_tree_Viterbi` – see Section 6.3.

Marking of origin

We have established two new attributes – `t_lemma_origin` and `formeme_origin`. Every block that changes `t_lemma` must change also `t_lemma_origin` (and similarly for formemes). Let us explain the purpose of these attributes on an example:

The block `Translate_L_try_rules` has a rule for translating decimal points to decimal commas, which are used in Czech to separate the fractional part of numbers. If this rule succeeds it sets the `t_lemma_origin` to the value `rule-Translate_L_-try_rules`. Other blocks that translate lemmas skip nodes whose `t_lemma_origin` starts with `rule`.

Other improvements

The block `Fix_verb_reflexivity` was intended to change some verbs into reflexive based on a heuristic rule. We have found that with our new implementation of the transfer, this block makes more errors than corrections, so we have deleted it from the scenario.

The block `Move_dicendi_closer_to_dsp` was intended to change word order in sentences with direct speech; namely, it moves *verba dicendi* (e.g. Czech equivalents of *say, ask, utter...*) immediately after the direct speech. We have adapted this block to handle also indirect speeches. For example, *It is bad, Marco says* should be translated as *Je to špatné, říká Marco* (not as *Je to špatné, Marco říká*). We have also fixed errors, when the block incorrectly moved words that should not be moved.

We have improved blocks `Fix_date_time` and `Reverse_number_noun_dependency`. The latter was also removed from the transfer phase to the synthesis phase, because it suites better the concept of the tectogrammatical layer.

6.1.3 Evaluation

The greatest improvement in the transfer is the introduction of Hidden Markov Tree Models and the related block `Translate_LF_tree_Viterbi`, which is discussed in the following three sections.

Modification	diff (BLEU)	diff (NIST)
no `Translate_LF_tree_Viterbi`	0.0130	0.2483
no `Translate_LF_numerals_by_rules`	0.0017	0.0380
no `Translate_L_filter_aspect`	0.0010	0.0151
added `Fix_verb_reflexivity`	0.0002	0.0157
original `Move_dicendi_closer_to_dsp`	0.0013	0.0068
original `Reverse_number_noun_dependency`	0.0005	0.0040
original `Fix_date_time`	0.0016	0.0082
all above together	0.0153	0.2879
original transfer (`Baseline...`)	0.0171	0.4189

Table 6.1: *Modifications of the transfer*
For explanation see Section 4.5.

6.2 Hidden Markov Tree Models

6.2.1 Motivation

Most errors are caused by the transfer of lemmas and formemes

In the manual annotation of translation errors we have discovered that more than half of all errors are caused by the transfer phase (Table 3.4) and 92% of these errors are wrong lemmas and wrong formemes (Table 3.8). The choice of correct lemma and formeme is of course very difficult task and the quality of translation depends heavily on the quality of the dictionaries used. However, even with an ideal dictionary many errors will occur if we just select the most probable variant for each node without considering the context.

Two meanings of the word *speaker*

For example, word *speaker* with the sense *loudspeaker* should be translated as *reproduktor* and according to the lemma dictionary used in our scenario[6] the translation probability is $P(reproduktor|speaker) = 0.45$. When the sense is *spokesperson*, the correct translation is *mluvčí* and $P(mluvčí|speaker) = 0.26$. Perhaps, there were more texts about loudspeakers than texts about spokespersons in the CzEng parallel corpus upon which the dictionary is based. The original implementation translates every word *speaker* as *reproduktor*, so we encounter errors in phrases like *speaker for the Ministry of Transport*.

Should we use a word sense disambiguation tool?

The problem with wrong word sense disambiguation is very characteristic for machine translation in general. One solution of this problem would be to use an automatic word sense disambiguation tool in the analysis phase. For example, we can use lemma speaker-1 for spokesperson and speaker-2 for loudspeaker (and maybe also speaker-3 for the sense *one that speaks*, e.g. *native speaker*). There are two major drawbacks of this solution:

- It is not clear what senses for a given word should be distinguished – how detailed classification do we want. The annotation guidelines will be in any case a matter of convention and a source of controversy. Different tasks have different needs for the classification. Especially in the MT, we would like to distinguish (just) those senses that are expressed by different words in the target language. However, this requirement goes against the idea of universal analysis phase independent on the target language.

  It it not feasible to distinguish senses that are expressed by different words in *any* target language (that is one of the reasons why interlingua-based MT is not feasible). For example, consider possible translations of the verb *cut*. There are at least thirty Czech equivalents – different lemmas are used for cutting grass (*sekat*), trees (*kácet*), flowers (*řezat*), hair (*stříhat*), beard (*holit*), bread (*krájet*), prices (*zlevnit*), taxes (*snížit*), corners etc. Moreover, different lemmas are used for different grammatical aspects and other variations in the

[6] `share/resource_data/translation_dictionaries/czeng_with_backward_probs.dict`

meaning (*sekat, nasekat, posekat, dosekat, osekat, seknout, useknout, vyseknout, rozseknout...*). And finally, there are other languages that may have other criteria for classification of senses of the word *cut*.

- It is difficult to develop a high-quality automatic word sense disambiguation tool. As we have shown in the previous paragraph, word sense classification will never be detailed enough, so context should be considered in the transfer phase anyway. But if we have a transfer method capable of choosing right translation based on the context, we can use it also for the word sense disambiguation. A substantial disadvantage of a word sense disambiguation performed in the analysis phase is that it can exacerbate data sparseness. Since the word sense disambiguation is not suited for a particular target-language, it can differentiate also some senses that could be correctly translated to the same lemma.

Linear context and tree context

In phrase-based MT, the context used to select the best translation of a word is *linear* – basically, the context is a phrase, i.e. a string of surrounding words. There are some experiments with "phrases with gaps" (Simard et al., 2005), but in most systems a phrase is defined as a contiguous string of words (not necessarily forming a phrase in a linguistic sense).

We believe that it is more appropriate to use a local *tree* context, i.e. the children and the parent of a given node. Not only that it is appropriate according to linguistic intuition, but it should help us to face the data sparseness.

For illustration, consider the before-mentioned example with the phrase *speaker for the Ministry of Transport*. Human translators recognize from semantics that the *speaker* is a human being (not a loudspeaker) and translate it as *mluvčí*. Phrase-based MT systems can learn the whole phrase or possibly just the phrase *speaker for the Ministry*, but they must also learn phrases like *speaker of the Ministry*, *speaker for the Chinese Ministry*, *speaker for the Foreign Ministry*, *speaker for the Indian External Affairs Ministry* etc. in order to translate them correctly.[7]

When using the local tree context, we can for example learn that *speaker* should be translated as *mluvčí* if it has a child node with the lemma ministry. This way we cover all the before-mentioned phrases including the unseen ones. Another knowledge learned from a parallel dependency treebank may be that *speaker* should be translated as *mluvčí* if its parent node has the lemma name (e.g. in phrases *speaker's name, name of the next speaker*) or that *speaker* should be translated as *reproduktor* if its parent node has the lemma buy (e.g. in a phrase *buy an expensive speaker*).

[7]The example if oversimplified. First, in phrase-based MT systems, it is the target-language model that should cover such long phrases, so it would be more accurate to present Czech translations of the phrases. Second, we suppose that the hypothetical phrase-based system is trained on the same parallel corpus as our dictionary, so $P(reproduktor|speaker) > P(mluvčí|speaker)$ and similarly for backward probabilities $P(speaker|reproduktor) > P(speaker|mluvčí)$. Otherwise, there would be no need for the language model to cover the phrases, if the translation model itself would choose the correct translation. Third, since the phrases learned by phrase-based MT systems are usually not constrained to linguistically adequate constituency phrases, it is possible that the system will learn that *speaker of the* should be translated as *mluvčí*. However, there are plenty of more relevant examples of long-distance dependencies, that are not covered even by 6-gram or 7-gram language models.

How to learn, represent and use tree context?

The obvious question is how can we learn, represent and use such knowledge. The preceding paragraph formulates the knowledge in a form of rules. Although this approach could be used in MT (rules can be automatically learned from the treebank), it is difficult to combine it with probabilistic methods. We have decided to represent the knowledge in a form of a model that describes the probability of a node given its parent node. More precisely, we model the probability of a lemma and formeme of the dependent node given a lemma and formeme of the governing node.

The model can be learned from a treebank using maximum likelihood estimate, but similarly to traditional (linear) language models it is necessary to smooth the probabilities and there are many possible ways how to perform the smoothing. We discuss this topic in Section 6.4.

Tree context: bilingual or target-language?

The probabilistic model introduced in the previous paragraph is a monolingual tree model and can be learned from a target-language treebank (Czech in our case). With the availability of parallel treebanks we can develop also "bilingual tree models". An example of bilingual tree model is the valency formeme translation dictionary. It specifies the probability of formeme of the target-side node given formeme of the source node and lemma of the source node's parent.

Ideally, we would like to use more complex bilingual tree model that defines also target-side lemmas and is conditioned also by other attributes (lemma of the source node, lemmas of its children etc.). This complex model would supersede both formeme and lemma dictionaries as well as the target-language tree model. However, we do not have enough parallel data to reliably train such a model. Since the amount of monolingual training data is much greater, we try to exploit it as much as possible.

First attempts at using tree context

In the original implementation of transfer phase in TectoMT (blocks `Baseline_tlemma_translation` and `Baseline_formeme_translation`), the only usage of tree context was in the valency formeme translation dictionary. Moreover, lemmas and formemes were translated almost independently – there was only a rule to check for compatibility of a lemma with a formeme, but no probabilistic model describing their joint or conditional probability. In other words, the target-language tree model was not used in the original implementation.

One of the first attempts at exploiting the target-language tree model was implemented in the block `Improve_translation_by_tree_LM` by Zdeněk Žabokrtský with the usage of `LanguageModel::LModel` module by Václav Novák. It performed a top-down depth-first traversal through the t-tree translated by the baseline blocks. Its main idea was to choose the best lemma and formeme according to a loglinear combination of three models: translation probability of lemma, translation probability of formeme and target-language tree model. The main difference from the tree-modified Viterbi algorithm presented in Section 6.3 is that the top-down traversal allows only local optimization based on the parent node (but no children nodes), whereas the tree-modified Viterbi algorithm searches for the global maximum.

Why do we need Hidden Markov Tree Models?

The apparent weak point of the before-mentioned top-down traversal occurs when the correct lemma or formeme can be determined only from the children, but not from the parent (e.g. *He is a speaker for the ministry* versus *It is an expensive speaker*). Of course, if we use a similar algorithm with bottom-up traversal, these cases will be handled correctly, but errors will be introduced in the opposite cases – when the correct lemma or formeme can be determined only from the parent, but not from children (e.g. *according to the speaker* versus *buy a speaker*).

Not only that both the types of cases (parent/children are important for translation) are frequent, but sometimes we need to know the parent as well as children to choose the correct translation. The child-parent dependencies are chained in the tree, so we need to find the combination of lemmas and formemes, which results in the maximal global probability of the whole tree. Hidden Markov Tree Models provide a theoretical background for the tree-modified Viterbi algorithm, which can efficiently find the global maximum.

6.2.2 Related work

Hidden Markov Models (HMM, see Chapter 9 in Manning and Schütze (1999))[8] belong to the most successful techniques in Computational Linguistics. There are many modifications of HMM: arc-emission versus state-emission, epsilon-emission (Bahl et al., 1990), HMM with Gaussian distribution of emission function etc. Hierarchical Hidden Markov Models, which are used for Information Extraction (Skounakis et al., 2003), make use of tree structures, but they still primarily work with linearly organized observations/states.

