# TectoMT: Modular NLP Framework

Martin Popel and Zdeněk Žabokrtský

Charles University in Prague
Institute of Formal and Applied Linguistics
{popel,zabokrtsky}@ufal.mff.cuni.cz

**Abstract.** In the present paper we describe TectoMT, a multi-purpose open-source NLP framework. It allows for fast and efficient development of NLP applications by exploiting a wide range of software modules already integrated in TectoMT, such as tools for sentence segmentation, tokenization, morphological analysis, POS tagging, shallow and deep syntax parsing, named entity recognition, anaphora resolution, tree-to-tree translation, natural language generation, word-level alignment of parallel corpora, and other tasks. One of the most complex applications of TectoMT is the English-Czech machine translation system with transfer on deep syntactic (tectogrammatical) layer. Several modules are available also for other languages (German, Russian, Arabic). Where possible, modules are implemented in a language-independent way, so they can be reused in many applications.

**Keywords:** NLP framework, linguistic processing pipeline, TectoMT.

## 1 Introduction

Most non-trivial NLP (natural language processing) applications exploit several tools (e.g. tokenizers, taggers, parsers) that process data in a pipeline. For developers of NLP applications it is beneficial to reuse available existing tools and integrate them in the processing pipeline. However, it is often the case that the developer has to spend more time with the integration and other auxiliary work than with the development of new tools and innovative approaches. The auxiliary work involves studying documentation of the reused tools, compiling and adjusting the tools in order to run them on the developer's computer, training models if these are needed and not included with the tools, writing scripts for data conversions (the tools may require different input format or encoding), resolving incompatibilities between the tools (e.g. different tagsets assumed), etc. Such a work is inefficient and frustrating. Moreover, if it is done in an ad-hoc style, it must be done again for other applications.

The described drawbacks can be reduced or eliminated by using an NLP framework that integrates the needed tools, so the tools can be combined into various pipelines serving for different purposes. Most of the auxiliary work is already implemented in the framework and developers can focus on the more creative part of their tasks. Some frameworks enable easy addition of third-party tools (usually using so-called wrappers) and development of new modules within the framework.

H. Loftsson, E. Rögnvaldsson, S. Helgadóttir (Eds.): IceTAL 2010, LNAI 6233, pp. 293–304, 2010.

In this paper, we report on a novel NLP framework called TectoMT.[1] In Sect. 2, we describe its architecture and main concepts. Sect. 3 concerns implementation issues. Finally, in Sect. 4, we briefly describe and compare other NLP frameworks.

## 2 TectoMT Architecture

### 2.1 Blocks and Scenarios

TectoMT framework emphasizes modularity and reusability at various levels. Following the fundamental assumption that every non-trivial NLP task can be decomposed into a sequence of subsequent steps, these steps are implemented as reusable components called *blocks*. Each block has a well defined (and documented) input and output specification and also a linguistically interpretable functionality in most cases. This facilitates rapid development of new applications by simply listing the names of existing blocks to be applied to the data. Moreover, blocks in this sequence (which is called *scenario*) can be easily substituted with an alternative solution (other blocks), which attempts at solving the same subtask using a different approach or method.[2]

For example, the task of morphological and shallow-syntax analysis (and disambiguation) for English text consists of five steps: sentence segmentation, tokenization, part-of-speech tagging, lemmatization and parsing. In TectoMT we can arrange various scenarios to solve this task, for example:

| Scenario A | Scenario B |
|---|---|
| `Sentence_segmentation_simple` | `Each_line_as_sentence` |
| `Penn_style_tokenization` | `Tokenize_and_tag` |
| `TagMxPost` | |
| `Lemmatize_mtree` | `Lemmatize_mtree` |
| `McD_parser` | `Malt_parser` |

In the scenario A, tokenization and tagging is done separately in two blocks (`Penn_style_tokenization` and `TagMxPost`, respectively), whereas in the scenario B, the same two steps are done in one block at once (`Tokenize_and_tag`). Also different parsers are used.[3]

[1] `http://ufal.mff.cuni.cz/tectomt/`

[2] Scenarios can be adjusted also by specifying parameters for individual blocks. Using parameters, we can define, for instance, which model should be used for parsing.

