Treex – an open-source framework for natural language processing*

Zdeněk Žabokrtský

Charles University in Prague, Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 118 00 Prague, Czech Republic
zabokrtsky@ufal.mff.cuni.cz
WWW home page: http://ufal.mff.cuni.cz/~zabokrtsky

Abstract. *The present paper describes Treex (formerly TectoMT), a multi-purpose open-source framework for developing Natural Language Processing applications. It facilitates the development by exploiting a wide range of software modules already integrated in Treex, such as tools for sentence segmentation, tokenization, morphological analysis, part-of-speech tagging, shallow and deep syntax parsing, named entity recognition, anaphora resolution, sentence synthesis, word-level alignment of parallel corpora, and other tasks. The most elaborate application of Treex is an English-Czech machine translation system with transfer on deep syntactic (tectogrammatical) layer. Besides research, Treex is used for teaching purposes and helps students to implement morphological and syntactic analyzers of foreign languages in a very short time.*

1 Introduction

Natural Language Processing (NLP) is a multidisciplinary field combining computer science, mathematics and linguistics, whose main aim is to allow computers to work with information expressed in human (natural) language.

The history of NLP goes back to 1950s. Early NLP systems were based on hand-written rules founded by linguistic intuitions. However, roughly two decades ago the growing availability of language data (especially textual corpora) and increasing capabilities of computer systems lead to a revolution in NLP: the field became dominated by data-driven approaches, often based on probabilistic modeling and machine learning.

In such data-driven scenario, the role of human experts was moved from designing rules rather to (i) preparing training data enriched with linguistically relevant information (usually by manual annotation), (ii) choice of an adequate probabilistic model, proposing features (various indicators potentially useful for making the desired predictions), and (iii) specifying an objective (evaluation) function. Optimization of the decision process (such as searching for optimal feature weights and other model parameters) is then entirely left to the learning algorithm.

Recent developments in NLP show that another paradigm shift might be approaching with unsupervised and semi-supervised algorithms, which are able to learn from data without hand-made annotations. However, such algorithms require considerably more complex models and for most NLP tasks they have not outperformed supervised solutions based on hand-annotated data so far.

Nowadays, researched NLP tasks range from relatively simple ones (like sentence segmentation, language identification), through tasks which already need a higher level of abstraction (such as morphological analysis, part-of-speech tagging, parsing, named entity recognition, coreference resolution, word sense disambiguation, sentiment analysis, natural language generation), to highly complex systems (machine translation, automatic summarization, or question answering). The importance of (and demand for) such tasks increases along with the rapidly growing amount of textual information available on the Internet.

Many NLP applications exploit several NLP modules chained in a pipeline (such as a sentence segmenter and part-of-speech tagger prior to a parser). However, if state-of-the-art solutions created by different authors – often written in different programming languages, with different interfaces, using different data formats and encodings – are to be used, a significant effort must be invested into integrating the tools. Even if these issues are only of technical nature, in real research they constitute one of limiting factors for building more complex NLP applications.

We try to eliminate such problems by introducing a common NLP framework that integrates a number of NLP tools and provides them with unified object-oriented interfaces, which hide the technical issues from the developer of a larger application. The framework's architecture seems viable – tens of researchers and students have already contributed to the system and the framework has been already used for a number of research tasks carried out at the Institute of For-

* The presented research is supported by the grants MSM0021620838 and by the European Commission's 7FP grant agreement n° 231720 (EuroMatrix Plus). We would like to thank Martin Popel for useful comments on the paper.

mal and Applied linguistics as well as at some other research institutions. The most complex application implemented within the framework is English-Czech machine translation. The framework is called Treex.[1]

The remainder of the paper is structured as follows. Section 2 overviews related work that had to be taken into account when developing such framework. Section 3 presents the main design decisions Treex is build on. English-Czech machine translation implemented in Treex is described in Section 4, while other Treex applications are mentioned in Section 5, which also concludes.