Hidden Markov Tree Models (HMTM) were introduced by Crouse et al. (1998), and used in applications such as image segmentation, signal classification, denoising and image document categorization. More information about HMTM can be found in Diligenti et al. (2003) and in Durand et al. (2004). The latter article contains also a detailed explanation of the tree-modified Viterbi algorithm. Parts of this Chapter are based on Žabokrtský and Popel (2009), where HMTM are introduced to dependency-based MT.[9]

6.2.3 Formal description of HMTM

Suppose that

- $V = \{1, \ldots, |V|\}$ is a set of tree nodes, $r \in V$ is the root node and $\rho : V \setminus \{r\} \rightarrow V$ is a function determining the parent node of each non-root node.
- $\boldsymbol{X} = (X_1, \ldots, X_{|V|})$ is a sequence of random variables taking values from a state space S. Random variable X_v is understood as a *hidden state* of the node v and $P(X_v|X_{\rho(v)})$ is called *transition probability.*

[8] To avoid any terminological confusion, we should note that by HMM we mean only Hidden Markov *Chain* Models.

[9] The idea to use HMTM in MT originates from Zdeněk Žabokrtský. The paper is a joint work. The implementation presented in this chapter was created entirely by the author of this thesis.

- $\boldsymbol{Y} = (Y_1, \ldots, Y_{|V|})$ is a sequence of *observable symbols* taking values from an alphabet K. $P(Y_v|X_v)$ is called *emission probability*.

We further introduce the following notation:

- $subtree : V \to 2^V$ is a function mapping node v to a set of all nodes of the subtree rooted in v, i.e.
  $subtree(v) = \{w \in V : \exists w = z_1, \ldots, z_n = v, \forall i \in \{1 \ldots n-1\} \quad \rho(z_i) = z_{i+1}\}$.
- $\mathbf{X}(v)$ is a sequence of hidden states of the subtree rooted in v, i.e.
  $\mathbf{X}(v) = \{X_w : w \in subtree(v)\}$.
  Hence $\mathbf{X} = \mathbf{X}(r) = \{X_r, \mathbf{X}(w) : \rho(w) = r\}$.
- Analogously, $\mathbf{Y}(v)$ is a sequence of symbols of the subtree rooted in v.

Similarly to stationary first-order state-emitting HMM, we state three independence assumptions for HMTM:

1. **stationary property** (analogy to time invariance property of HMM)
   $\forall v, w \in V \setminus \{r\} : P(X_v|X_{\rho(v)}) = P(X_w|X_{\rho(w)})$ &
   $\forall v, w \in V : P(Y_v|X_v) = P(Y_w|X_w)$
   i.e. transition and emission probabilities are independent of nodes.
2. **tree-Markov property** (analogy to limited horizon property of HMM)
   $\forall v \in V \setminus \{r\}, \forall w \in V \setminus subtree(v) : P(\mathbf{X}(v)|X_{\rho(v)}, X_w) = P(\mathbf{X}(v)|X_{\rho(v)})$
   i.e. given $X_{\rho(v)}$, all hidden states of the subtree rooted in v are conditionally independent of any other nodes.
3. **state-emission property**
   $\forall v, w \in V : P(Y_v|X_v, X_w, Y_w) = P(Y_v|X_v)$
   i.e. given X_v, Y_v is conditionally independent of any other nodes.

Let $v_1, \ldots, v_n$ be children of the root r, then using the tree-Markov property and mathematical induction we get:

$$\begin{aligned} P(\mathbf{X}) &= P(X_r, \mathbf{X}(v_1), \ldots, \mathbf{X}(v_n)) \\ &= P(X_r)P(\mathbf{X}(v_1), \ldots, \mathbf{X}(v_n)|X_r) \\ &= P(X_r)P(\mathbf{X}(v_1)|X_r)P(\mathbf{X}(v_2), \ldots, \mathbf{X}(v_n)|X_r, \mathbf{X}(v_1)) \\ &= P(X_r)P(\mathbf{X}(v_1)|X_r)P(\mathbf{X}(v_2), \ldots, \mathbf{X}(v_n)|X_r) \\ &= P(X_r)P(\mathbf{X}(v_1)|X_r) \ldots P(\mathbf{X}(v_n)|X_r) \\ &= P(X_r) \prod_{v \in V \setminus \{r\}} P(X_v|X_{\rho(v)}) \end{aligned} \tag{6.1}$$

Using the state-emission property and mathematical induction we get:

$$\begin{aligned} P(\mathbf{Y}|\mathbf{X}) &= P(Y_r|\mathbf{X})P(\mathbf{Y}(v_1), \ldots, \mathbf{Y}(v_n)|\mathbf{X}(v_1), \ldots, \mathbf{X}(v_n), X_r, Y_r) \\ &= P(Y_r|X_r)P(\mathbf{Y}(v_1), \ldots, \mathbf{Y}(v_n)|\mathbf{X}(v_1), \ldots, \mathbf{X}(v_n)) \\ &= \prod_{v \in V} P(Y_v|X_v) \end{aligned} \tag{6.2}$$

From Equations 6.1 and 6.2 we can deduce the following factorization formula:

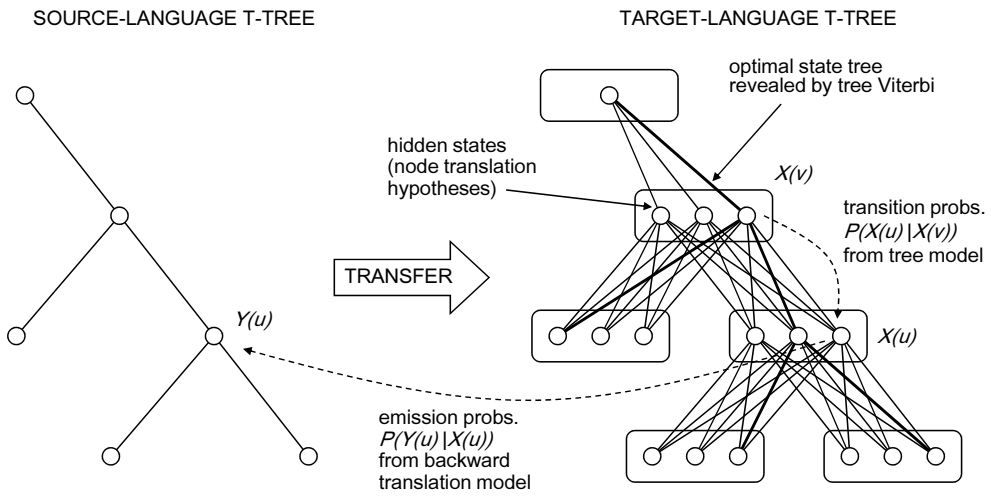


Figure 6.1: *Tectogrammatical transfer as a task for HMTM.*

$$P(\boldsymbol{Y}, \boldsymbol{X}) = P(Y_r|X_r)P(X_r) \cdot \prod_{v \in V \setminus \{r\}} P(Y_v|X_v)P(X_v|X_{\rho(v)}) \qquad (6.3)$$

We see that HMTM (analogously to HMM, again) is defined by the following parameters:[10]

- $P(X_v|X_{\rho(v)})$ – transition probabilities between the hidden states of two tree-adjacent nodes,[11]
- $P(Y_v|X_v)$ – emission probabilities.

6.2.4 Application of HMTM in MT

How to estimate emission and translation probabilities?

When using HMTM in MT, labels of the source-language nodes can be interpreted as observable symbols and labels of the target-language nodes can be interpreted as hidden states (see Figure 6.1). In the case of TectoMT transfer, a label of a node is a pair of lemma and formeme. Therefore, the hidden states space (S) is the Cartesian product of lemmas and formemes possible for the target language and the alphabet of observable symbols (K) is the Cartesian product of lemmas and formemes possible for the source language.

HMTM emission probabilities can be estimated from the "backward" (source given target) node-to-node translation model. This node-to-node translation model can be further estimated by factorization to the lemma translation dictionary and formeme translation dictionary.

[10] As follows from the stationary property, the parameters are independent on the node v.

[11] The need for parametrizing also $P(X_r)$ (prior probabilities of hidden states in the root node) can be avoided by adding an artificial root whose state is fixed.

HMTM transition probabilities can be estimated from the target-language tree model, which will be described in Section 6.4.

The decomposition into *translation model* and *language model* proved to be extremely useful in Statistical Machine Translation since Brown et al. (1993). It allows to compensate the lack of parallel resources by the relative abundance of monolingual resources.

Limitations of HMTM

There are several limitations implied by the definition of HMTM, which we have to consider before applying it to MT.

The first limitation is merely a technical detail. The set of hidden states and the alphabet of observable symbols are supposed to be finite. This assumption can be easily fulfilled by introducing an artificial symbol/state for unknown tokens. However, in practice we are able to consider only a limited number of possible hidden states for each node (see Section 6.3.2), so the trick with an artificial symbol is not actually needed.

More serious limitations are induced by the three independence assumptions:

- **stationary property**
  We assume that the position of a node in a tree cannot influence its translation and emission probabilities. For example, this property would be violated if some words should be translated differently when being children of the main clause verb (i.e. grandchildren of the technical root).[12] According to our observations, such a dependence on node's level (i.e. distance from the root) is not a substantial issue.

  Another violation of the stationary property can be a dependency on word order. For example, some words should be translated differently when being on the beginning of a sentence.[12] These cases are also not a substantial problem.[13]

- **tree-Markov property**
  This assumption concerns only the target-language tree model. The conditional dependency (in the probabilistic sense) of a node on its parent corresponds well to the intuition behind dependency relations (in the linguistic sense) in dependency trees.

  However, there are special linguistic phenomena that violate this assumption. These phenomena are addressed in the manual for English tectogrammatical annotation (Cinková et al., 2006, pp. 11–23) in Sections: Non-dependency edges, Dual dependency and Ambiguous dependency.

  Treatment of coordinations (which are represent using non-dependency edges) is discussed separately in Section 6.2.5. Predicative complements have the so-called dual dependency – on a verb and on a semantic noun, but only the

[12] . . . and this difference could be determined neither from the source node nor the target-side parent node.

[13] PDT-style tectogrammatical nodes have an attribute `deepord`, which specifies the so-called *deep word order* for the purpose of communication dynamism. TectoMT tectogrammatical trees use this attribute for surface word order. Nevertheless, if there was a reason, the attribute could be incorporated to the source node's label to circumvent the violation of the stationary property.

former is represented by a tree edge.[14] In the following examples[15] we mark the predicative complement with an underline; its second dependency is always the subject (*He*). *He spoke of him as of his father. He left whistling. He lives alone.*

Although not considered a dual dependency, copula constructions also violate the assumption. For example, in sentences *He is a speaker.* and *It is a speaker.* we can disambiguate the sense of the object (*speaker*) based on the subject (*He* or *It*), but these nodes are siblings, so the probabilistic dependency cannot be directly used in HMTM.

A possible solution to circumvent these violations and perspectively improve the translation quality is to incorporate the before-mentioned secondary dependencies into the labels of source nodes to be handled by the translation model.

