Treex: Modular NLP Framework

Martin Popel
ÚFAL (Institute of Formal and Applied Linguistics)
Charles University in Prague

September 2015, Prague, MT Marathon
Outline

- Motivation, Treex vs. TectoMT
- Treex architecture
- Treex internals
- Future plans
- Conclusion and examples
Motivation

Goals of Treex

- elegant integration of in-house and third-party NLP tools
- modularity, reusability, cooperation
- ability to easily modify and add code in a full-fledged programming language (Perl)
Treex vs. TectoMT

2005 (Zdeněk Žabokrtský)

NLP framework

TectoMT

MT system

TectoMT

lemmatization
tagging

parsing
## Treex vs. TectoMT

<table>
<thead>
<tr>
<th>Year</th>
<th>NLP Framework</th>
<th>MT System</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>TectoMT</td>
<td>TectoMT</td>
<td>lemmatization, tagging, parsing</td>
</tr>
<tr>
<td></td>
<td>multi-purpose</td>
<td>Treex</td>
<td>lemmatization, tagging, parsing, coreference, PEDT preprocessing, CzEng analysis, named entity resolution, treebank conversions, alignment (word,tree), SMT preproc., etc.</td>
</tr>
</tbody>
</table>

---

**NLP framework**
- **TectoMT**
  - MT system: TectoMT
  - Features: lemmatization, tagging, parsing

**Multi-purpose NLP framework**
- **Treex**
  - MT system: TectoMT
  - Features: lemmatization, tagging, parsing, coreference, PEDT preprocessing, CzEng analysis, named entity resolution, treebank conversions, alignment (word,tree), SMT preproc., etc.
## Treex vs. TectoMT

**2005**

**NLP framework**
- TectoMT

**MT system**
- TectoMT

- lemmatization
- tagging
- parsing

**2011**

**multi-purpose NLP framework**
- Treex

**MT system**
- TectoMT

- lemmatization
- tagging
- parsing

- coreference
- CzEng analysis
- named entity r.
- SMT preproc.

- PEDT preprocessing
- treebank conversions
- alignment (word,tree)
- etc.

---

Now not only **tecto**grammatics and not only **MT** renamed.
Treex vs. TectoMT

2005

NLP framework
TectoMT

MT system
TectoMT

lemmatization
tagging
parsing

redesigned and reimplemented
easier to use
more flexible

2011

multi-purpose NLP framework
Treex

MT system
TectoMT

lemmatization
tagging
parsing

coreference
PEDT preprocessing
CzEng analysis
treebank conversions
named entity r.
alignment (word,tree)
SMT preproc.

etc.
**Treex vs. TectoMT**

2005

- English
- Czech

**NLP framework**

- **TectoMT**

**MT system**

- **TectoMT**

- **TectoMT**

- **English**
- **Czech**
- **Russian**
- **Tamil**
- **Polish**
- **Esperanto**
- **Spanish**
- **French**
- **German**
- **Arabic**
- **Vietnamese**
- **Hindi**
- **Urdu**
- **Finish**

- **CzEng analysis**
- **named entity r.**
- **PEDT preprocessing**
- **treebank conversions**
- **alignment (word,tree)**
- **SMT preproc.**

- **lemmatization**
- **tagging**
- **parsing**

- **redesigned and reimplemented**
  - **easier to use**
  - **more flexible**
  - **more langs**

*) Most of the listed languages are only drafts of analysis made by students, not converted to Treex yet. The entire risk as to the quality and performance of the program is with you.
Treex vs. TectoMT

2005

Treex

TectoMT

MT system

TectoMT

multi-purpose

NLP framework

Treex

lemmatization
tagging

parsing

coreference

CzEng analysis

PEDT preprocessing
treebank conversions
alignment (word,tree)
SMT preproc.

etc.

redesigned and reimplemented

easier to use
easier to use
more flexible
more flexible
more langs

more langs

English

Czech

Russian

Tamil

Esperanto

French

German

Arabic

Vietnamese

Hindi

Urdu

Finish

*) Most of the listed languages are only drafts of analysis made by students, not converted to Treex yet. The entire risk as to the quality and performance of the program is with you.