2 Related work

2.1 Theoretical background

Natural language is an immensely complicated phenomenon. Modeling the language in its entirety would be extremely complex, therefore its description is often decomposed into several subsequent layers (levels). There is no broadly accepted consensus on details concerning the individual levels, however, the layers typically roughly correspond to the following scale: phonetics, phonology, morphology, syntax, semantics, and pragmatics.

One of such stratificational hypotheses is Functional Generative Description (FGD), developed by Petr Sgall and his colleagues in Prague since the 1960s [18]. FGD was used with certain modifications as the theoretical framework underlying the Prague Dependency Treebank [6], which is a manually annotated corpus of Czech newspaper texts from the 1990s. PDT in version 2.0 (PDT 2.0) adds three layers of linguistic annotation to the original texts:

1. **morphological layer (m-layer)**
   Each sentence is tokenized and each token is annotated with a lemma (basic word form, such as nominative singular for nouns) and morphological tag (describing morphological categories such as part of speech, number, and tense).
2. **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-nodes). 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).
3. **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, functor (which represents the semantic value of syntactic dependency relation) and a set of grammatemes (e.g. tense, number, verb modality, deontic modality, negation).
   Edges in t-trees represent linguistic dependencies except for several special cases, the most notable of which are paratactic structures (coordinations).

All three layers of annotation are described in annotation manuals distributed with PDT 2.0.

This annotation scheme has been adopted and further modified in Treex. One of the modifications consists in merging m-layer and a-layer sentence representations into a single data structure.[2]

Treex also profits from the technology developed during the PDT project, especially from the existence of the highly customizable tree editor TrEd, which is used as the main visualization tool in Treex, and from the XML-based file format PML (Prague Markup Language, [14]), which is used as the main data format in Treex.

2.2 Other NLP frameworks

Treex is not the only existing general NLP framework. We are aware of the following other frameworks (a more detailed comparison can be found in [15]):

- ETAP-3 [1] is a C/C++ closed-source NLP framework for English-Russian and Russian-English translation, developed in the Russian Academy of Sciences.

[1] The framework was originally called TectoMT since starting its development in autumn 2005 [23], because one of the sources of motivation for building the framework was developing a Machine translation (MT) system using tectogrammatical (deep-syntactic) sentence representation as the transfer medium. However, MT is by far not the only application of the framework. As the name seemed to be rather discouraging for those NLP developers whose research interests did not overlap with tectogrammatics nor with MT, TectoMT was rebranded to Treex in spring 2011. To avoid confusion, the name Treex is used throughout the whole text even if it refers to a more distant history.

[2] As mentioned above, their units are in a one-to-one relation anyway; merging the two structures together has led to a significant reduction of time and memory requirements when processing large data, as well as to a lower burden for eyes when browsing the structures.

- GATE (Java, LGPL) is one of the most widely used NLP frameworks with integrated graphical user interface. It is being developed at University of Sheffield [4].
- Apache OpenNLP (Java, LGPL)[3] is an organizational center for open source NLP projects.
- WebLicht[4] is a Service Oriented Architecture for building annotated German text corpora.
- Apertium [20] is a free/open-source machine translation platform with shallow transfer.

In our opinion, none of these frameworks seems feasible (or mature enough) for experiments on MT based on deep-syntactic dependency transfer. The only exception is ETAP-3, whose theoretical assumptions are similar to that of Treex (its dependency-based stratificational background theory called Meaning-Text Theory [13] bears several resemblances to FGD), however, it is not an open-source project.

2.3 Contemporary machine translation

MT is a notoriously hard problem and it is studied by a broad research field nowadays: every year there are several conferences, workshops and tutorials dedicated to it (or even to its subfields). It goes beyond the scope of this work even to mention all the contemporary approaches to MT, but several elaborate surveys of current approaches to MT are already available to the reader elsewhere, e.g. in [10].