- **state-emission property**
  This property can be weakened to "arc-emission property":
  given X_v and $X_{\rho(v)}$, Y_v is conditionally independent of any other nodes, i.e.
  $\forall v, w \in V : P(Y_v|X_v, X_{\rho(v)}, X_w, Y_w) = P(Y_v|X_v, X_{\rho(v)})$

  A factorization formula, analogical to Equation 6.3, can be then proved:

$$P(\boldsymbol{Y}, \boldsymbol{X}) = P(Y_r|X_r)P(X_r) \cdot \prod_{v \in V \setminus \{r\}} P(Y_v|X_v, X_{\rho(v)})P(X_v|X_{\rho(v)}) \qquad (6.4)$$

  With this generalization we can condition emission probabilities (i.e. translation model) on the parent node. Another (actually equivalent) method how to use richer translation model, without the need of weakening the state-emission property, is to incorporate the needed attributes to the labels of target-side nodes.

The most limiting assumption from the MT viewpoint was not expressed explicitly yet:

- **isomorphism presumption**
  The source-language tree and the target-language tree are required to be isomorphic. In other words, only node labeling can be changed in the HMTM transfer step. This assumption concerning the tree isomorphism is problematic. As we have shown in Chapter 3, there are cases when it is not possible to translate a sentence correctly without violating the isomorphism presumption. On the other hand, only 8% of all translation errors in our annotation experiment were caused by such cases. Possible solutions to the problem are discussed in Section 6.2.6.

6.2.5 Treatment of coordinations

The parent of coordination members in tectogrammatical trees is the coordination head (usually a conjunction *and, or*, see Figure 5.3 on page 49). The effective

[14] The latter dependency relation is indicated by the attribute `compl.rf`.

[15] We present English examples, but since the violations concern the target-language tree model, it would be more accurate to present Czech equivalents.

parent (i.e. the node that should govern the members from a linguistic point of view) of the members is their grandparent or some higher ancestor in case of nested coordinations.

Suppose we want to translate the sentence *I will buy a microphone and a speaker*. If we use the original topology of t-trees, the target-language tree model will be queried for a transition probability $P(speaker|and)$.[16] However, it seems to be more appropriate to consider the effective parent – $P(speaker|buy)$.

In order to solve this problem, we have created a pair of blocks: `Rehang_to_eff_parents` and `Rehang_to_orig_parents`. The former changes the topology of t-trees, so topological parents are actually the effective parents. The original topology is saved in an attribute `original_parent.rf` to be able to restore it with the latter block.

Hence the part of translation scenario responsible for the HMTM transfer step consists of three blocks:

```
SEnglishT_to_TCzechT::Rehang_to_eff_parents
SEnglishT_to_TCzechT::Translate_LF_tree_Viterbi
SEnglishT_to_TCzechT::Rehang_to_orig_parents.
```

6.2.6 HMTM and non-isomorphic transfer

We propose three methods how to solve the cases where it is needed to translate one source node to more target nodes, more source nodes to one target node or generally more source nodes to more target nodes. These methods are of course designed to be combined with the HMTM transfer block and they can also combined together.

1. **deterministic rules after the HMTM transfer step**
   An example of this method is the original block `Override_pp_with_phrase_translation`. It exploits a translation dictionary of prepositional phrases such as *at all* → *vůbec*, *on the other hand* → *na druhé straně*, *for Christ's sake* → *proboha* etc. If the block finds a subtree on the source a-layer that matches an entry in the dictionary, the equivalent subtree on the target t-layer is substituted with one artificial node, whose lemma is filled with the Czech phrase.[17] Although this block does not use the translation made by the HMTM transfer block, there are other blocks (e.g. `Fix_date_time`) that do.

2. **deterministic rules before the HMTM transfer step**
   An example of this method is the new block `Translate_LF_phrases`. It should serve a similar purpose as `Override_pp_with_phrase_translation`, but it is not limited to prepositional phrases. In present, it is just a proof of concept covering only a few rules to translate *this year* as *letos*, *letošní* or *tento rok* depending on a context. The main difference from `Override_pp_with_phrase_translation` is that our new block creates standard t-nodes, which can be used in the HMTM transfer block to disambiguate neighbouring nodes.

[16] For simplicity, we use here notation $X_v = speaker$ and $X_{\rho(v)} = and$. The target-language tree model will be actually queried for Czech lemmas and formemes: $P(X_v = (mluvčí, \mathsf{n{:}4})|X_{\rho(v)} = (and, \mathsf{x}))$, $P(X_v = (reproduktor, \mathsf{n{:}4})|X_{\rho(v)} = (and, \mathsf{x}))$ etc.

[17] This block goes against several ideas of tectogrammatical translation, but it used to improve BLEU scores in the original translation scenario. In the new scenario with the HMTM transfer block its effect is negligible, so we have deleted the block from the scenario.

3. **generate more variants to be used by the HMTM transfer step**
   This last method was not implemented yet, but seems to be most adequate. Instead of writing linguistic rules for choosing between *letos*, *letošní* and *tento rok*, we can learn these phrases automatically from a parallel treebank, add them as translation variants and let the HMTM transfer block to choose the best one. This can be done if all the phrases comprise the same number of nodes (and have the same topology). In the opposite case (like in our example) we could treat some nodes as one artificial to formally fulfill the requirement of same topology. For example, we can treat nodes *tento* and *rok* as a one node, which has its real topology saved in an attribute. This attribute would be used for computing transition probabilities. However, it is an open question how to actually compute the transition probability – i.e. the target-language tree model probability of this artificial node being a child of another node and vice versa.

6.3 Tree-modified Viterbi algorithm

6.3.1 The algorithm

Naturally the question appears how to restore the most probable hidden tree labeling $\hat{\mathbf{X}}$ given the observed tree labeling $\mathbf{Y}$ (and given the tree topology, of course). Using the factorization formula from Equation 6.3, we can write:

$$\begin{aligned}\hat{\mathbf{X}} &= \arg\max_{\mathbf{X}} P(\mathbf{X}|\mathbf{Y}) \\ &= \arg\max_{\mathbf{X}} P(\mathbf{X},\mathbf{Y}) \\ &= \arg\max_{\mathbf{X}} P(Y_r|X_r)P(X_r)\cdot\prod_{v\in V\setminus\{r\}} P(Y_v|X_v)P(X_v|X_{\rho(v)})\end{aligned} \tag{6.5}$$

Similarly to the classical Viterbi algorithm, we can use dynamic programming to achieve an effective implementation – $\mathcal{O}(|V|\cdot K^2)$ for $|V|$ nodes and K states considered for every node.

However, we cannot start at the root node and perform top-down traversal, which would be the most straightforward analogy to the classical Viterbi algorithm. Instead, the tree-modified Viterbi algorithm starts at leaf nodes and continues upwards, storing in each node for each state and each its child the optimal downward pointer ("backpointer") to the child's hidden state. When the root is reached, the optimal state tree is retrieved by downward recursion along the pointers from the optimal root state.

6.3.2 Implementation

In practice, HMTM serves us as an inspiration, but for pragmatic reasons the implementation differs in four aspects from the theory:

- **Logarithmic space**
  Instead of multiplying probabilities, we sum logarithms of the probabilities. This is an usual practice, which facilitates computing with very small numbers.
- **Computing emission probabilities**
  We use a factorization of the translation model into two channels: lemmas and formemes. Moreover, we use an forward translation model (target given source) in addition to the backward translation model (source given target), because it proved to to have positive effect on the translation quality. The emission probability is computed as a weighted average of the models.
- **Computing transition probabilities**
  Target-language tree model probabilities must be smoothed. This is discussed in Section 6.4.
- **Number of considered states**
  We cannot consider all possible hidden states for a node, when running the tree-modified Viterbi algorithm. We consider only L most probable[18] lemmas

[18]The probability used to sort translation variants is obtained from the forward translation model.

and F most probable formemes, so there are $L \cdot F$ labels (hidden states being considered) for a node. After several experiments we chose $L = 6, F = 6$ as a good balance between speed and performance.

Another speed-up is achieved by pruning states that have incompatible lemma and formeme (so the number of states per node can be lower than $L \cdot F$). The incompatibility is determined according to the PoS of the lemma and semantic PoS that is encoded in the formeme. For example, verbal lemmas are compatible only with formemes that start with `v`.

```
sub Tree_Viterbi( $node ) {
  foreach my $child ($node->get_children()) {
    Tree_Viterbi($child);

    foreach my $my_state ( $node->get_states() ) {
      my ( $max_score, $best_child_state ) = ( -INFINITY, undef );

      foreach my $child_state ($child->get_states()) {
        my $score = $child_state->get_score();
        $score += $child_state->get_logprob_given_parent($my_state);
        if ( $score > $max_score ) {
          ( $max_score, $best_child_state ) = ( $score, $child_state );
        }
      }
      $my_state->add_backpointer($best_child_state);
      $my_state->increment_score($max_score);
    }
  }

  foreach my $my_state ( $node->get_states() ) {
    $my_state->increment_score($my_state->get_logprob());
  }

  return @states;
}
```

Figure 6.2: *Implementation of the tree-modified Viterbi algorithm*
The presented pseudo-Perl code is only a slightly modified version of the real code. Method get_logprob_given_parent returns the transition probability. Method get_logprob returns the emission probability – the method has no argument, because a reference to the observable symbol (i.e. the source node) is saved in each node.

The tree-modified Viterbi algorithm itself is implemented in the module `TreeViterbi` in an universal way, so it can be used for various purposes (not only MT). The main subroutine is presented in Figure 6.2.

The block `Translate_LF_tree_Viterbi` uses this modul and supplies all methods specific for the usage in MT:

- a method returning all states that should be considered for a given node
- a method of the states returning their emission probabilities
- a method of the states returning their transition probabilities
- a method of the states that stores a backpointers to the best child states

The main subroutine of `TreeViterbi` returns the optimal state of the root node. The root is a technical node with only one possible state, which is of course the returned optimal state, but it has now filled backpointers. By recursive following these backpointers the optimal state tree is reconstructed. This means that for every node the selected lemma (formeme) is filled in the attribute `t_lemma` (`formeme`) and if it differs from the value that was there before, also the attribute `t_lemma_origin` (`formeme_origin`) is set to the value viterbi.

6.4 Target-language tree model

6.4.1 Related work

N-gram language models

A statistical language model is usually defined as a probabilistic function on a sequence of words. Probably best known are n-gram language models, which predict the probability of a word given $n-1$ preceding words. The probability of a whole sequence of words is then approximated using $(n-1)^{\text{th}}$ order Markov property as a product of probabilities of individual words conditioned by their limited histories.

Factored language models

Factored language models (FLM, described in Bilmes and Kirchhoff (2003)) still predict the probability of a sequence, but each member of the sequence (each word) is viewed as a vector of k factors, for example word form, lemma and PoS tag. Standard class-based language models are an example of two-factor FLM. Smoothing of FLM is more complex than with classical n-gram models, because there is no obvious (e.g. temporal) natural backoff order. Classical word-only language models usually drop first the "oldest" word of the history, then the second oldest and so on until only unigrams remain. FLM can, for example, first drop word form and lemma of the oldest word, then word form of the second oldest, then PoS of the oldest etc.

In order to fit the definition of (statistical) language models, FLM must be able to predict the probability of a plain sequence of words. This means that there must be a deterministic method how to gather all factors from the sequence of words.