[3] `Penn_style_tokenization` is a rule-based block for tokenization according to Penn Treebank guidelines (`http://www.cis.upenn.edu/~treebank/tokenization.html`). `TagMxPost` uses Adwait Ratnaparkhi's tagger [1]. `Tokenize_and_tag` uses Aaron Coburn's `Lingua::EN::Tagger` CPAN module. `Lemmatize_mtree` is a block for English lemmatization handling verbs, noun plurals, comparatives, superlatives and negative prefixes. It uses a set of rules (about one hundred regular expressions inspired by `morpha` [2]) and a list of words with irregular lemmatization. `McD_parser` uses MST parser 0.4.3b [3], `Malt_parser` uses Malt parser 1.3.1 [4].

TectoMT currently includes over 400 blocks – approximately 140 blocks are specific for English, 120 for Czech, 60 for English-to-Czech transfer, 30 for other languages and 50 blocks are language-independent. Some of them contain only few lines of code, some solve complex linguistic phenomena. In order to prevent code duplications, many tools and routines are implemented as separate modules, which can be used in more blocks. TectoMT integrates NLP tools such as:

- five taggers for English, three taggers for Czech, one tagger for German, Russian and Spanish,
- two constituency parsers for English, two dependency parsers for English, three dependency parsers for Czech, two dependency parsers for German,
- a named entity recognizer for English, and two named entity recognizers for Czech.

New components are still being added, as there are more than ten programmers contributing to the TectoMT repository at present.

### 2.2 Applications

Applications in TectoMT correspond to end-to-end NLP tasks, be they real end-user applications (such as machine translation), or only NLP-related experiments. Applications usually consist of three phases:

1. conversion of the input data to the TectoMT internal format, possibly split into more files,
2. applying a scenario (i.e. a sequence of blocks) to the files,
3. conversion of the resulting files to the desired output format.

Technically, applications are often implemented as Makefiles, which only glue the three phases.

Besides developing the English-Czech translation system [5], TectoMT was also used in applications such as:

- machine translation based on Synchronous Tree Substitution Grammars and factored translation [6],
- aligning tectogrammatical structures of parallel Czech and English sentences [7],
- building a large, automatically annotated parallel English-Czech treebank CzEng 0.9 [8],
- compiling a probabilistic English-Czech translation dictionary [9],
- evaluating metrics for measuring translation quality [10],
- complex pre-annotation of English tectogrammatical trees within the Prague Czech English Dependency Treebank project [11],
- tagging the Czech data set for the CoNLL Shared Task [12],
- gaining syntax-based features for prosody prediction [13],
- experiments on information retrieval [14],
- experiments on named entity recognition [15],
- conversion between different deep-syntactic representations of Russian sentences [16].

### 2.3 Layers of Language Description

TectoMT profits from the stratificational approach to the language, namely it defines four layers of language description (listed in the order of increasing level of abstraction): raw text (word layer, w-layer), morphological layer (m-layer), shallow-syntax layer (analytical layer, a-layer), and deep-syntax layer (layer of linguistic meaning, tectogrammatical layer, t-layer).

The strategy is adopted from the Functional Generative Description theory [17], which has been further elaborated and implemented in the Prague Dependency Treebank (PDT) [18]. We give here only a very brief summary of the key points.

- **morphological layer (m-layer)**
  Each sentence is tokenized and each token is annotated with a lemma and morphological tag. For details see [19].
- **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). For details see [20].
- **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, 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 [21].

Apart from the described layers, TectoMT also defines phrase-structure layer (p-layer) for storing constituency trees. This layer is approximately on the same level of abstraction as the a-layer.