A distinction is usually made between two MT paradigms: rule-based MT (RBMT) and statistical MT (SMT). The rule-based MT systems are dependent on the availability of linguistic knowledge (such as grammar rules and dictionaries), whereas statistical MT systems require human-translated parallel text, from which they extract the translation knowledge automatically. One of the representatives of the first group is the already mention system ETAP-3.

Nowadays, the most popular representatives of the second group are phrase-based systems (in which the term 'phrase' stands simply for a sequence of words, not necessarily corresponding to phrases in constituent syntax), e.g. [8], derived from the IBM models [3].

Even if phrase-based systems have more or less dominated the field in the recent years, their translation quality is still far from perfect. Therefore we believe it makes sense to investigate also alternative approaches.

MT implemented in Treex lies somewhere between the two main paradigms. Like in RBMS, sentence representations used in Treex are linguistically interpretable. However, the most important decisions during the translation process are made by statistical models like in SMT, not by rules.

3 Treex architecture overview

3.1 Basic design decisions

The architecture of Treex is based on the following decisions:

- Treex is primarily developed in Linux. However, platform independent solutions are searched for wherever possible.
- The main programming language of Treex is Perl. However, a number of tools written in other languages have been integrated into Treex (after providing them with a Perl wrapper).
- Linguistic interpretability – data structures representing natural language sentences in Treex must be understandable by a human (so that e.g. translation errors can be traced back to their source). Comfortable visualization of the data structures is supported.
- Modularity – NLP tools in Treex are designed so that they are easily reusable for various tasks (not only for MT),
- Rules-vs-statistics neutrality – Treex architecture is neutral with respect to the rules vs. statistics opposition (rule-based as well as statistical solutions are combined).
- Massive data – Treex must be capable of processing large data (such as millions of sentence pairs in parallel corpora), which implies that distributed processing must be supported.
- Language universality – ideally, Treex should be easily extendable to any natural language.
- Data interchange support – XML is used as the main storage format in Treex, but Treex must be able to work with a number of other data formats used in NLP.

3.2 Data structure units

In Treex, representations of a text in a natural language is structured as follows:

- *Document.* A Treex document is the smallest independently storable unit. A document represents a piece of text (or several parallel pieces of texts in the case of multilingual data) and its linguistic representations. A document contains an ordered sequence of bundles.
- *Bundle.* A bundle corresponds to a sentence (or a tuple of sentences in the case of parallel data) and its linguistic representations. A bundle contains a set of zones.

[3] http://opennlp.sourceforge.net
[4] http://weblicht.sfs.uni-tuebingen.de/englisch/index.shtml

– *Zone.* Each language (languages are distinguished using ISO 639-2 codes in Treex) can have one or more zones in a bundle.[5] Each zone corresponds to one particular sentence and at most one tree for each layer of linguistic description.
– *Tree.* All sentence representations in Treex have the shape of an oriented tree.[6] At this moment there are four types of trees: (1) a-trees – morphology and surface-dependency (analytical) trees, (2) t-trees – tectogrammatical trees, (3) p-trees – phrase-structure (constituency) trees, (4) n-trees – trees of named entities.
– *Node.* Each nodes contains (is labeled by) a set of attributes (name-value pairs).
– *Attribute.* Some node attributes are universal (such as identifier), but most of them are specific for a certain layer. The set of attribute names and their values for a node on a particular layer is declared using the Treex PML schema.[7] Attribute values can be further structured.

Of course, there are also many other types of data structures used by individual integrated modules (such as dictionary lists, weight vectors and other trained parameters, etc.), but they are usually hidden behind module interfaces and no uniform structure is required for them.

3.3 Processing units

There are two basic levels of processing units in Treex:

– *Block.* Blocks are the smallest processing units independently applicable on a document.
– *Scenario.* Scenarios are sequences of blocks. When a scenario is applied on a document, the blocks from the sequence are applied on the document one after another.