Dependency language models

According to our knowledge, there is no universal (generally accepted) definition of the term *dependency language model*, but there are papers (Chelba et al., 1997; Shen et al., 2008) that use the term (with different meanings though). Nevertheless, the common idea behind the term is to use some kind of dependency grammar and dependency trees for language modeling.

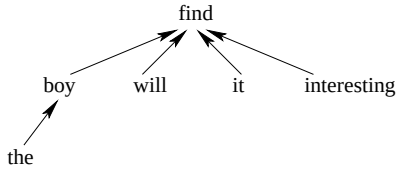
There are many possible ways how to exploit dependency trees in a dependency language model.

- It can be used as an additional factor to n-gram-based models. The alternative is to disregard the neighbouring words unless they are in a dependency relation (parent or child) with the current word (i.e. the word that is being predicted).

- Generally in language modeling, the probability of a whole sentence is factored (using the "chain rule") to a product of conditional probabilities. In n-gram language models this factorization is left-to-right and words with index $i-n$ or lower are disregarded (where i is the index of the current word). In factored language models the factorization is also left-to-right and all factors with index $i-n$ or lower are disregarded, but the remaining vectors of factors or used for smoothing as described above. In dependency language models the factorization can be either left-to-right or bottom-up.

Left-to-right style is used in Chelba et al. (1997) – briefly, the context considered for a word comprises the preceding bigram and a *link stack*, which is a list of words that precede the current word, but their parent does not.

Bottom-up style is used in Shen et al. (2008). The probability of a tree (which represents the given sentence) is computed using probabilistic distributions P_L and P_R for left and right side generative probabilities respectively and P_T for a probability of a word being the root. See the following example for illustration:



$$\begin{aligned} Prob = & P_L(the|boy\text{-as-parent}) \\ & \times P_L(boy|will, find\text{-as-parent}) \times P_L(will|find\text{-as-parent}) \\ & \times P_T(find) \\ & \times P_R(it|find\text{-as-parent}) \times P_R(interesting|it, find\text{-as-parent}) \end{aligned}$$

The probability of a word is conditioned by its parent and also siblings that lie between the word and the parent. Apparently, it is assumed that left children of a node are independent on right children.

- Is the dependency language model used to predict the probability of a sentence or the probability of its tree? The latter case is actually equivalent to predicting the joint probability of a sentence and its tree, because the sentence can be unambiguously derived from its tree. If a particular type of dependency language model is to be considered a (statistical) language model, it must be able to predict the probability of a sentence. This can be formally done by stating that the tree can be obtained by a deterministic method from the sentence. Another question is whether such interpretation of the probabilities predicted by the model is appropriate.

  In Chelba et al. (1997), the sentence is denoted as S an its dependency tree as K. The problem is solved by the following statement:

  > *Our aim is to develop an expression for the joint probability* $P(S, K)$. *In principle, we can then recover* $P(S)$ *as the marginal* $\sum_K P(S, K)$. *In practice, we make the assumption that the sum is dominated by a single term* $P(S, K^*)$, *where* $K^* = \arg\max_K P(S, K)$, *and then approximate* $P(S)$ *by* $P(S, K^*)$.

6.4.2 Definition of target-language tree model

Similarly to before-mentioned dependency language models, we want to develop a probabilistic model that exploits dependency trees. The trees we use are tectogrammatical and we are interested only in the probability of a node given its parent node. Although it would be possible to parse a sentence, use our model to compute the overall probability of the tree and interpret it as a probability of the sentence,

this is not the purpose of our model.[19] Therefore, we have decided to call it *target-language tree model* rather than *dependency language model* or *tectogrammatical language model*.

The primary motivation for building the model was to use it in the tree-modified Viterbi algorithm for English-Czech translation, but it could be useful also in other areas (translation from other languages, parsing,...).

6.4.3 Implementation

Training

We used a file `joint_table.tsv.gz` created by Václav Novák from a large amount of Czech texts (roughly 800 million words, mostly from the Czech National Corpus[20] and from WWW), analyzed in TectoMT up to the t-layer. Each line of this file represents one t-layer edge by five values: lemma of the dependent node (`Ld`), lemma of the governing node (`Lg`), formeme of the dependent node (`Fd`), PoS of the governing node (`Pg`) and PoS of the dependent node (`Pd`). The formeme of the governing node (`Fg`) is not included, because the model $P(Ld, Fd|Lg, Fg)$ would be extremely sparse and the approximation $P(Ld, Fd|Lg, Fg) \approx P(Ld, Fd|Lg)$ seems to be suitable.[21]

PoS is represented by one letter – the first position of Czech morphological tags (N = noun, V = verb, D = adverb etc.). We added these PoS letters to lemmas in order to disambiguate cases when the same lemma can have different PoS letters.[22] Afterwards we lowercased the lemmas and converted to unique ID numbers. This is just a technical step to improve memory and speed efficiency (frequent lemmas have lower IDs).

We filtered out edges appearing only once in the data and also edges where one of the nodes is a number written in digits. This reduced the number of edges to 35%.

We created two files: `c_LgFdLd.pls.gz` and `c_PgFdLd.pls.gz`. The former contains aggregated counts of triples `Lg, Fd, Ld`, pairs `Lg,Fd` and unigrams `Lg`. The latter has a similar structure, but instead of lemmas of governing nodes it contains only their PoS.

[19] Of course, such interpretation would not be adequate from the linguistic point of view. Apart from the limitations induced by the tree-Markov property and discussed on page 63, our tree model, for example, ignores the word order. On the other hand, n-gram language models also ignore many aspects adequate from the linguistic point of view. Perhaps, the notion 'probability of a sentence' is not adequate under any known linguistic point of view, so this footnote is entirely useless. Or has anything changed since 1969?

[20] `http://www.korpus.cz`

[21] For example, the approximation states that the probability of lemma *expensive* with formeme adj:attr given parent's lemma *speaker* is constant regardless of the formeme of *speaker* – it can be a subject (n:subj), direct object (n:obj) or prepositional object (n:for+X, n:to+X, n:with+X etc.). We are aware of cases, when this approximation is not suitable, e.g. if the governing formeme in our example was n:in_accordance_with+X.

[22] These cases are in Czech much less common than in English. For example, *stát* can be a noun or a verb. In PDT style, there are different lemmas for homonymous words – stát-1 and stát-2, but in TectoMT style, there are no such technical suffixes in lemmas.

Smoothing

Our module `LanguageModel::TreeLM` exploits the before-mentioned two files to predict smoothed probabilities of $P(Ld, Fd|Lg)$. The simplest solution would be to return maximum likelihood estimates (i.e. $C(Ld, Fd, Lg)/C(Lg)$ – both counts are saved in the first file) and for unseen triples return $1/C(Lg)$ or some other small number.

We have decided to return a linear combination of two models:

1. $P(Ld, Fd|Lg)$
2. $P(Fd|Lg) \cdot P(Ld|Fd, Pg)$

The second model is an approximation of the first one:
$P(Ld, Fd|Lg) = P(Fd|Lg) \cdot P(Ld|Fd, Lg) \approx P(Fd|Lg) \cdot P(Ld|Fd, Pg)$.

The weights are assigned according to the frequency of the governing lemma ($C(Lg)$) – the first model is preferred when enough data for a reliable estimate is available. In cases when the governing lemma is unseen ($C(Lg) = 0$), we use $P(Ld, Fd|Pg)$ as a fallback.

Future plans

Since reliable target-language tree model is essential for good performance of the HMTM transfer, we plan to improve it. One possible solution is to use maximum entropy model and train the best set of weights automatically.

Another solution is to use Generalized Parallel Backoff (Bilmes and Kirchhoff, 2003) technique. Our model is similar to Factored Language Models in the aspect that we have more factors and there is no obvious backoff order.[23] Generalized Parallel Backoff addresses this problem.

[23]Our factors are: lemma, formeme, PoS and semantic PoS. PoS can be determined from the lemma (because we have encoded it to the lemma in the training phase), semantic PoS is a part of the formeme. However, if we had a general tool that would select the optimal backoff path through a lattice of combinations of factors, we could integrate also other factors like grammatemes.

Chapter 7

Synthesis

Spíše bych byl kladivo než nehet.
TectoMT, 2009[1]

7.1 Modifications

All steps of the synthesis phase are described in (Žabokrtský et al., 2008). Here we only briefly discuss the blocks we have modified.

Block `Init_morphcat`

This block uses t-layer attributes (grammatemes, formeme, semantic PoS etc.) to fill morphological categories (structured attribute `morphcat` comprises PoS, fine-grained PoS, gender, number, case, possessive gender, possessive number, person, tense, grade, negation and voice).

Our modifications regard personal pronouns. For example, if a possessive personal pronoun has a coreference link to the subject of a clause, its morphological lemma should be *svůj* instead of *jeho*. However, there is an exception for cases when the pronoun has the nominative case: *Má štěstí jako *svůj otec.*

Block `Impose_subjpred_agr`

According to the subject-predicate agreement rule, attributes for morphological categories gender, number and person are copied from the subject node into the predicate node (i.e. the node with the finite verb).

We have added rules for a special treatment of agreement with copula constructions and coordinated subjects. Coordinated subjects usually need a plural verb (*Peter and Paul are...*) except for disjunctives (*Peter or Paul is...*).

Block `Resolve_verbs`

We have added rules for generation of infinitives and imperatives.

Block `Add_subord_clause_punct`

Commas (more precisely, a-nodes corresponding to commas) are added to boundaries of finite clauses. We have refined the rules for special cases such as quotations. We have also created a new block `Fill_clause_number` that coindexes all nodes belonging to the same finite clause.

[1] *I'd rather be a hammer than a nail.*

Block `Add_coord_punct`

Commas are added in front of conjunction *ale* and between multiple coordination members (*A, B, C and D*). The original implementation of this block should be almost equivalent to our new implementation if the sentences with coordinations had a correct structure. As we show in Section 3.3.1, parsing of coordinations in the analysis phase is problematic. For example, there are cases when *A and B C* is analyzed as if also the word *C* was a member of the coordination (but it is actually a modifier of *B*). In the original implementation such incorrectly parsed structures are translated with superfluous commas (*A a B, C*), while our implementation produces correctly *A a B C*.

Block `Generate_wordforms`

Word forms are generated according to lemmas and morphological categories (attributes `m_lemma` and `morphcat`). In theory, the word form should be fully specified by the lemma and morphological tag and there is a deterministic Czech word generator suited for the task (Hajič, 2004). In practice, the tags are "underspecified", because they are generated from the t-layer that was translated from English. Some categories are not known and must be guessed.

We have created a module `LanguageModel::MorphoLM`, which has a method for returning all forms of a given lemma whose tags match a given regular expression. The word forms are sorted according to their frequency. The model was trained on the corpus SYN (with 500 million words) of Czech National Corpus.[2]

Block `Vocalize_prepositions`

Prepositions *k, s, v* and *z* are vocalized (changed to *ke, se, ve, ze* and rarely also *ku*) according to the prefix of the following word. We have reduced the size of regular expressions involved (for speed-up) and added several missing cases. For example, numerals written in digits are treated as if they were written in words.

Block `Concatenate_tokens`

The resulting sentence is created by flattening the tree into a string. The only additional work in this block should be removing of spaces around punctuation. The original implementation contained also a heuristic rule for adding commas in front of verba dicendi that follow a direct speech. We have removed this rule, because

- With our new block for numbering of clauses and adding commas (as described above), there are not so many commas missing in the output.
- These errors should be anyway repaired already before the concatenation of tokens.