### 2.4 Documents, Bundles and Trees

Every document is saved in one file and consists of a sequence of sentences. Each sentence is represented by a structure called *bundle*, which stands for 'a bundle of trees'. Each tree can be classified according to:

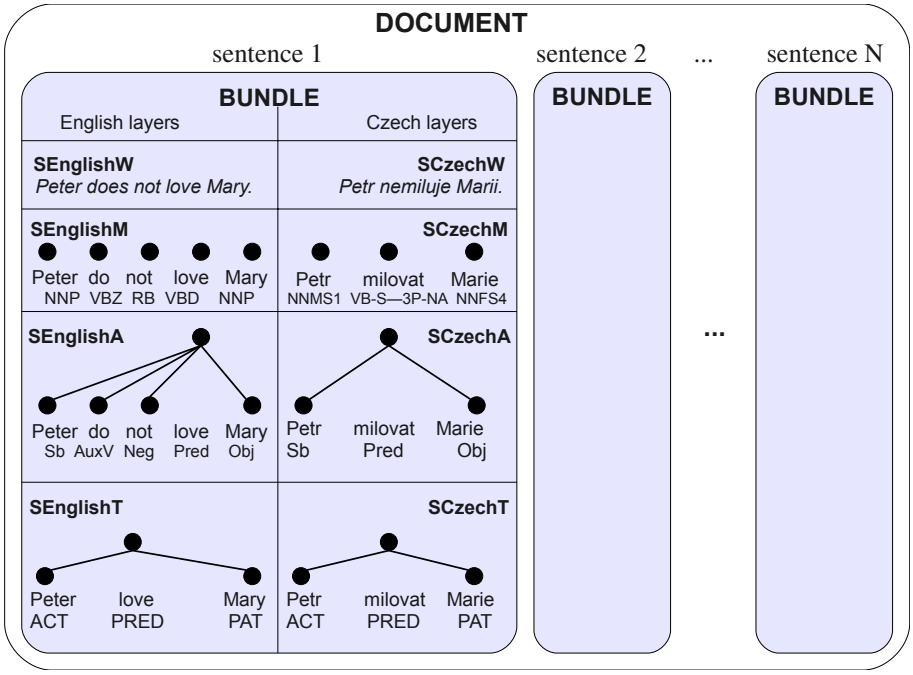


**Fig. 1.** English-Czech parallel text annotated on three layers of language description is saved in a TectoMT document, each sentence in one bundle. We show only a simplified representation of the trees in the first bundle.

- layer of language description (M=m-layer, A=a-layer, T=t-layer),
- language (e.g. Arabic, Czech, English, German[4]),
- indication whether the sentence was created by analysis (S=source) or by transfer or synthesis (T=target).

In other words, each bundle contains trees that represent the same sentence in different languages, layers and source/target direction (hence sentences in multilingual documents are implicitly aligned, see Fig. 1). TectoMT trees are denoted by the three coordinates, e.g. analytical layer representation of an English sentence acquired by analysis is denoted as `SEnglishA`, tectogrammatical layer representation of a sentence translated to Czech is denoted as `TCzechT`. This naming convention is used on many places in TectoMT: for naming blocks (see Sect. 3), for naming node identifiers, etc. The convention is extremely useful for a machine translation that follows the analysis-transfer-synthesis scheme as illustrated in Fig. 2 using Vauquois diagram. Nevertheless, also the blocks for other NLP tasks can be classified according to the languages and layers on which they operate.

[4] In the near future, TectoMT will migrate to using ISO 639 language codes (e.g. ar, cs, en, de) instead of full names.