(a) Simple Treex scenario:

```
Util::SetGlobal language=en # do everyth. in English zone
Block::Read::Text    # read a text from STDIN
W2A::Segment         # segment it into sentences
W2A::Tokenize        # divide sentences into words
W2A::EN::TagMorce    # morphological tagging
W2A::EN::Lemmatiz    # lemmatization (basic word forms)
W2A::EN::ParseMST    # dependency parsing
W2A::EN::SetAfunAuxCPCoord # fill analytical functions
W2A::EN::SetAfun     # fill analytical functions
Write::CoNLLX        # print trees in CoNLLX format
Write::Treex         # store trees into XML file
```

(b) Input text example:

```
When the prince mentions the rose, the geographer explains
that he does not record roses, calling them "ephemeral".
The prince is shocked and hurt by this revelation. The
geographer recommends that he visit the Earth.
```

(c) Fragment from the printed output (simplified):

```
1   The         the         DT   2
2   prince      prince      NN   3
3   is          be          VBZ  0
4   shocked     shock       VBN  5
5   and         and         CC   3
6   hurt        hurt        VBN  5
7   by          by          IN   5
8   this        this        DT   9
9   revelation  revelation  NN   7
10  .           .           .    3
```

(d) A-tree visualization in TrEd:

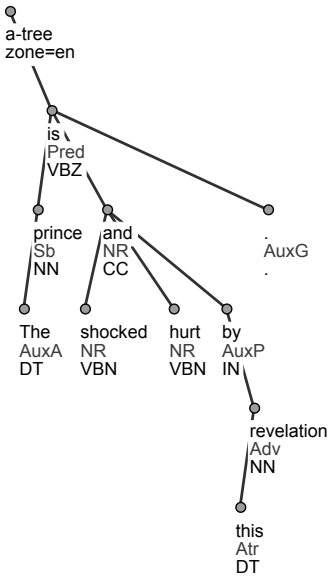

Fig. 1. Simple scenario for morphological and surface-syntactic analysis of English texts. Generated trees are printed in the CoNLLX format, which is a simple line-oriented format for representing dependency trees.

[5] Having more zones per language is useful e.g. for comparing machine translation with reference translation, or translation outputs from several systems. Moreover it highly simplifies processing of parallel corpora, or comparisons of alternative implementations of a certain tasks (such as different dependency parsers).
[6] However, tree-crossing edges such as anaphora links in a dependency tree can be represented too (as node attributes).
[7] There are also "wild" attributes allowed, which can store any Perl data structure without its prior declaration by PML. However, such undeclared attributes should serve only for tentative or rapid development purposes, as they cannot be validated.

A block can change a document's content "in place"[8] via a predefined object-oriented interface. One can distinguish several broad categories of blocks:

- blocks for sentence analysis – blocks for tokenization, morphological tagging, parsing, anaphora resolution, etc.
- blocks for sentence synthesis – blocks for propagating agreement categories, ordering words, inflecting word forms, adding punctuation, etc.
- blocks for transfer – blocks for translating a component of a linguistic representation from one language to another, etc.
- blocks for parallel texts – blocks for word alignment, etc.
- writer and reader blocks – block for storing/loading Treex documents into/from files or other streams (in the PML or other format),[9]
- auxiliary blocks – blocks for testing, printing, etc.

If possible, we try to implement blocks in a language independent way. However, many blocks will remain language specific (for instance a block for moving clitics in Czech clauses can hardly be reused for any other language).

There are large differences in complexity of blocks. Some blocks contain just a few simple rules (such as regular expressions for sentence segmentation), while other blocks are Perl wrappers for quite complex probabilistic models resulting from several years of research (such as blocks for parsing).

As for block granularity, there are no widely agreed conventions for decomposing large NLP applications.[10] We only follow general recommendations for system modularization. A piece of functionality should be performed by a separate block if it has well defined input and output states of Treex data structures, if it can be reused in more applications and/or it can be (at least potentially) replaced by some other solution.