[2] `http://www.korpus.cz`

7.2 Evaluation

Modification	diff (BLEU)	diff (NIST)
original `Init_morphcat`	0.0005	0.0135
original `Impose_subjpred`	0.0002	0.0055
original `Resolve_verbs`	0.0002	0.0018
original `Add_subord_clause_punct`	0.0020	0.0125
original `Add_coord_punct`	0.0005	0.0106
original `Generate_wordforms`	0.0010	0.0263
original `Vocalize_prepositions`	0.0001	0.0014
original `Concatenate_tokens`	0.0001	-0.0002
all above together	0.0031	0.0621

Table 7.1: *Modifications of the synthesis*
For explanation see Section 4.5.

Improvements in the synthesis phase were not our priority, because our analysis of translation errors revealed that only 3% of the errors are caused by the synthesis phase (see Section 3.3.1).

Misplaced commas cause a reduction of BLEU score, but these errors are usually considered less serious by human judges. Hence we can conclude that the most important improvement in the synthesis phase is the introduction of the morphological model for better generation of word forms.

Note that modifications of the block `Concatenate_tokens` caused a negligible improvement in terms of BLEU score, but a negligible impairment in terms of NIST score.

Chapter 8

Other Improvements

Pták v ruce je cenný dvakrát v Bushovi.
TectoMT, 2009[1]

TectoMT internals

We have implemented several methods of the class `TectoMT::Node`. These methods greatly simplify creating of new blocks. For details see the documentation[2].

- `create_new_child()`
- `get_clause_root()` and `get_clause_nodes()`
- methods for changing word order ("shifting of nodes")
  and methods `precedes($node)`, `get_next_node()` and `get_prev_node()`
- Methods `get_children()`, `get_descendants()` and `get_siblings()` were enriched with options ordered, preceding_only, following_only, first_only, last_only, add_self.
- Methods for effective children and parents were rewritten in TectoMT style.

We have also added the method `process_bundle` to the class `TectoMT::Block`, so blocks can be written in a shorter way. We have implemented a support for including of scenario files into other scenario files to facilitate modular design.

Additional blocks and utilities

- `Eval::Bleu`
  fast Perl implementation of BLEU score evaluation
- `Print::MT_stats`
  A block that prints sentences (source, TectoMT translation and reference translation) with numbers of matching n-grams. This output is very useful for quick development of MT systems. After every change, we can compare the outputs of both versions. Sentences are sorted according to the differences in numbers of matching n-grams, so we can easily find sentences that were affected by the change (either positively or negatively).

[1] *A bird in the hand is worth two in the bush.*
[2] `http://ufal.mff.cuni.cz/tectomt/documentation.htm`

- refactoring of over 100 blocks
  Most of the refactoring were just casual updates to current TectoMT guidelines (inspired by Perl Best Practices (Conway, 2005)); some blocks were completely rewritten to improve the understandability of the code.

- support for TectoMT development
  We have adapted tools `perltidy`[3] (automatic reformatting of Perl source code) and `perlcritic`[4] (framework for applying coding standards to Perl source code) for use in TectoMT.

- script for collecting error markers
  Tables 3.4 – 3.10 were automatically generated by this script from files with manual error analysis.

Documentation

- small portions of text in TectoMT Developers Guidelines[5]

- 4 commented blocks for Jana Straková's TectoMT Tutorial[6]

[3] `http://perltidy.sourceforge.net/`
[4] `http://perlcritic.com/`
[5] `http://ufal.mff.cuni.cz/tectomt/guide/guidelines.html`
[6] `https://wiki.ufal.ms.mff.cuni.cz/external:tectomt:tutorial`

Chapter 9

Conclusion

Chléb je zaměstnanec života.
TectoMT, 2009[1]

In the presented text, we have described improvements of English-Czech translation system TectoMT. We have annotated 250 sentences produced by the baseline system and identified the most prominent errors and their sources. Subsequently, we have designed and implemented methods that repair some of these errors.

We have achieved an improvement over the baseline 0.0659 BLEU (3.9735 NIST). Our new version of TectoMT reaches 0.0981 BLEU (4.7157 NIST). Although these results are still lower than those of the state-of-the-art English-Czech MT systems, our system is rapidly evolving and we see a great potential for further improvements.

Unfortunately, we do not have any manual evalutation of translation quality of our new version yet, but it should be available after evaluating the WMT 2010 Shared Task.

	#errors		**diff (BLEU)**	
Analysis	422	30%	0.0078	28%
Transfer	954	67%	0.0171	61%
Synthesis	42	3%	0.0031	11%

Table 9.1: *Comparison of distribution of errors and effect of our modifications This table combines results from Table 3.4 (the left column) and Table 4.1 (the right column). Note that each of these two columns is measured on a different (mutually non-overlapping) set of sentences. For details see Sections 4.3 and 4.5.*

In Table 9.1 we summarize amount of errors found in the three main phases of the baseline translation (analysis, transfer and synthesis) and the effect of our modifications measured by the BLEU score. It is interesting that these values are proportional though it is a rather coincidence than our aim.

If we should choose just a single improvement (from those we have implemented) as the most important one, it would be the introduction of Hidden Markov Tree Models to the transfer phase.

We are looking forward to continue on improving TectoMT system. In future we would like to further improve the transfer phase by exploiting fully automatized techniques such as MERT (Och, 2003), perceptron or conditional random fields (Roark et al., 2004).

[1] *Bread is the staff of life.*

Bibliography

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. pages 308–319, 1990. 60

Jeff Bilmes and Katrin Kirchhoff. Factored Language Models and Generalized Parallel Backoff. In *Human Language Technology Conference/North American chapter of the Association for Computational Linguistics (HLT/NAACL-2003)*, Edmonton, Alberta, May/June 2003. 70, 73

I. Boguslavsky, S. Grigorieva, N. Grigoriev, L. Kreidlin, and N. Frid. Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 987–991. Association for Computational Linguistics Morristown, NJ, USA, 2000. 48

Igor Boguslavsky, Leonid Iomdin, and Victor Sizov. Multilinguality in ETAP-3: Reuse of Lexical Resources. In Gilles Sérasset, editor, *COLING 2004 Multilingual Linguistic Resources*, pages 1–8, Geneva, Switzerland, August 28 2004. COLING. 1

Ondřej Bojar, Miroslav Janíček, Zdeněk Žabokrtský, Pavel Češka, and Peter Beňa. CzEng 0.7: Parallel Corpus with Community-Supplied Translations. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. ELRA. 55

Ondřej Bojar. *Exploiting Linguistic Data in Machine Translation.* PhD thesis, ÚFAL, MFF UK, Prague, Czech Republic, October 2008. 3

Ondřej Bojar, David Mareček, Václav Novák, Martin Popel, Jan Ptáček, Jan Rouš, and Zdeněk Žabokrtský. English-Czech MT in 2008. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 125–129, Athens, Greece, March 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W09/W09-0x22`. 3, 18

Thorsten Brants. TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 224–231, 2000. 25

Peter E. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 1993. URL `http://acl.ldc.upenn.edu/J/J93/J93-2003.pdf`. 63

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Further Meta-Evaluation of Machine Translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106, Columbus,

Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W08/W08-0309`. 7

C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke, et al. Structure and performance of a dependency language model. In *Proceedings of Eurospeech*, 1997. 70, 71

Silvie Cinková, Jan Hajič, Marie Mikulová, Lucie Mladová, Anja Nedolužko, Petr Pajas, Jarmila Panevová, Jiří Semecký, Jana Šindlerová, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. Annotation of English on the tectogrammatical level. Technical Report 35, ÚFAL MFF UK, 2006. 47, 63

Martin Čmejrek. *Using Dependency Tree Structure for Czech-English Machine Translation*. PhD thesis, ÚFAL, MFF UK, Prague, Czech Republic, 2006. 3

Michael Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, 1999. 37

Damian Conway. *Perl Best Practices*. O'Reilly Media, Inc., 2005. 78

Matthew Crouse, Robert Nowak, and Richard Baraniuk. Wavelet-Based Statistical Signal Processing Using Hidden Markov Models. *IEEE Transactions on Signal Processing*, 46(4):886–902, 1998. 60

Jan Cuřín, Martin Čmejrek, Jiří Havelka, Jan Hajič, Vladislav Kuboň, and Zdeněk Žabokrtský. Prague Czech-English Dependency Treebank, Version 1.0. Linguistics Data Consortium, Catalog No.: LDC2004T25, 2004. 55

Michelangelo Diligenti, Paolo Frasconi, and Marco Gori. Hidden tree Markov models for document image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:2003, 2003. 60

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *HLT 2002: Human Language Technology Conference: proceedings of the second international conference on human language technology research*, pages 138–145, San Diego, California, March 2002. 7, 52

Jean-Baptiste Durand, Paulo Goncalvès, and Yann Guédon. Computational methods for hidden Markov tree models - An application to wavelet trees. *IEEE Transactions on Signal Processing*, 52(9):2551–2560, 2004. 60

Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 2*, pages 205–208, July 2003. 3

Jan Hajič, Martin Čmejrek, Jason Eisner, Gerald Penn, Owen Rambow, Drago Radev, Yuan Ding, Terry Koo, and Kristen Parton. Natural Language Generation in the Context of Machine Translation. Technical Report 1, CLSP JHU, USA, Baltimore, MD, 2002. 3

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, and Marie Mikulová. Prague Dependency Treebank 2.0. Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia, 2006. 3, 44

Jan Hajič, Silvie Cinková, Kristýna Čermáková, Lucie Mladová, Anja Nedolužko, Pajas Petr, Jiří Semecký, Jana Šindlerová, Josef Toman, Kristýna Tomšů, Matěj Korvas, Magdaléna Rysová, Kateřina Veselovská, and Zdeněk Žabokrtský. Prague English Dependency Treebank, Version 1.0, Jan 2009. 47

Eva Hajičová, Zdeněk Kirschner, and Petr Sgall. A Manual for Analytic Layer Annotation of the Prague Dependency Treebank (English translation). Technical report, ÚFAL MFF UK, Prague, Czech Republic, 1999. 3, 43

Jan Hajič. *Disambiguation of Rich Inflection – Computational Morphology of Czech*. Charles University – The Karolinum Press, Prague, 2004. 75

Barbora Vidová Hladká, Jan Hajič, Jiří Hana, Jaroslava Hlaváčová, Jiří Mírovský, and Jan Raab. The Czech Academic Corpus 2.0. Linguistic Data Consortium, LDC Catalog No.: LDC2008T22, Philadelphia, Oct 2008. 44

Petr Homola, Vladislav Kuboň, and Pavel Pecina. A Simple Automatic MT Evaluation Metric. In *Fourth Workshop on Statistical Machine Translation*, page 33, March 2009. 7

Mark Hopkins and Jonas Kuhn. Machine Translation as Tree Labeling. In *Proceedings of SSST, NAACL-HLT*, pages 41–48, 2007. 7

Václav Klimeš. Transformation-Based Tectogrammatical Dependency Analysis of English. In *Proceedings of the 10th International Conference on Text, Speech and Dialogue*, pages 15–22, 2007. 50

Philipp Koehn and Christof Monz. Manual and automatic evaluation of machine translation between European languages. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 102–121, 2006. 7