Special offer

Call now and get one extra Treex for free
TectoMT

linguistically motivated MT system (English to Czech pilot)

- deep syntactic (tectogrammatical) transfer
- translation process divided to more than 90 “blocks“
- combining statistical and rule based blocks

**ANALYSIS**
- tectogrammatical layer
  - fill formems
  - grammatemes
  - build t-tree
  - mark edges to contract

**TRANSFER**
- analytical layer
  - analytical functions
  - parser (McDonald's MST)
  - tagger (Morce)

**SYNTHESIS**
- morphological layer
  - lemmatization
  - segmentation

- source language (English)
- target language (Czech)
- t-layer
- a-layer
- m-layer
- w-layer

- query dictionary
- use HMTM
- fill morphological categories
- impose agreement
- add functional words
- generate wordforms
- concatenate
4 layers of language description implemented in Prague Dependency Treebank (PDT)

- **tectogrammatical layer**
  deep-syntactic dependency trees

- **analytical layer**
  surface-syntactic dependency trees, labeled edges

- **morphological layer**
  lemma & POS tag for each word

- **word layer**
  raw (tokenized) text
4 layers of language description implemented in Prague Dependency Treebank (PDT)

- **tectogrammatical layer**
  - deep-syntactic dependency trees
  - abstraction from many language-specific phenomena
  - autosemantic (meaningful) words ~ nodes
  - functional words (prepositions, auxiliaries) ~ attributes
  - syntactic-semantic relations (dependencies) ~ edges
  - added nodes (e.g. because of pro-drop)
  - ...
layers of language description implemented in Treex

- Mostly backward compatible adaptations (adding attributes)
  - **formeme** \( (n:2, n:k+3, v:že+vfin, v:rc, adj:attr) \)
  - attributes for clauses, is_passive \( \rightarrow \) diathesis, ...
  - is_member (for conjuncts on a-layer) is stored with prepositions

- All layers stored in **one file**
- A-layer and m-layer merged into one
- Two more layers:
  - P-layer phrase-structure trees
  - N-layer named entities
Treex architecture

- **input files**
- **document reader**
- **In-memory document representation (OOP API)**
- **block 1**
- **block n**
- **document writer**
- **output files**
Treex architecture parallelization (using SGE cluster)

- Input files
- In-memory document
- Document reader
- Block
- ... (Multiple blocks)
- Writer
- Output files
Treex architecture
processing units

- **block** – elementary processing unit in Treex
  - corresponding to a given NLP subtask
  - one Perl class (Treex::Block::*), saved in one file
- **scenario** – a sequence of blocks
  - saved in plain text files or a Treex::Scen::* Perl class
  - just a list of the blocks' names and their parameters
- **application** – represents an end-to-end NLP task
  - described by a scenario that
    - starts with a **reader** (input conversion)
    - ends with a **writer** (output conversion)
- Readers can split the input file into more in-memory docs.
- There are readers&writers for a number of popular formats: plain text, CoNLL, PDT PML, Penn MRG, Tiger...
  
  * .treex.gz
Treex architecture processing units

Blocks can be easily substituted with an alternative solution.

Scenario 1:  
Scenario 2:  
Scenario 3:
Treex architecture processing units

Blocks can be easily substituted with an alternative solution.

<table>
<thead>
<tr>
<th>Scenario A</th>
<th>Scenario B</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2A::EN::Segment</td>
<td>W2A::SegmentOnNewlines</td>
</tr>
<tr>
<td>W2A::EN::Tokenize</td>
<td>W2A::EN::TagLinguaEn</td>
</tr>
<tr>
<td>W2A::EN::TagMorce</td>
<td></td>
</tr>
<tr>
<td>W2A::EN::Lemmatize</td>
<td>W2A::EN::Lemmatize</td>
</tr>
<tr>
<td>W2A::EN::ParseMST</td>
<td>W2A::EN::ParseMalt</td>
</tr>
</tbody>
</table>
Treex architecture
data units

- **Document**
  - stored in one file
  - sequence of sentences

- **Bundle** (“bundle of trees”)
  - corresponds to one sentence

- **Zone**
  - one for each language (Arabic, Czech, English,...)
  - and optionally a variant (“selectors” src, trg, ref,...)

- **Tree**
  - layer of language description: A, T (plus P, N)
  - m-layer is stored with the a-layer in one tree
Treex architecture
data units
Treex architecture
data units

**DOCUMENT**

<table>
<thead>
<tr>
<th>Zone en_src</th>
<th>BUNDLE</th>
<th>Zone cs_src</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>W-layer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Peter does not love Mary.</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>M-layer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peter do not love Mary</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A-layer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peter do not love Mary</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>T-layer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peter love Mary</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>BUNDLE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>W-layer</strong></td>
</tr>
<tr>
<td><em>Petr nemiluje Marii.</em></td>
</tr>
<tr>
<td><strong>M-layer</strong></td>
</tr>
<tr>
<td>Petr milovat Marie</td>
</tr>
<tr>
<td><strong>A-layer</strong></td>
</tr>
<tr>
<td>Petr milovat Marie</td>
</tr>
<tr>
<td><strong>T-layer</strong></td>
</tr>
<tr>
<td>Petr milovat Marie</td>
</tr>
</tbody>
</table>

...
Treex architecture

data units

DOCUMENT

sentence 1

BUNDLE

Zone en_src

W-layer

Peter does not love Mary.