[8] Pipeline processing (like with Unix text-processing commands) is not feasible here since linguistic data are deeply structured and the price for serializing the data at each boundary would be high.

[9] In the former versions, format converters were considered as tools separated from scenarios. However, providing the converters with the uniform block interface allows to read/write data directly within a scenario, which is not only more elegant, but also more efficient (intermediate serialization and storage can be skipped).

[10] For instance, some taggers provides both morphological tag and lemma for each word form, while other taggers must be followed by a subsequent lemmatizer in order to achieve the same functionality.

4 English-Czech machine translation in Treex

The translation scenario implemented in Treex composes of three steps described in the following sections: (1) analysis of the input sentences up to tectogrammatical layer of abstraction, (2) transfer of the abstract representation to the target language, and (3) synthesis (generating) of sentences in the target language. See an example in Figure 2.

4.1 Analysis

The analysis step can be decomposed into three phases corresponding to morphological, analytical and tectogrammatical analysis.

In the morphological phase, a text to be translated is segmented into sentences and each sentence is tokenized (segmented into words and punctuation marks). Tokens are tagged with part of speech and other morphological categories by the Morce tagger [19], and lemmatized.

In the analytical phase, each sentence is parsed using the dependency parser [12] based on Maximum Spanning Tree algorithm, which results in an analytical tree for each sentence. Tree nodes are labeled with analytical functions (such as Sb for subject, Pred for predicate, and Adv for adverbial).

Then the analytical trees are converted to the tectogrammatical trees. Each autosemantic word with its associated functional words is collapsed into a single tectogrammatical node, labeled with lemma, functor (semantic role), formeme,[11] and semantically indispensable morphologically categories (such as tense with verbs and number with nouns, but not number with verbs as it is only imposed by subject-predicate agreement). Coreference of pronouns is also resolved and tectogrammatical nodes are enriched with information on named entities (such as the distinction between location, person and organization) resulting from Stanford Named Entity Recognizer [5].

[11] Formemes specify how tectogrammatical nodes are realized in the surface sentence shape. For instance, n:subj stands for semantic noun in the subject position, n:for+X for semantic noun with preposition *for*, v:because+fin for semantic verb in a subordinating clause introduced by the conjunction *because*, adj:attr for semantic adjective in attributive position. Formemes do not constitute a genuine tectogrammatical component as they are not oriented semantically (but rather morphologically and syntactically). However, they have been added to t-trees in Treex as they facilitate the transfer.