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-2045`. 2

Geoffrey Leech, Roger Garside, and Michael Bryant. CLAWS4: the tagging of the British National Corpus. In *Proceedings of the 15th conference on Computational linguistics*, pages 622–628, Morristown, NJ, USA, 1994. Association for Computational Linguistics. 29

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999. 34, 60

Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6, 2006. 39

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Langauge Technology Conference and Conference on Empirical Methods in Natural Language Processing (HTL/EMNLP)*, pages 523–530, Vancouver, BC, Canada, 2005. 37, 39

Arul Menezes and Stephen D. Richardson. A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. In *Proceedings of the workshop on Data-driven methods in machine translation*, volume 14, pages 1–8, 2001. 1

Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský. Annotation on the tectogrammatical level in the Prague Dependency Treebank. Annotation manual. Technical Report 30, ÚFAL MFF UK, Prague, Czech Rep., 2006. 4, 41

Guido Minnen, John Carroll, and Darren Pearce. Robust Applied Morphological Generation. In *Proceedings of the 1st International Natural Language Generation Conference*, pages 201–208, Israel, 2000. 28, 30

Václav Novák and Zdeněk Žabokrtský. Feature Engineering in Maximum Spanning Tree Dependency Parser. In *Proceedings of the 10th International Conference on Text, Speech and Dialogue*, pages 92–98, 2007. 37

Franz Josef Och. Minimum Error Rate Training in Statistical Machine Translation. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003. URL `http://www.aclweb.org/anthology/P03-1021.pdf`. 79

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL 2002, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, 2002. 7

Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: Syntactically informed phrasal smt. In *ACL*, pages 271–279, Ann Arbor, Michigan, 2005. 1

Adwait Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1996, pages 133–142, 1996. 25

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 47, Morristown, NJ, USA, 2004. Association for Computational Linguistics. 79

Beatrice Santorini. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). *Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Tech. Rep. MS-CIS-90-47, Line Lab*, 178, 1990. 25, 26, 29

Petr Sgall. *Generativní popis jazyka a česká deklinace*. Academia, Prague, Czech Republic, 1967. 1, 3

Libin Shen, Jinxi Xu, and Ralph Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. pages 577–585, 2008. 70, 71

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. Translating with non-contiguous phrases. In *Proceedings of HLT-EMNLP*, pages 755–762, 2005. 58

Marios Skounakis, Mark Craven, and Soumya Ray. Hierarchical Hidden Markov Models for Information Extraction. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 427–433. Morgan Kaufmann, 2003. 60

Drahomíra Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing, ACL 2007*, pages 67–74, Praha, 2007. 25, 44

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. Error Analysis of Machine Translation Output. In *Proceedings of the Fifth International Language Resources and Evaluation (LREC'06)*, pages 697–702, Genoa, Italy, May 2006. 7, 8

Wei Wang, Kevin Knight, and Daniel Marcu. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of EMNLP and CoNLL 2007*, pages 746–754, 2007. 1

Zdeněk Žabokrtský and Martin Popel. Hidden Markov Tree Model in Dependency-based Machine Translation. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, August 2009. in print. 60

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In *Proceedings of the 3rd Workshop on Statistical Machine Translation, ACL*, 2008. 3, 5, 54, 74

Dan Zeman, Jiří Hana, Hana Hanová, Jan Hajič, Barbora Hladká, and Emil Jeřábek. A Manual for Morphological Annotation, 2nd edition. Technical Report 27, ÚFAL MFF UK, Prague, Czech Republic, 2005. 3

Appendix A

List of abbreviations

afun	analytical function – an attribute of nodes on the a-layer, see page 43
AIM	aimed translation of a sentence (Czech), see page 11
a-layer	analytical layer of language description (in PDT)
a-node	a node at a-layer
BNC	British National Corpus
FGD	Functional Generative Description
HMTM	Hidden Markov Tree Model
MT	Machine Translation
NLP	Natural Language Processing
PDT	Prague Dependency Treebank
PEDT	Prague English Dependency Treebank
PennTB	Penn Treebank
PoS	Part of Speech
REF	reference translation of a sentence (Czech)
SRC	source sentence to be translated (English)
TST	sentence translated by TectoMT (Czech)
t-layer	tectogrammatical layer of language description (in PDT)
t-node	a node at t-layer
WMT	Workshop on Statistical Machine Translation

Appendix B

Content of the enclosed DVD

The enclosed DVD contains this thesis in PDF format (`thesis.pdf`) and three directories: `annotation_of_errors`, `experiments` and `tectomt`. For illustration we present here file listings (incomplete, of course):

Annotation of errors

resources for the manual analysis of translation errors presented in Chapter 3

```
annotation_of_errors
├── annotations.an          250 sentences annotated with error markers
├── an.xml                  syntax highlighting rules for our annotation format
└── Makefile                for automatic generation of summary tables (3.4 – 3.10)
```

Experiments

files needed to reproduce the BLEU/NIST results presented in this thesis

```
experiments
├── comparison              translated sentences sorted using Print::MT_stats as
│                           described on page 77
│   ├── c01_02_no_Fix_tokenization
│   ├── c01_03_orig_Fix_mtags
│   ├── ...
│   └── c01_52_all_2-51     comparison of baseline and our best scenario
├── input_raw_sgm
│   ├── our_test2009-src.cz     our test set in sgm format
│   └── our_test2009-src.en
├── old_blocks              original implementation of blocks that should by copied
│                           to tectomt/libs/blocks before evaluation
├── scenarios
│   ├── 01_best             our best scenario with all our modifications included
│   ├── 02_no_Fix_tokenization.scen
│   ├── 03_orig_Fix_mtags.scen
│   ├── ...
│   └── 52_all_2-51.scen    baseline scenario without our modifications
└── Makefile                for automatic evaluation using Grid computing[1]
```

[1] `http://ufal.mff.cuni.cz/tectomt/guide/guidelines.html#qrunblocks`

TectoMT

TectoMT framework is composed of a small versioned part and a large unversioned part. The unversioned part (called `share`) contains third party tools, models and other data. Due to space limitations only a portion (needed for English-Czech translation) of the data is included on the DVD.

```
tectomt                                          SVN revision number 2310
├── applications
│   └── translation
│       └── en2cs
│           ├── s_morce_mcd_hmtm.scen
│           └── Makefile
├── libs
│   ├── blocks
│   │   ├── SEnglishM_to_SEnglishA
│   │   │   ├── Fill_afun.pm
│   │   │   └── Fix_atree.pm
│   │   ├── SEnglishA_to_SEnglishT
│   │   │   ├── Detect_formeme.pm
│   │   │   └── Mark_edges_to_collapse.pm
│   │   ├── SEnglishT_to_TCzechT
│   │   │   └── Translate_LF_tree_Viterbi.pm
│   │   ├── TCzechT_to_TCzechA
│   │   │   └── Generate_wordforms.pm
│   │   └── TCzechA_to_TCzechW
│   │       └── Concatenate_tokens.pm
│   ├── core
│   │   └── TectoMT
│   │       ├── Document.pm
│   │       └── Node.pm
│   └── other
│       ├── EnglishMorpho
│       ├── Eval
│       │   ├── Bleu
│       │   └── Lemmatizer.pm
│       ├── LanguageModel
│       │   └── TreeLM.pm
│       └── TreeViterbi.pm
├── share                    a portion of the unversioned part of TectoMT
└── training
    └── language_model
```

Appendix C

Translation scenarios

C.1 Translation scenario used for WMT09

```
# ENGLISH_ANALYSIS:
SEnglishW_to_SEnglishM::Penn_style_tokenization
SEnglishW_to_SEnglishM::TagMorce
SEnglishW_to_SEnglishM::Fix_mtags
SEnglishW_to_SEnglishM::Lemmatize_mtree
SEnglishM_to_SEnglishA::McD_parser
     TMT_PARAM_MCD_EN_MODEL=conll_mcd_order2_0.01.model
SEnglishM_to_SEnglishA::Fix_McD_Tree
SEnglishP_to_SEnglishA::Fix_multiword_prep_and_conj
SEnglishA_to_SEnglishT::Mark_auxiliary_nodes
SEnglishA_to_SEnglishT::Mark_negator_as_aux
SEnglishA_to_SEnglishT::Build_ttree
SEnglishA_to_SEnglishT::Mark_named_entities
     TMT_PARAM_NER_EN_MODEL=ner-eng-ie.crf-3-all2008.ser.gz
SEnglishA_to_SEnglishT::Fill_is_member
SEnglishA_to_SEnglishT::Fix_tlemmas
SEnglishA_to_SEnglishT::Assign_coap_functors
SEnglishA_to_SEnglishT::Fix_is_member
SEnglishA_to_SEnglishT::Distrib_coord_aux
SEnglishA_to_SEnglishT::Mark_clause_heads
SEnglishA_to_SEnglishT::Mark_passives
SEnglishA_to_SEnglishT::Assign_functors
SEnglishA_to_SEnglishT::Mark_infin
SEnglishA_to_SEnglishT::Mark_relclause_heads
SEnglishA_to_SEnglishT::Mark_relclause_coref
SEnglishA_to_SEnglishT::Mark_dsp_root
SEnglishA_to_SEnglishT::Mark_parentheses
SEnglishA_to_SEnglishT::Recompute_deepord
SEnglishA_to_SEnglishT::Assign_nodetype
SEnglishA_to_SEnglishT::Assign_sempos
SEnglishA_to_SEnglishT::Assign_grammatemes
SEnglishA_to_SEnglishT::Detect_formeme
SEnglishA_to_SEnglishT::Detect_voice
SEnglishA_to_SEnglishT::Mark_person_names
```

```
# TRANSFER:
SEnglishT_to_TCzechT::Clone_ttree
SEnglishT_to_TCzechT::Baseline_formeme_translation
SEnglishT_to_TCzechT::Baseline_tlemma_translation
SEnglishT_to_TCzechT::Fix_transfer_choices
SEnglishT_to_TCzechT::Add_noun_gender
SEnglishT_to_TCzechT::Add_PersPron_below_vfin
SEnglishT_to_TCzechT::Add_verb_aspect
SEnglishT_to_TCzechT::Fix_date_time
SEnglishT_to_TCzechT::Fix_grammatemes_after_transfer
SEnglishT_to_TCzechT::Fix_negation
SEnglishT_to_TCzechT::Fix_verb_reflexivity
SEnglishT_to_TCzechT::Move_genitives_to_postposit
SEnglishT_to_TCzechT::Reverse_number_noun_dependency
SEnglishT_to_TCzechT::Move_dicendi_closer_to_dsp
SEnglishT_to_TCzechT::Override_pp_with_phrase_translation
SEnglishT_to_TCzechT::Recompute_deepord
SEnglishT_to_TCzechT::Find_gram_coref_for_refl_pron

# CZECH_SYNTHESIS:
TCzechT_to_TCzechA::Clone_atree
TCzechT_to_TCzechA::Init_morphcat
TCzechT_to_TCzechA::Impose_rel_pron_agr
TCzechT_to_TCzechA::Impose_subjpred_agr
TCzechT_to_TCzechA::Impose_attr_agr
TCzechT_to_TCzechA::Impose_compl_agr
TCzechT_to_TCzechA::Drop_subj_pers_prons
TCzechT_to_TCzechA::Add_prepositions
TCzechT_to_TCzechA::Add_subconjs
TCzechT_to_TCzechA::Add_reflex_particles
TCzechT_to_TCzechA::Add_auxverb_compound_passive
TCzechT_to_TCzechA::Add_auxverb_modal
TCzechT_to_TCzechA::Add_auxverb_compound_future
TCzechT_to_TCzechA::Add_auxverb_conditional
TCzechT_to_TCzechA::Add_auxverb_compound_past
TCzechT_to_TCzechA::Resolve_verbs
TCzechT_to_TCzechA::Clause_numbering
TCzechT_to_TCzechA::Move_clitics_to_wackernagel
TCzechT_to_TCzechA::Add_sent_final_punct
TCzechT_to_TCzechA::Add_subord_clause_punct
TCzechT_to_TCzechA::Add_coord_punct
TCzechT_to_TCzechA::Add_parentheses
TCzechT_to_TCzechA::Choose_mlemma_for_PersPron
TCzechT_to_TCzechA::Generate_wordforms
TCzechT_to_TCzechA::Recompute_ordering
TCzechT_to_TCzechA::Delete_superfluous_prepos
TCzechT_to_TCzechA::Vocalize_prepositions
TCzechT_to_TCzechA::Capitalize_named_entities
TCzechT_to_TCzechA::Capitalize_sent_start
TCzechA_to_TCzechW::Concatenate_tokens
```

C.2 New translation scenario

Description:

N	completely new block, my own work
•	major modifications of an original block (possibly including renaming)
○	minor modifications of an original block (possibly including renaming)
no mark	either an unchanged original block or only minor modifications not mentioned in this thesis

Important blocks where my attribution to the code is significant are printed in red. In the last column there is a number of the page in this thesis where the given block is described.