M-layer

Peter NNP do VBZ Obj VBD

A-layer

Sb AuxV Neg Pred Obj

T-layer

Peter ACT love PRED Mary PAT

sentence 2

...  

sentence N

BUNDLE

Zone cs_trg

W-layer

Petr nemiluje Marii.

M-layer

Petr NNMS1 milovat Obj VBD

A-layer

Sb Pred Obj

T-layer

Petr ACT milovat PRED Marie PAT

BUNDLE

...
Internals – Design decisions

- Perl (wrappers for binaries, Java,...)
- Linux (some applications platform-independent)
- OOP (Moose)
- Open source (GNU GPL for the versioned part)
- Neutral w.r.t. methodology (statistical, rule-based)
- Multilingual
- Open standards (Unicode, XML)
Internals – Components

Treex Core Classes
- Treex::Core::Document
- Treex::Core::Node
- Treex::Core::Scenario
- Treex::Core::Block

Data
- models for stochastic tools
- translation dictionaries
- special-purpose lexical databases

Third-party Tools
- Malt parser
- TreeTagger
- fnTBL, CRF++

Visualization
- TrEd
  (Tree Editor with SVG and PDF export options)

In-house Tools
- taggers, parsers
- NE recognizers
- language models API
- machine learning tools

Treex Blocks
- TagTNT
- ParseMST
- MarkPassives

Applications
- scenarios
- + format conversions

Readers and Writers
- plain text
- HTML & various XML
- corpora PDT, PennTB, EMILLE, PADT, CoNLL, vertical

In-house Tools
- taggers, parsers
- NE recognizers
- language models API
- machine learning tools

Treex Blocks
- TagTNT
- ParseMST
- MarkPassives

Applications
- scenarios
- + format conversions

Readers and Writers
- plain text
- HTML & various XML
- corpora PDT, PennTB, EMILLE, PADT, CoNLL, vertical

Data
- models for stochastic tools
- translation dictionaries
- special-purpose lexical databases

Third-party Tools
- Malt parser
- TreeTagger
- fnTBL, CRF++

Visualization
- TrEd
  (Tree Editor with SVG and PDF export options)

In-house Tools
- taggers, parsers
- NE recognizers
- language models API
- machine learning tools

Treex Blocks
- TagTNT
- ParseMST
- MarkPassives

Applications
- scenarios
- + format conversions

Readers and Writers
- plain text
- HTML & various XML
- corpora PDT, PennTB, EMILLE, PADT, CoNLL, vertical
Internals – Statistics

- Developed since 2005, over ten developers
- Over 400 blocks (140 English, 120 Czech, 60 English-to-Czech, 30 other languages, 50 language independent)
- Taggers (5 English, 3 Czech, 1 German and Russian, Tamil)
  Parsers (Dep. 2 English, 3 Czech, 2 German; Const. 2 English)
  Named Entity Recognizers (2 Czech, 1 English)
- Speed example: Best version of English-to-Czech MT
  1.2 seconds per sentence plus 90 seconds loading, with 20 computers in cluster: 2000 sentences in 4 min
Conclusion
Treex main properties

- emphasized efficient development, modular design and reusability
- stratificational approach to the language
- unified object-oriented interface for accessing data structures
- comfortable development
The firm began series production last November.
Firma začala výrobu řady poslední listopad.
TrEd visualization

word alignment on the morphological layer
TrEd visualization

word alignment on the tectogrammatical layer
TrEd visualization

named entities

Tři utonulí jsou z Jeseníku nad Odrou na Novojičínsku a jedna žena utonula v Novém Jičíně Žilně.
package Tutorial::Svo2SovSolution;
use Moose;
use Treex::Core::Common;
extends 'Treex::Core::Block';

sub process_anode {
    my ( $self, $a_node ) = @_;  
    if ( $a_node->tag =~ /^V/ ) {  # verb found
        foreach my $child ( $a_node->get_echildren() ) {
            if ( $child->afun eq 'Obj' ) {  # object found
                # Move the object and its subtree so it precedes the verb
                $child->shift_before_node($a_node);
            }
        }
    }
    return;
}
Thank you

Cooperation is welcomed.

http://ufal.mff.cuni.cz/treex
Thank you

Treex is growing!

http://ufal.mff.cuni.cz/treex