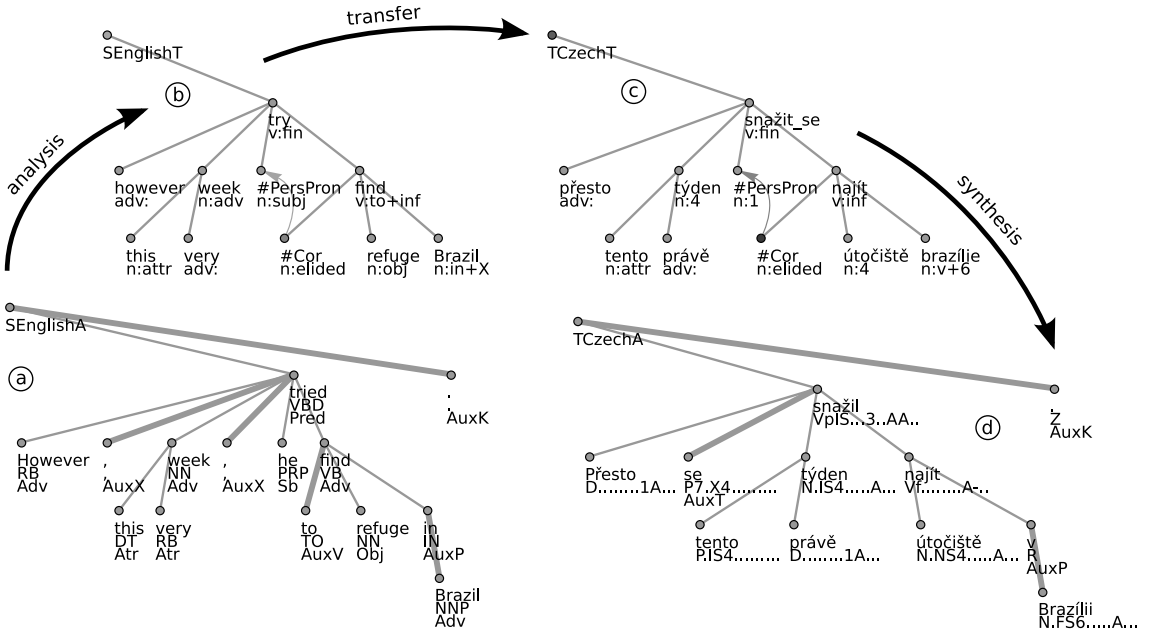

Fig. 2. Analysis-transfer-synthesis translation scenario in Treex applied on the English sentence *"However, this very week, he tried to find refuge in Brazil."*, leading to the Czech translation *"Přesto se tento právě týden snažil najít útočiště v Brazílii."*. Thick edges indicate functional and autosemantic a-nodes to be merged.

4.2 Transfer

The transfer phase follows, whose most difficult part consists in labeling the tree with target-language lemmas and formemes. Changes of tree topology and of other attributes[12] are required relatively infrequently.

Our model for choosing the right target-language lemmas and formemes in inspired by Noisy Channel Model which is the standard approach in the contemporary SMT and which combines a translation model and a language model of the target language. In other words, one should not rely only on the information on how faithfully the meaning is transfered by some translation equivalent, but also the additional model can be used which estimates how well some translation equivalent fits to the surrounding context.[13]

Unlike in the mainstream SMT, in tectogrammatical transfer we do not use this idea for linear structures, but for trees. So the translation model estimates the probability of source and target lemma pair, while the language tree model estimates the probability of a lemma given its parent. The globally optimal tree labelling is then revealed by the tree-modified Viterbi algorithm [22].

Originally, we estimated the translation model simply by using pair frequencies extracted from English-Czech parallel data. A significant improvement was reached after replacing such model by Maximum Entropy model. In the model, we employed a wide range of features resulting from the source-side analysis. The weights were optimized using training data extracted from the CzEng parallel treebank [2], which contains roughly 6 million English-Czech pairs of analyzed and aligned sentences.

4.3 Synthesis

Finally, surface sentence shape is synthesized from the tectogrammatical tree, which is basically a reverse operation for the tectogrammatical analysis: adding punctuation and functional words, spreading morphological categories according to grammatical agreement, performing inflection (using Czech morphology database [7]), arranging word order etc.

4.4 Evaluating translation quality

There are two general methods for evaluating translation quality of outputs of MT systems: (1) the quality can be judged by humans (either using a set of criteria such as grammaticality and intelligibility, or relatively by comparing outputs of different MT systems), or

[12] For instance, number of nouns must be changed to plural if the selected target Czech lemma is a plurale tantum. Similarly, verb tense must be predicted if an English infinitive or gerund verb form is translated to a finite verb form.

[13] This corresponds to the intuition that translating to one's native language is simpler for a human than translating to a foreign language.

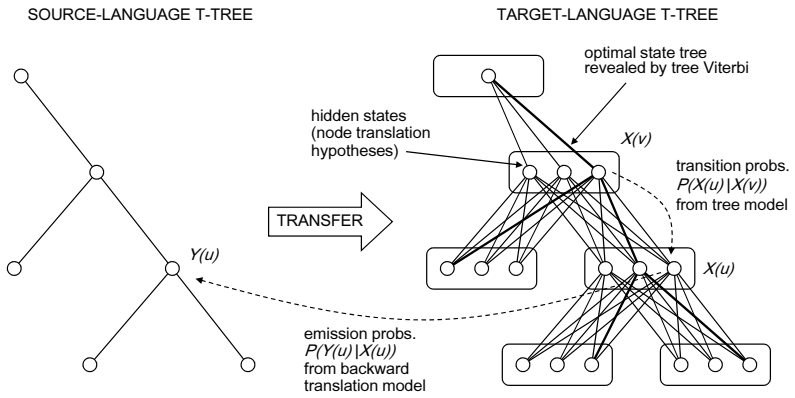

Fig. 3. Tectogrammatical transfer implemented as Hidden Markov Tree Model.