Analysis

○	SEnglishW_to_SEnglishM::Penn_style_tokenization	23
N	SEnglishW_to_SEnglishM::Fix_tokenization	23
	SEnglishW_to_SEnglishM::TagMorce	25
○	SEnglishW_to_SEnglishM::Fix_mtags	26
•	SEnglishW_to_SEnglishM::Lemmatize_mtree	30
•	SEnglishM_to_SEnglishA::McD_parser	38
	TMT_PARAM_MCD_EN_MODEL=conll_mcd_order2_0.01.model	
N	SEnglishM_to_SEnglishA::Fill_is_member_from_deprel	38
N	SEnglishM_to_SEnglishA::Fix_tags_after_parse	26
•	SEnglishM_to_SEnglishA::McD_parser REPARSE=1	40
	TMT_PARAM_MCD_EN_MODEL=conll_mcd_order2_0.01.model	
N	SEnglishM_to_SEnglishA::Fill_is_member_from_deprel	38
•	SEnglishM_to_SEnglishA::Fix_McD_topology	38
N	SEnglishM_to_SEnglishA::Fix_is_member	38
N	SEnglishM_to_SEnglishA::Fix_atree	39
○	SEnglishM_to_SEnglishA::Fix_multiword_prep_and_conj	40
N	SEnglishM_to_SEnglishA::Fill_afun_AuxCP_Coord	45
N	SEnglishM_to_SEnglishA::Fill_afun	45
N	SEnglishA_to_SEnglishT::Mark_edges_to_collapse	48
○	SEnglishA_to_SEnglishT::Mark_edges_to_collapse_neg	48
N	SxxA_to_SxxT::Build_ttree LANGUAGE=English	48
N	SxxA_to_SxxT::Fill_is_member LANGUAGE=English	
N	SxxA_to_SxxT::Move_aux_from_coord_to_members LANGUAGE=English	49
	SEnglishA_to_SEnglishT::Mark_named_entities	
	TMT_PARAM_NER_EN_MODEL=ner-eng-ie.crf-3-all2008.ser.gz	
	SEnglishA_to_SEnglishT::Fix_tlemmas	
○	SEnglishA_to_SEnglishT::Assign_coap_functors	
	SEnglishA_to_SEnglishT::Fix_is_member	
○	SEnglishA_to_SEnglishT::Mark_clause_heads	50
	SEnglishA_to_SEnglishT::Mark_passives	
	SEnglishA_to_SEnglishT::Assign_functors	
	SEnglishA_to_SEnglishT::Mark_infin	
○	SEnglishA_to_SEnglishT::Mark_relclause_heads	50
	SEnglishA_to_SEnglishT::Mark_relclause_coref	
	SEnglishA_to_SEnglishT::Mark_dsp_root	
	SEnglishA_to_SEnglishT::Mark_parentheses	

	SEnglishA_to_SEnglishT::Recompute_deepord	
	SEnglishA_to_SEnglishT::Assign_nodetype	
•	SEnglishA_to_SEnglishT::Assign_grammatemes	50
•	SEnglishA_to_SEnglishT::Detect_formeme	50
	SEnglishA_to_SEnglishT::Detect_voice	
	SEnglishA_to_SEnglishT::Mark_person_names	
	Transfer	
	SEnglishT_to_TCzechT::Clone_ttree	
N	SEnglishT_to_TCzechT::Translate_LF_phrases	65
N	SEnglishT_to_TCzechT::Translate_F_try_rules	55
N	SEnglishT_to_TCzechT::Translate_F_add_variants	55
N	SEnglishT_to_TCzechT::Translate_F_rerank	55
N	SEnglishT_to_TCzechT::Translate_L_try_rules	55
N	SEnglishT_to_TCzechT::Translate_L_add_variants	55
N	SEnglishT_to_TCzechT::Translate_LF_numerals_by_rules	55
N	SEnglishT_to_TCzechT::Translate_L_filter_aspect	55
N	SEnglishT_to_TCzechT::Cut_variants MAX_LEMMA_VARIANTS=6 MAX_FORMEME_VARIANTS=6	
N	SEnglishT_to_TCzechT::Rehang_to_eff_parents	64
N	SEnglishT_to_TCzechT::Translate_LF_tree_Viterbi2 LM_WEIGHT=0.4 FORMEME_WEIGHT=1 BACKWARD_WEIGHT=0.3	67
N	SEnglishT_to_TCzechT::Rehang_to_orig_parents	64
○	SEnglishT_to_TCzechT::Fix_transfer_choices	
○	SEnglishT_to_TCzechT::Add_noun_gender	
○	SEnglishT_to_TCzechT::Add_PersPron_below_vfin	
	SEnglishT_to_TCzechT::Add_verb_aspect	
○	SEnglishT_to_TCzechT::Fix_date_time	56
	SEnglishT_to_TCzechT::Fix_grammatemes_after_transfer	
○	SEnglishT_to_TCzechT::Fix_negation	
	SEnglishT_to_TCzechT::Move_genitives_to_postposit	
○	SEnglishT_to_TCzechT::Move_dicendi_closer_to_dsp	56
	SEnglishT_to_TCzechT::Recompute_deepord	
	SEnglishT_to_TCzechT::Find_gram_coref_for_refl_pron	
	Synthesis	
	TCzechT_to_TCzechA::Clone_atree	
•	TCzechT_to_TCzechA::Reverse_number_noun_dependency	56
○	TCzechT_to_TCzechA::Init_morphcat	74
	TCzechT_to_TCzechA::Impose_rel_pron_agr	
○	TCzechT_to_TCzechA::Impose_subjpred_agr	74
	TCzechT_to_TCzechA::Impose_attr_agr	
	TCzechT_to_TCzechA::Impose_compl_agr	
	TCzechT_to_TCzechA::Drop_subj_pers_prons	
○	TCzechT_to_TCzechA::Add_prepositions	
	TCzechT_to_TCzechA::Add_subconjs	
	TCzechT_to_TCzechA::Add_reflex_particles	
	TCzechT_to_TCzechA::Add_auxverb_compound_passive	
	TCzechT_to_TCzechA::Add_auxverb_modal	
	TCzechT_to_TCzechA::Add_auxverb_compound_future	

	TCzechT_to_TCzechA::Add_auxverb_conditional	
	TCzechT_to_TCzechA::Add_auxverb_compound_past	
○	TCzechT_to_TCzechA::Resolve_verbs	74
	TCzechT_to_TCzechA::Fill_clause_number	
	TCzechT_to_TCzechA::Move_clitics_to_wackernagel	
○	TCzechT_to_TCzechA::Add_sent_final_punct	
○	TCzechT_to_TCzechA::Add_subord_clause_punct	74
○	TCzechT_to_TCzechA::Add_coord_punct	75
	TCzechT_to_TCzechA::Add_parentheses	
○	TCzechT_to_TCzechA::Choose_mlemma_for_PersPron	
●	TCzechT_to_TCzechA::Generate_wordforms	75
	TCzechT_to_TCzechA::Recompute_ordering	
	TCzechT_to_TCzechA::Delete_superfluous_prepos	
○	TCzechT_to_TCzechA::Vocalize_prepositions	75
○	TCzechT_to_TCzechA::Capitalize_sent_start	
○	TCzechT_to_TCzechA::Capitalize_named_entities	
○	TCzechA_to_TCzechW::Concatenate_tokens	75

Appendix D

Sample of annotated translation errors

In the following annotations, `l` is a shorthand for `lex` (incorrect lemma), `f` is a shorthand for `form` (incorrect form) and `lf` is a shorthand for `lex::form` (incorrect lemma and formeme). Other markers are used as described in Chapter 3.

```
ID  ihned.cz-2008-09-30-36776---1.0 (newstest2009-src.sgm-001.tmt 1)
SRC Prague Stock Market falls to minus by the end of the trading day
REF Pražská burza se ke konci obchodování propadla do minusu
TST Pražský l::zásobní trh l0::spadá f-tagger::minus koncem lf0::obchodování dne.

ID  ihned.cz-2008-09-30-36776---2.0 (newstest2009-src.sgm-001.tmt 2)
SRC After a sharp drop in the morning, the Prague Stock Market corrected its losses.
REF Po ranním prudkém propadu pražská burza korigovala ztráty.
TST Po l0::ostré l::kapce lf0::ráno Pražský l::zásobní trh l0::opravil gram-svuj-tecto::jeho
ztráty.

ID  ihned.cz-2008-09-30-36776---3.0 (newstest2009-src.sgm-001.tmt 3)
SRC Transactions with stocks from the Czech Energy Enterprise (ČEZ) reached nearly half of
the regular daily trading.
REF Transakce s akciemi ČEZ dosáhly téměř poloviny běžného denního obchodu.
TST Transakce s akciemi z České energetické organizace z punct-brack::ČEZ dosáhly téměř
poloviny pravidelného denního obchodování.

ID  ihned.cz-2008-09-30-36776---4.0 (newstest2009-src.sgm-001.tmt 4)
SRC The Prague Stock Market immediately continued its fall from Monday at the beginning of
Tuesday's trading, when it dropped by nearly six percent.
REF Pražská burza navázala hned od počátku úterního obchodování na svůj pondělní propad, když
klesala o téměř šest procent.
TST Pražský l::zásobní trh okamžitě pokračoval v gram-svuj-tecto::jeho pádu z pondělí na
f::začátek obchodování lf::úterý, kdy l::gram-gender-tecto::odhodilo téměř šest f-
num::procenty.

ID  ihned.cz-2008-09-30-36776---5.0 (newstest2009-src.sgm-001.tmt 5)
SRC This time the fall in stocks on Wall Street is responsible for the drop.
REF Tentokrát za poklesem stojí propad akcií na Wall Street.
TST Tento phrase-x::čas lf::podzimní do f::akcií na l-neT::Ulici Wall order::je odpovědný za
l::kapku.

ID  ihned.cz-2008-09-30-36776---6.0 (newstest2009-src.sgm-001.tmt 6)
SRC The reaction of the market to the results of the vote in the American House of
Representatives, which refused to support the plan for the stabilization of the financial
sector there, has manifested itself here as well.
REF I u nás se tak projevuje reakce trhu na výsledek hlasování americké Sněmovny
reprezentantů, která odmítla podpořit plán stabilizace tamního finančního sektoru.
TST Reakce trhu na výsledky l::hlasu v Americkém l-ne::domě zástupců, který odmítl podporovat
```

ID ihned.cz-2008-09-30-36776---9.0 (newstest2009-src.sgm-001.tmt 9)
SRC The Hang Seng Index of the Hong Kong Stock Exchange wrote off nearly four percent during the day, but later it erased a part of the losses and reduced the decrease to roughly 2.5 percent.
REF Index Hang Seng burzy v Hongkongu si v průběhu obchodování odepsal téměř čtyři procenta, později ale část ztrát smazal a pokles zredukoval na zhruba 2,5 procent a.
TST **l-neT-tecto::**Vydržet index šengů Hongkongské **extra-x-ne::**čínské **l::**zásobní **l::**výměny **gram-gender::**odepsala téměř čtyři **f-num::**procentům během dne, ale později **extra-parser::**to, **gram-gender-parser::**vymazalo část ztrát a **gram-gender-parser-coord::**snížily pokles na zhruba 2,5 procenta.