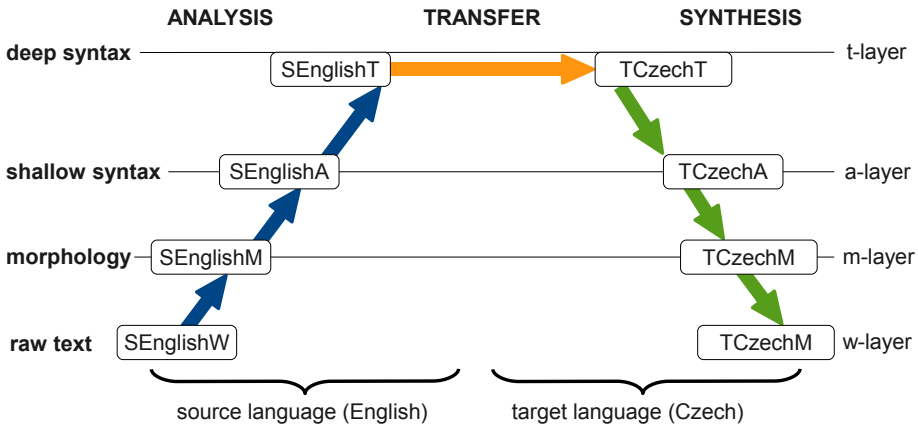


**Fig. 2.** Vauquois diagram for translation with transfer on tectogrammatical layer

A-layer and t-layer structures are dependency trees, so it is natural to handle them as tree data structures. M-layer structure is a sequence of tokens with associated attributes, which is handled in TectoMT as a special case of a tree (all nodes except the technical root are leaves of the tree). W-layer (raw text) is represented as string attributes stored within bundles.

## 3 TectoMT Implementation

### 3.1 Design Decisions

TectoMT is implemented in Perl programming language under Linux. This does not exclude the possibility of releasing platform-independent applications made of selected components (platform-independent solutions are always preferred in TectoMT). TectoMT modules are programmed in object-oriented programming style using inside-out classes (following [22]). Some of the modules are just Perl wrappers for tools written in other languages (especially Java and C).

TectoMT is a modern multilingual framework and it uses open standards such as Unicode and XML. TectoMT is neutral with respect to the methodology employed in the individual blocks: fully stochastic, hybrid, or fully rule-based approaches can be used.

### 3.2 TectoMT Components and Directory Structure

Each block is a Perl class inherited from `TectoMT::Block` and each block is saved in one file. The blocks are distributed into directories according to the languages and layers on which they operate. For example, all blocks for deep-syntactic analysis of English (i.e. for generating t-trees from a-trees) are stored in a directory `SEnglishA_to_SEnglishT`. In this paper, we use for simplicity only short names

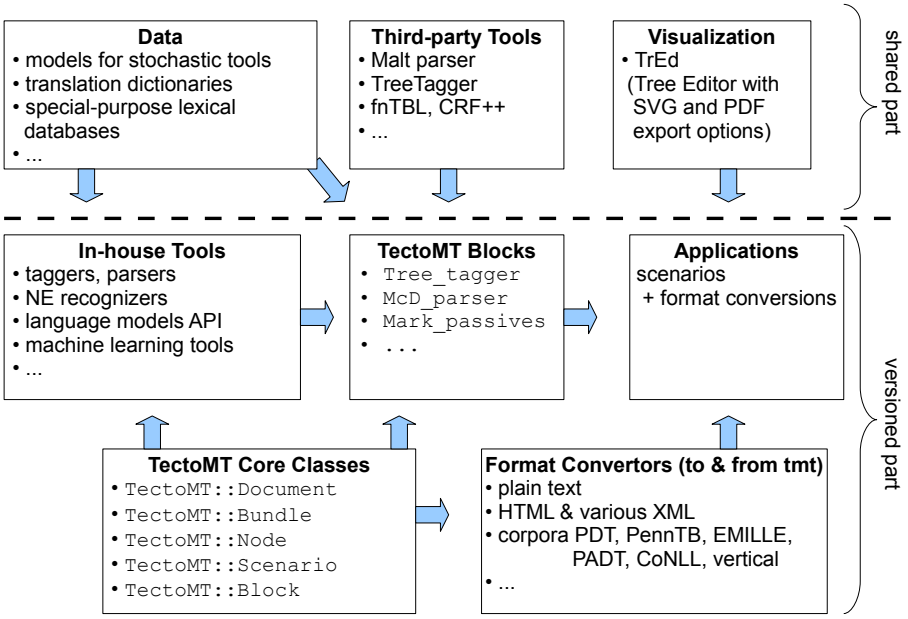


**Fig. 3.** Components of the TectoMT framework

of blocks, so e.g. instead of the full name `SEnglishA_to_SEnglishT::Assign_grammatemes` we write only `Assign_grammatemes`.