(2) the quality can be estimated by automatic metrics, which usually measure some form of string-wise overlap of an MT system's output with one or more reference (human-made) translations.

Both types of evaluation are used regularly during the development of our MT system. Automatic metrics are used after any change of the translation scenario, as they are cheap and fast to perform. Large scale evaluations by volunteer judges are organized annually as a shared task with the Workshop on Statistical Machine Translation.[14] Performance of the tectogrammatical translation increases every year in both measures, and it already outperforms some commercial as well as academic systems. Actually, it is the participation in this shared task (a competition, in other words) what provides the strongest motivation momentum for Treex developers.

5 Final remarks and conclusions

Even if tectogrammatical translation is considered as the main application of Treex, Treex has been used for a number of other research purposes as well:

- other MT-related tasks – Treex has been used for developing alternative MT quality measures in [9], and for improving outputs of other MT systems by grammatical post-processing in [11],
- building linguistic data resources – Treex has been employed in the development of resources such as the Prague Czech-English Dependency Treebank [21], the Czech-English parallel corpus CzEng [2], and Tamil Dependency Treebank [16].
- linguistic data processing service for other research carried out in other institutions, such as data analyses for prosody prediction for The University of West Bohemia [17].

Treex significantly simplifies code sharing across individual research projects in our institute. There are around 15 programmers (postgraduate students and researchers) who have significantly contributed to the development of Treex in the last years; four of them are responsible for developing the central components of the framework infrastructure called Treex Core.

Last but not least, Treex is used for teaching purposes in our institute. Undergraduate students are supposed to develop their own modules for morphological and syntactic analysis for foreign languages of their choice. Not only that the existence of Treex enables the students to make very fast progress, but their contributions are accumulated in the Treex Subversion repository too, which enlarges the repertory of languages treatable by Treex.[15]

There are two main challenges for the Treex developers now. The first challenge is to continue improving the tectogrammatical translation quality by better exploitation of the training data. The second challenge is to widen the community of Treex users and developers by distributing majority of Treex modules via CPAN (Comprehensive Perl Archive Network), which is a broadly respected repository of Perl modules.

When thinking about a more distant future of MT and NLP in general, an exciting question arises about the future relationship of linguistically interpretable

[14] `http://www.statmt.org/wmt11/`

[15] There are modules for more than 20 languages available in Treex now.

approaches (like that of Treex) and purely statistical phrase-based approaches. Promising results of [11], which uses Treex for improving the output of a phrase-based system and thus reaches the state-of-the-art MT quality in English-Czech MT, show that combinations of both approaches might be viable.