ID ihned.cz-2008-09-30-36776---10.0 (newstest2009-src.sgm-001.tmt 10)
SRC The Hang Seng China Enterprises Index, which follows the movement of Chinese stocks on the stock market in Hong Kong, dropped by 3.8 percent, in Shanghai the markets were closed.
REF Index Hang Seng China Enterprises, který sleduje pohyb čínských akcií na burze v Hongkongu, klesl o 3,8 procenta, v Šanghaji byly trhy uzavřeny.
TST **l-neT::**Vydržet čínský Enterprises index šengů, který **l0::**následuje pohyb čínských akcií na trhu **gram-numberU::**akcie v Hongkongském **l-x-ne::**Kongu, **l::**odhodil 3,8 **f-num::**procenty, že ve **l0-ne::**Shanghaii trhy **order::**byly **f-ze-parser-coord::**zavřeny.

ID ihned.cz-2008-09-30-36776---11.0 (newstest2009-src.sgm-001.tmt 11)
SRC Stocks on the market in Sydney lost more than five percent, but ultimately lowered their losses to 4.3 percent.
REF Akcie na burze v Sydney ztrácely více než pět procent, ale nakonec ztráty snížily na 4,3 procent a.
TST Akcie na trhu **f-parser::**Sydney ztratily více než pět procent, ale nakonec **gram-tense-parser-coord::**sníží **gram-svuj-parser-coord::**jejich ztráty 4,3 **f-num::**procentům.

ID ihned.cz-2008-09-30-36776---12.0 (newstest2009-src.sgm-001.tmt 12)
SRC The stock exchange in Taiwan dropped by 3.6 percent according to the local index.
REF Burza na Tchaj-wanu podle tamního indexu klesla o 3,6 procent a.
TST **l::**Výměna **gram-numberU::**akcie ve **l-ne::**Wan **l::**odhodila 3,6 **f-num::**procenty podle **gram-gender-syn::**místní indexu.

ID ihned.cz-2008-09-30-36776---13.0 (newstest2009-src.sgm-001.tmt 13)
SRC "The timing of the bailout action in the USA is uncertain and it will influence financial markets all over the world," remarked the head of the Hong Kong Currency Board, Joseph Yam.
REF "Načasování záchranné akce v USA je nejisté a ovlivní to finanční trhy v celém světě," poznamenal šéf hongkongského měnového úřadu Joseph Yam.
TST „Načasování **l-unk::**bailout akce v USA je **l::**neurčité a že **f-ze::**ovlivní finanční trhy na **extra-x::**všechno po celém světě,“ poznamenala **l0::**hlava Hongkongské **extra-x-ne::**čínské měnové rady Joseph Yam.

ID ihned.cz-2008-09-30-36776---14.0 (newstest2009-src.sgm-001.tmt 14)
SRC Despite the fact that it is a part of China, Hong Kong determines its currency policy separately, that is, without being dependent on the Chinese Central Bank.
REF Hongkong, přestože je součástí Číny, si měnovou politiku určuje samostatně, tedy bez závislosti na čínské centrální bance.
TST Navzdory skutečnosti, že **extra0-tecto::**to je **form0::**část Číny, Hongkongské **l-x-ne::**Kongo určuje svou **lf::**měnu **l::**zásadu odděleně, zda to **f-parser::punct::**je, bez **extra-x::**být **lf-x::**závislé na Čínské centrální bance.

ID ihned.cz-2008-09-30-36776---15.0 (newstest2009-src.sgm-001.tmt 15)
SRC Hong Kong has interest rates at the same level as the United States.
REF Úrokové sazby má Hongkong na stejné úrovni jako Spojené státy.
TST Hongkongské **l-x-ne::**Kongo má úrokové sazby na stejnou **f::**úroveň jako Spojené státy.

ID ihned.cz-2008-09-30-36776---16.0 (newstest2009-src.sgm-001.tmt 16)
SRC American legislators should quickly return to their negotiations and approve the bill to support the financial system, according to Australian Prime Minister Kevin Rudd.
REF Američtí zákonodárci by se podle australského premiéra Kevina Rudda měli urychleně vrátit k jednání a návrh podpory finančního systému schválit.
TST Američtí zákonodárci **gram-mod-parser-coord::**má rychle vrátit do **gram0-svuj-parser::**jejich **f::**jednání a schválit **l-neX::**Billu **lf::**podporovat finanční systém **lf-tecto-ne::**Australana premiéra **extra-x::**ministra **f0-syn-ne::**Kevin **f-parser-coord::**Rudd.

ID ihned.cz-2008-09-30-36776---17.0 (newstest2009-src.sgm-001.tmt 17)
SRC Otherwise there reputedly looms the threat that other countries will also feel the impacts.
REF Jinak prý hrozí, že dopady budou cítit i další země.
TST Jinak se **l::**nádeník rýsuje **f::**hrozbu, že se jiné země také budou **l-se::**cítit dopady.

ID ihned.cz-2008-09-30-36776---18.0 (newstest2009-src.sgm-001.tmt 18)
SRC American stock bloodbath
REF Krvavá lázeň amerických akcií
TST Americký **l::**zásobní masakr.

ID ihned.cz-2008-09-30-36776---19.0 (newstest2009-src.sgm-001.tmt 19)
SRC On Monday the American House of Representatives rejected the plan to support the financial system, into which up to 700 billion dollars (nearly 12 billion Czech crowns) was to be invested.
REF Americká Sněmovna reprezentantů v pondělí odmítla plán podpory finančního systému, na který mělo být vynaloženo až 700 miliard dolarů (téměř 12 bilionů Kč).
TST V pondělí Americký **l-ne::**dům zástupců odmítl plán podporovat finanční systém, na který 700 **form0-?-num::**miliardy dolarů **punct1-brack-parser-coord::**téměř 12 **form0-?-num::**miliardy českých **lf-parser-coord::**korunních **gram-mod-tecto::**bylo, že je **order::f-ze-tecto::**investováno.

ID ihned.cz-2008-09-30-36776---20.0 (newstest2009-src.sgm-001.tmt 20)
SRC The legislators thus ignored President George Bush's appeal for them to support the plan.
REF Zákonodárci tak ignorovali výzvu prezidenta George Bushe, aby plán podpořili.
TST Zákonodárci tak ignorovali **l::**odvolání prezidenta George Bushe **f0::**podporovat plán.

ID ihned.cz-2008-09-30-36776---21.0 (newstest2009-src.sgm-001.tmt 21)
SRC According to Bush, the plan would tackle the basic causes of the financial crisis and help stabilize the entire economy.
REF Podle Bushe by plán řešil základní příčiny finanční krize a pomohl by stabilizovat celou ekonomiku.
TST Podle Bushe plán **gram-mod-parser::**je **lf-parser::**řešení základních příčin finanční krize a **lf-parser::**pomoci stabilizovat celou ekonomiku.

ID ihned.cz-2008-09-30-36776---22.0 (newstest2009-src.sgm-001.tmt 22)
SRC American stocks suffered a bloodbath on Monday and the major stock indexes registered their greatest fall in more than 20 years.
REF Americké akcie v pondělí zažily krvavou lázeň a hlavní akciové indexy zaznamenaly největší pád za více než 20 let.
TST Americké akcie **l0::**trpěly masakr v pondělí, že a **f-ze-parser-coord::**hlavního indexy **gram-numberU::**akcie **f-parser-coord::**zaregistrovaly **gram-svuj-parser-coord::**jejich největší pád ve **f-?::**více 20 **f::**letech.

Appendix E

Sample of translated text

Překlad počítače (MT) dohání více a více význam v současném světě. Je mnoho přístupů k MT, který tradicí je zařazována do dvou paradigm: rule-bases a statistických. Klasické rule-based MT systémy jsou využití jazykové znalosti (pravidla gramatik slovníků napsala lidští znalci), ale používají, žádná informace nedozvěděla se automaticky od corpora. Překlad obvykle zahrnuje tři fáze: a přenosu analýzy syntézy. MT systémy další mohou být klasifikovány podle úrovně abstrakce jazyka, používá pro přenos - některé systémy provádějí povrchní analýzu, některý provádí hlubokou (nebo bohatou) analýzu. Výhoda hlubší analýzy je, že přenos měl by být snadnější a kdy vybuduje systém pro překlad více než dva jazyků, analýza a syntéza mohou být sdíleny všechny párky jazyka s, dají, zdrojem nebo cílovým jazykem respektive.
Klasické statistické MT systémy nejsou použití large-scale human-translated paralelní corpora a vícejazyčného corpora, ale téměř žádnou jazykovou znalost. Toto má výhodu, že stejný systém může být použit pro dvojici anu jazyků, pro které jsou dostatečné údaje o vzdělávání dostupné.
V posledních letech je tendence využít jazykovou znalost k většímu rozšíření, aby zlepšilo výkon statistických MT systémů. Na druhé straně rule-based nebo syntax-based MT systémy zahrnují statističtější metody (pravidla automaticky mohou se být naučena od paralelní corpora, stochastické značkovače a analyzátory jsou používány etc.). Tyto výsledky konvergence obou paradigm - zdají se, že moderní high-quality MT systémy budou používat statistické metody stejně well as jazykové znalost, současná rivalita mezi statistickým MT a rule-based MT bude se nedůležitá.

Tato teze vystihuje zlepšení English-Czech systému převodu, říká TectoMT. TectoMT je jeden ze slibných MT systémů které kombinovat statistické metody a jazykovou znalost. Zaměřuje se na přenos na so-called tectogrammatical vrstvu, která je vrstva hlubokých syntaktických stromů závislosti.
Nyní to je experimentální systém, který je překonán state-of-the-art MT systémy Google překladu nebo otevřených zdrojových Mojžíšů. Na druhé ruce má možnost vyřešit problémy s překladem společné n-gram, založí, systémům a navíc celý proces převodu je přiměřený také jazyková věc názoru.

The source text can be found on page 1.