TectoMT is composed of two parts (see Fig. 3). The first part (the *versioned* part), which contains TectoMT core classes and utilities, format converters, blocks, applications, and in-house tools, is stored in an SVN repository, so that it can be developed in parallel by more developers. The second part (the *shared* part), which contains linguistic data resources, downloaded third-party tools and the software for visualization of TectoMT files (Tree editor TrEd [23]), is shared without versioning because (a) it is supposed to be changed rather additively, (b) it is huge, as it contains large data resources, and (c) it should be automatically reconstructible simply by downloading (and installing) the needed components.

### 3.3 Data Formats

The internal TectoMT format (`tmt` files) is an XML with a schema defined in Prague Markup Language [24]. It is similar to the format used for the Prague Dependency Treebank 2.0 [18], but all representations of a textual document at the individual layers of language description are stored in a single file. TectoMT includes converters for various formats and corpora (e.g. Penn Treebank [25], CoNLL [12], EMILLE [26], PADT [27]).

Scenarios are saved in plain text files with a simple format that enables including of other scenarios.

### 3.4 Parallel Processing of Large Data

TectoMT can be used for processing huge data resources using a cluster of computers.[5] There are utilities that take care of input data distribution, filtering, parallel processing, logging, error checks, and output data collection. In order to allow efficient and flawless processing, input data (i.e. corpora) should be distributed into many `tmt` files (with about 50 to 200 sentences per file). Each file can be processed independently, so this method scales well to any number of computers in a cluster. For example, the best version of translation from English to Czech takes about 1.2 seconds per sentence plus 90 seconds for initial loading of blocks in memory per computer (more precisely per cluster job).[6] Using 20 computers in a cluster we can translate 2000 sentences in less than 4 minutes.

TectoMT was used to automatically annotate the parallel treebank CzEng 0.9 [8] with 8 million sentences, 93 million English and 82 million Czech words.

## 4 Other NLP Frameworks

In Tab. 1, we summarize some properties of TectoMT and four other NLP frameworks:

- ETAP-3 is an NLP framework for English-Russian and Russian-English translation developed in the Computational linguistics laboratory of the Institute for Information Transmission Problems of the Russian Academy of Sciences. [28]
- GATE is one of the most widely used NLP frameworks with integrated graphical user interface. It is being developed at University of Sheffield. [29]
- OpenNLP[7] is an organizational center for open source NLP projects, which offers several NLP tools (and maximum entropy language models).
- WebLicht[8] is a Service Oriented Architecture (SOA) for building annotated German text corpora.

Each of the frameworks was developed for different purposes with different preferences in mind, so it is not possible to choose the universally best framework. There is a number of NLP frameworks for shallow analysis or translation (apart from those listed in Tab. 1, e.g. Apertium [31] or UIMA [32]). However, we are aware only of two tree-oriented (rather than sequence-oriented or chunk-oriented) frameworks capable of deep syntax analysis (and translation) – TectoMT and ETAP-3. Unlike ETAP-3, TectoMT is publicly available and allows for combining statistical and rule-based approaches (whereas ETAP-3 is strictly rule-based).

[5] We use Sun Grid Engine, `http://gridengine.sunsource.net/`

[6] Most of the initialization time is spent with loading translation and language models (about 8 GiB). Other applications presented in this paper are not so resource-demanding, so they are loaded in a few seconds.

[7] `http://opennlp.sourceforge.net`

[8] `http://weblicht.sfs.uni-tuebingen.de/englisch/index.shtml`

**Table 1.** Comparison of NLP frameworks. Notes:
a: ETAP-3 is a closed-source project, only a small demo is available online – `http://cl.iitp.ru/etap`
b: WebLicht is designed to offer web services for five universities in Germany, but we have not found any service for public use.
c: Functional Generative Description theory [17]
d: Meaning-Text Theory [30]
e: The purpose of NLP frameworks is to serve for various applications. However, some applications may be considered characteristic for a given framework.
MT = Machine Translation, IE = information extraction.

| | TectoMT | ETAP-3 | GATE | OpenNLP | WebLicht |
|---|---|---|---|---|---|
| developed since | 2005 | 1980s | 1996 | 2003 | 2008 |
| license for public use | GPL | no[a] | LGPL | LGPL | no[b] |
| main prog. language | Perl | C/C++ | Java | Java | ? |
| linguistic theory | FGD[c] | MTT[d] | | | |
| strictly rule-based | no | yes | no | no | no |
| main application[e] | MT | MT | IE | | annotation |
| uses deep syntax | yes | yes | no | no | no |