References

1. I. Boguslavsky, L. Iomdin, and V. Sizov: *Multilinguality in ETAP-3: reuse of lexical resources.* In G. Sérasset, (ed.), COLING 2004 Multilingual Linguistic Resources, pp. 1–8, Geneva, Switzerland, August 28 2004. COLING.
2. O. Bojar, M. Janíček, Z. Žabokrtský, P. Češka, and P. Beňa: *CzEng 0.7: parallel corpus with community-supplied translations.* In Proceedings of the Sixth International Language Resources and Evaluation, Marrakech, Morocco, 2008. ELRA.
3. P.E. Brown, V.J. Della Pietra, S.A. Della Pietra, and R.L. Mercer: *The mathematics of statistical machine translation: parameter estimation.* Computational Linguistics, 1993.
4. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan: *GATE: an architecture for development of robust HLT applications.* In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, July, pp. 07–12, 2002.
5. J.R. Finkel, T. Grenager, and C. Manning: *Incorporating non-local information into information extraction systems by gibbs sampling.* In ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.
6. J. Hajič, E. Hajičová, J. Panevová, P. Sgall, P. Pajas, J. Štěpánek, J. Havelka, and M. Mikulová: *Prague Dependency Treebank 2.0.* Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia, 2006.
7. J. Hajič: *Disambiguation of rich inflection – computational morphology of Czech.* Charles University – The Karolinum Press, Prague, 2004.
8. P. Koehn et al: *Moses: open source toolkit for statistical machine translation.* In Proceedings of the Demo and Poster Sessions, 45th Annual Meeting of ACL, pp. 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
9. K. Kos and O. Bojar: *Evaluation of machine translation metrics for Czech as the target language.* Prague Bulletin of Mathematical Linguistics, 92, 2009.
10. A. Lopez: *A survey of statistical machine translation.* Technical Report, Institute for Advanced Computer Studies, University of Maryland, 2007.
11. D. Mareček, R. Rosa, P. Galuščáková, and O. Bojar: *Two-step translation with grammatical post-processing.* In Proceedings of the 6th Workshop on Statistical Machine Translation, pp.426–432, Edinburgh, Scotland, 2011. Association for Computational Linguistics.
12. R . McDonald, F. Pereira, K. Ribarov, and J. 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, pp. 523–530, Vancouver, BC, Canada, 2005.
13. I.A. Mel'čuk: *Dependency syntax: theory and practice.* State University of New York Press, 1988.
14. P. Pajas and J. Štěpánek: *Recent advances in a feature-rich framework for treebank annotation.* In Proceedings of the 22nd International Conference on Computational Linguistics, volume 2, pp. 673–680, Manchester, UK, 2008.
15. M. Popel and Z. Žabokrtský: *TectoMT: modular NLP framework.* In Lecture Notes in Artificial Intelligence, Proceedings of the 7th International Conference on Advances in Natural Language Processing (IceTAL 2010), volume 6233 of *LNCS*, pp. 293–304, Berlin / Heidelberg, 2010. Springer.
16. L. Ramasamy and Z. Žabokrtský: *Tamil dependency parsing: results using rule based and corpus based approaches.* In Proceedings of 12th International Conference CICLing 2011, volume 6608 of Lecture Notes in Computer Science, pp. 82–95, Berlin / Heidelberg, 2011. Springer.
17. J. Romportl: *Zvyšování přirozenosti strojově vytvářené řeči v oblasti suprasegmentálních zvukových jevů.* PhD Thesis, Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic, 2008.
18. P. Sgall, E. Hajičová, and J. Panevová: *The Meaning of the sentence in its semantic and pragmatic aspects.* D. Reidel Publishing Company, Dordrecht, 1986.
19. D. Spoustová, J. Hajič, J. Votrubec, P. Krbec, and P. 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, pp. 67–74, Praha, 2007.
20. F.M. Tyers, F.Sánchez-Martánez, S Ortiz-Rojas, and M.L. Forcada: *Free/open-source resources in the Apertium platform for machine translation research and development.* Prague Bulletin of Mathematical Linguistics, 93, 2010, 67–76.
21. J. Šindlerová, L. Mladová, J. Toman, and S. Cinková: *An application of the PDT-scheme to a parallel treebank.* In Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories (TLT 2007), pp. 163–174, Bergen, Norway, 2007.
22. Z. Žabokrtský and M. Popel: *Hidden Markov tree model in dependency-based machine translation.* In Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics, 2009.
23. Z. Žabokrtský, J. Ptáček, and P. Pajas. *TectoMT: Highly modular MT system with tectogrammatics used as transfer layer.* In Proceedings of the 3rd Workshop on Statistical Machine Translation, ACL, 2008.