Treex: Modular NLP Framework

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

September 2011, Malá Skála
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

2005

NLP framework

*TectoMT*

MT system

*TectoMT*

- lemmatization
- tagging
- parsing

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                           …                            2011

NLP framework
TectoMT

MT system
TectoMT

lemmatization

tagging

Now not only
tectogrammatics
and not only MT
renamed

MT system
TectoMT

lemmatization
tagging

coreference
PEDT preprocessing

CzEng analysis
treebank conversions

named entity r.
alignment (word,tree)

SMT preproc.
etc.
Treex vs. TectoMT

2005  |  ...  |  2011

NLP framework

*Treex*

MT system

*TectoMT*

MT system

*TectoMT*

multi-purpose NLP framework

*Treex*

- lemmatization
- tagging
- parsing
- coreference
- PEDT preprocessing
- CzEng analysis
- treebank conversions
- named entity r.
- alignment (word,tree)
- SMT preproc.
- etc.

redesigned and reimplemented
easier to use
more flexible
Treex vs. TectoMT

2005

NLP framework

TectoMT

MT system

TectoMT

redesigned and reimplemented

easier to use

more flexible

more langs

2005

English

Czech

NLP framework

TectoMT

MT system

TectoMT

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.
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.
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
  - analytical functions

TRANSFER

- query dictionary
- use HMTM

SYNTHESIS

- fill morphological categories
- impose agreement
- add functional words
- generate wordforms
- concatenate

morphological layer

- parser (McDonald’s MST)
- tagger (Morce)
- lemmatization
- tokenization
- segmentation

source language (English)

target language (Czech)

a-layer

m-layer

w-layer
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
depth-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 (→ 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

In-memory document representation (OOP API)

input files → document reader → block 1 → ... → block n → document writer → output files
Treex architecture parallelization (using SGE cluster)

- **Input files**
  - **Document reader**
  - **Block**
- **In-memory document**
  - **Block**
  - **Writer**
- **Output files**
  - **Document reader**
  - **Block**
  - **In-memory document**
  - **Block**
  - **Writer**
Treex architecture processing units

- **block** – elementary processing unit in Treex
  - corresponding to a given NLP subtask
  - one Perl class, saved in one file
- **scenario** – a sequence of blocks
  - saved in plain text files
  - 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.

Scenario A
W2A::EN::Segment
W2A::EN::Tokenize
W2A::EN::TagMorce
W2A::EN::Lemmatize
W2A::EN::ParseMST

Scenario B
W2A::SegmentOnNewlines
W2A::EN::TagLinguaEn
W2A::EN::Lemmatize
W2A::EN::ParseMalt
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**

<table>
<thead>
<tr>
<th>BUNDLE</th>
<th>BUNDLE</th>
<th>BUNDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Zone en</strong>&lt;sub&gt;src&lt;/sub&gt;</td>
<td><strong>Zone cs</strong>&lt;sub&gt;src&lt;/sub&gt;</td>
<td><strong>BUNDLE</strong></td>
</tr>
</tbody>
</table>
| **W-layer**  
*Peter does not love Mary.* | **W-layer**  
*Petr nemiluje Marii.* |  |
| **M-layer**  
Peter  
**NNP**  
do  
**VBZ**  
not  
**RB**  
love  
**VBD**  
Mary  
**NNP**  
| **M-layer**  
Petr  
**NNMS1**  
milovat  
**VB-S—3P-NA**  
Marie  
**NNFS4**  
|  |
| **A-layer**  
Peter  
Sb  
**AuxV**  
do  
**Neg**  
love  
**Pred**  
Mary  
**Obj**  
| **A-layer**  
Petr  
Sb  
milovat  
**Pred**  
Marie  
**Obj**  
|  |
| **T-layer**  
Peter  
**ACT**  
love  
**PRED**  
Mary  
**PAT**  
| **T-layer**  
Petr  
**ACT**  
milovat  
**PRED**  
Marie  
**PAT**  
|  |
Treex architecture

data units
Treex architecture
data units
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

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

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

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

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

**Treex Blocks**
- TagTNT
- ParseMST
- MarkPassives

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

**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

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

**Treex Blocks**
- TagTNT
- ParseMST
- MarkPassives

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

**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
Future plans

- CPAN release
  - Treex::Core done
  - Treex::EN soon
  - Treex::CS, Treex::DE, ...
- Lot of “invisible” work (testing)
- Manual, tutorial, FAQ, demos
- Adding new document readers more easily
- Integrating more tools, more languages
- Web services? Alternative parallelization?
- Your applications, your requests...

Tomáš Kraut
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.
Sériovou výrobu firma rozjela loni v listopadu.
Firma začala výrobu řady poslední listopad.
TrEd visualization

word alignment on the morphological layer
TrEd visualization

word alignment on the tectogrammatical layer
Tři utonuli jsou z Jeseníku nad Odrou na Novojičínsku a jedna žena utonula v Novém Jičíně, Žilině.
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/tectomt
Thank you

Treex is growing!

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