## 5 Summary

TectoMT is a multilingual NLP framework with a wide range of applications and integrated tools. Its main properties are:

- **emphasized efficient development, modular design and reusability**
  There are cleanly separated low-level core routines (for processing documents, handling dependency trees and data serialization) and blocks for processing linguistic task. The users can easily add new blocks, which are written in a full-fledged programming language. TectoMT also integrates third-party tools and software for viewing the processed data.
- **stratificational approach to the language**
  TectoMT uses four layers of language description, which are linguistically interpretable (though this does not mean that TectoMT is a strictly rule-based framework). On the shallow and deep syntactic layer, sentences are represented as dependency trees. Annotation conventions are adopted mainly from commonly used corpora (PennTB, PDT).
- **unified object-oriented interface for accessing data structures**
  TectoMT tries to minimize file-based communication between the blocks in processing pipelines. The unified object-oriented interface allows for processing large amounts of data with complex data structures.
- **comfortable development**
  The analysis of sentences can be examined and edited in the tree editor TrEd. TectoMT also offers tools for parallel processing, testing and debugging, standard structure of blocks and documentation.

**Acknowledgments.** This work was supported by the grants GAUK 116310, MSM0021620838, MŠMT ČR LC536, and FP7-ICT-2007-3-231720 (EuroMatrix Plus). We thank three anonymous reviewers for helpful comments.

## References

1. Ratnaparkhi, A.: A maximum entropy part-of-speech tagger. In: Proceedings of the conference on Empirical Methods in Natural Language Processing, pp. 133–142 (1996)
2. Minnen, G., Carroll, J., Pearce, D.: Robust Applied Morphological Generation. In: Proceedings of the 1st International Natural Language Generation Conference, Israel, pp. 201–208 (2000)
3. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: 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), Vancouver, BC, Canada, pp. 523–530 (2005)
4. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E.: MaltParser: A language-independent system for data-driven dependency parsing. Natural Language Engineering 13(2), 95–135 (2007)
5. Bojar, O., Mareček, D., Novák, V., Popel, M., Ptáček, J., Rouš, J., Žabokrtský, Z.: English-Czech MT in 2008. In: Proceedings of the Fourth Workshop on Statistical Machine Translation, Association for Computational Linguistics, Athens, Greece, pp. 125–129 (March 2009)
6. Bojar, O., Hajič, J.: Phrase-Based and Deep Syntactic English-to-Czech Statistical Machine Translation. In: ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation, Association for Computational Linguistics, Columbus, OH, USA, pp. 143–146 (2008)
7. Mareček, D., Žabokrtský, Z., Novák, V.: Automatic Alignment of Czech and English Deep Syntactic Dependency Trees. In: Hutchins, J., Hahn, W. (eds.) Proceedings of the Twelfth EAMT Conference, Hamburg, HITEC e.V, pp. 102–111 (2008)
8. Bojar, O., Žabokrtský, Z.: Building a Large Czech-English Automatic Parallel Treebank. Prague Bulletin of Mathematical Linguistics 92 (2009)
9. Rouš, J.: Probabilistic translation dictionary. Master's thesis, Faculty of Mathematics and Physics, Charles University in Prague (2009)
10. Kos, K., Bojar, O.: Evaluation of Machine Translation Metrics for Czech as the Target Language. Prague Bulletin of Mathematical Linguistics 92 (2009)
11. Hajič, J., Cinková, S., Čermáková, K., Mladová, L., Nedolužko, A., Petr, P., Semecký, J., Šindlerová, J., Toman, J., Tomšů, K., Korvas, M., Rysová, M., Veselovská, K., Žabokrtský, Z.: Prague English Dependency Treebank, Version 1.0 (January 2009)
12. Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M.A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., Zhang, Y.: The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In: Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), Boulder, Colorado, USA, June 4-5 (2009)

13. Romportl, J.: Zvyšování přirozenosti strojově vytvářené řeči v oblasti suprasegmentálních zvukových jevů. PhD thesis, Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic (2008)
14. Kravalová, J.: Využití syntaxe v metodách pro vyhledávání informací (using syntax in information retrieval). Master's thesis, Faculty of Mathematics and Physics, Charles University in Prague (2009)
15. Kravalová, J., Žabokrtský, Z.: Czech Named Entity Corpus and SVM-based Recognizer. In: Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009), Association for Computational Linguistics, Suntec, Singapore, pp. 194–201 (2009)
16. Mareček, D., Kljueva, N.: Converting Russian Treebank SynTagRus into Praguian PDT Style. In: Proceedings of the RANLP 2009, International Conference on Recent Advances in Natural Language Processing, Borovets, Bulgaria (2009)
17. Sgall, P.: Generativní popis jazyka a česká deklinace. Academia, Prague (1967)
18. Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M.: Prague Dependency Treebank 2.0. Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia (2006)
19. Zeman, D., Hana, J., Hanová, H., Hajič, J., Hladká, B., Jeřábek, E.: A Manual for Morphological Annotation, 2nd edn., Technical Report 27, ÚFAL MFF UK, Prague, Czech Republic (2005)
20. Hajičová, E., Kirschner, Z., Sgall, P.: A Manual for Analytic Layer Annotation of the Prague Dependency Treebank (English translation). Technical report, ÚFAL MFF UK, Prague, Czech Republic (1999)
21. Mikulová, M., Bémová, A., Hajič, J., Hajičová, E., Havelka, J., Kolářová, V., Kučová, L., Lopatková, M., Pajas, P., Panevová, J., Razímová, M., Sgall, P., Štěpánek, J., Urešová, Z., Veselá, K., Žabokrtský, Z.: Annotation on the tectogrammatical level in the Prague Dependency Treebank. Annotation manual. Technical Report 30, ÚFAL MFF UK, Prague, Czech Rep (2006)
22. Conway, D.: Perl Best Practices. O'Reilly Media, Inc., Sebastopol (2005)
23. Pajas, P., Štěpánek, J.: Recent advances in a feature-rich framework for treebank annotation. In: Scott, D., Uszkoreit, H. (eds.) The 22nd International Conference on Computational Linguistics - Proceedings of the Conference, The Coling 2008 Organizing Committee, Manchester, UK, vol. 2, pp. 673–680 (2008)
24. Pajas, P., Štěpánek, J.: XML-based representation of multi-layered annotation in the PDT 2.0. In: Hinrichs, R.E., Ide, N., Palmer, M., Pustejovsky, J. (eds.) Proceedings of the LREC Workshop on Merging and Layering Linguistic Information (LREC 2006), Genova, Italy, pp. 40–47 (2006)
25. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics 19(2), 313–330 (1994)
26. McEnery, A., Baker, P., Gaizauskas, R., Cunningham, H.: EMILLE: Building a corpus of South Asian languages. Vivek-Bombay 13(3), 22–28 (2000)
27. Smrž, O., Bielický, V., Kouřilová, I., Kráčmar, J., Hajič, J., Zemánek, P.: Prague Arabic Dependency Treebank: A Word on the Million Words. In: Proceedings of the Workshop on Arabic and Local Languages (LREC 2008), Marrakech, Morocco, pp. 16–23 (2008)
28. Boguslavsky, I., Iomdin, L., Sizov, V.: Multilinguality in ETAP-3: Reuse of Lexical Resources. In: Sérasset, G. (ed.) COLING 2004 Multilingual Linguistic Resources, Geneva, Switzerland, August 28, pp. 1–8 (2004)

29. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: an architecture for development of robust HLT applications. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, July 07-12 (2002)
30. Mel'čuk, I.A.: Towards a functioning model of language. Mouton (1970)
31. Tyers, F.M., Sánchez-Martínez, F., Ortiz-Rojas, S., Forcada, M.L.: Free/open-source resources in the Apertium platform for machine translation research and development. Prague Bulletin of Mathematical Linguistics 93, 67–76 (2010)
32. Wilcock, G.: Linguistic Processing Pipelines: Problems and Solutions. In: Book of Abstracts GSCL Workshop: Linguistic Processing Pipelines (2009)