Introduction to Treex
Modular NLP Framework

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

CLARA Winter School
on New Developments in Computational Linguistics
February 16th 2012, Prague
Outline

- Motivation, Treex origin (TectoMT)
- Layers of language description
- Treex architecture
- Treex internals
- Overview of tools and applications
- 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 origin (TectoMT)

2005 (Zdeněk Žabokrtský)

NLP framework
TectoMT

MT system
TectoMT

lemmatization

tagging

parsing
Treex origin (TectoMT)

NLP framework

*TectoMT*

MT system

*TectoMT*

lemmatization
tagging

parsing

MT system

*TectoMT*

lemmatization
tagging

parsing

coreference

PEDT preprocessing

CzEng analysis
treebank conversions

named entity r.

alignment (word,tree)

SMT preproc.

etc.
Treex origin (TectoMT)

2005

NLP framework
*TectoMT*

MT system
*TectoMT*

... 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.

Now not only tectogrammatics and not only MT renamed
Treex origin (TectoMT)

2005                           ...                            2011

NLP framework

*TectoMT*

MT system

*TectoMT*

redesigned and reimplemented
easier to use
more flexible

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 origin (TectoMT)

- Redesigned and reimplemented
- Easier to use
- More flexible
- More langs

NLP framework: TectoMT

MT system: TectoMT

Languages:
- English
- Czech
- Russian
- Tamil
- Polish
- Esperanto
- Spanish
- 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.
Treex origin (TectoMT)

2005

NLP framework
TectoMT
MT system
TectoMT

multi-purpose
NLP framework
Treex

lemmatization
tagging
parsing

coreference
Czech

SMT preproc.
easier to use
easier to use
more flexible
more flexible
more langs

CzEng analysis
treebank conversions
alignment (word,tree)

redesigned and
reimplemented

2011

English
Czech
Russian
Tamil
Polish
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.
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 layer
    - analytical functions
  - morphological layer
    - tagger (Morce)
    - lemmatization
    - segmentation

**TRANSFER**
- query dictionary
- use HMTM

**SYNTHESIS**
- t-layer
  - fill morphological categories
  - impose agreement
  - add functional words
- a-layer
  - generate wordforms
- m-layer
  - concatenate
- w-layer

source language (English)  target language (Czech)
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` (→ 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) → output files

block 1 → block 2 → ... → block n

Java tool X → C++ tool Y

document writer
Treex architecture parallelization (using SGE cluster)

- **Input files**
  - Document reader
  - Block
  - In-memory document
  - Block

- **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
  • can be saved in plain text *.scen 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:
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, trans, 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

DOCUMENT

sentence 1

BUNDLE

Zone en_src

W-layer

Peter does not love Mary.

A-layer

Peter

Sb

NNP
does
AuxV

VBP
not

Neg

love

Pred

VBD

Mary

Obj

NNP

BUNDLE

Zone cs_src

W-layer

Petr nemiluje Marii.

A-layer

Petr

Sb

NNMS1

does

AuxV

VBP
not

Neg

love

Pred

VBD

Mary

Obj

NNP

T-layer

Peter

n:subj

sg,...

love

v:fin

neg:1,sim,...

Mary

n:obj

sg,...

T-layer

Peter

n:subj

sg,...

milovat

v:fin

neg:1,sim,...

Marie

n:4

sg,...

sentence 2

...
### Treex architecture data units

#### BUNDLE

<table>
<thead>
<tr>
<th>Zone en_src</th>
<th>Zone cs_src</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>W-layer</strong></td>
<td><strong>W-layer</strong></td>
</tr>
<tr>
<td><em>Peter does not love Mary.</em></td>
<td><em>Petr nemiluje Marii.</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>A-layer</strong></th>
<th><strong>A-layer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>Petr</td>
</tr>
<tr>
<td>NNP</td>
<td>NNMS1</td>
</tr>
<tr>
<td>does</td>
<td>nemiluje</td>
</tr>
<tr>
<td>AuxV</td>
<td>Pred</td>
</tr>
<tr>
<td>not</td>
<td>neg:1,sim,...</td>
</tr>
<tr>
<td>Neg</td>
<td>Obj</td>
</tr>
<tr>
<td>love</td>
<td>milovat</td>
</tr>
<tr>
<td>Pred</td>
<td>Obj</td>
</tr>
<tr>
<td>Mary</td>
<td>Marie</td>
</tr>
<tr>
<td>NNP</td>
<td>NNFS4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>T-layer</strong></th>
<th><strong>T-layer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>Petr</td>
</tr>
<tr>
<td>ACT</td>
<td>ACT</td>
</tr>
<tr>
<td>n:subj</td>
<td>n:subj</td>
</tr>
<tr>
<td>sg,...</td>
<td>sg,...</td>
</tr>
<tr>
<td>love</td>
<td>love</td>
</tr>
<tr>
<td>PRED</td>
<td>PRED</td>
</tr>
<tr>
<td>v:fin</td>
<td>v:fin</td>
</tr>
<tr>
<td>neg:1,sim,...</td>
<td>neg:1,sim,...</td>
</tr>
<tr>
<td>Mary</td>
<td>Marie</td>
</tr>
<tr>
<td>PAT</td>
<td>PAT</td>
</tr>
<tr>
<td>sg,...</td>
<td>sg,...</td>
</tr>
</tbody>
</table>

**DOCUMENT**

<table>
<thead>
<tr>
<th>sentence 1</th>
<th>sentence 2</th>
<th>...</th>
<th>sentence N</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BUNDLE</strong></td>
<td><strong>BUNDLE</strong></td>
<td></td>
<td><strong>BUNDLE</strong></td>
</tr>
<tr>
<td>Zone en_src</td>
<td>Zone cs_src</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>W-layer</strong></td>
<td><strong>W-layer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Peter does not love Mary.</em></td>
<td><em>Petr nemiluje Marii.</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A-layer</strong></td>
<td><strong>A-layer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peter</td>
<td>Petr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNP</td>
<td>NNMS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>does</td>
<td>nemiluje</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AuxV</td>
<td>Pred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td>neg:1,sim,...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neg</td>
<td>Obj</td>
<td></td>
<td></td>
</tr>
<tr>
<td>love</td>
<td>milovat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pred</td>
<td>Obj</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mary</td>
<td>Marie</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNP</td>
<td>NNFS4</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>T-layer</strong></td>
<td><strong>T-layer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peter</td>
<td>Petr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACT</td>
<td>ACT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n:subj</td>
<td>n:subj</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sg,...</td>
<td>sg,...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>love</td>
<td>love</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRED</td>
<td>PRED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v:fin</td>
<td>v:fin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>neg:1,sim,...</td>
<td>neg:1,sim,...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mary</td>
<td>Marie</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAT</td>
<td>PAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sg,...</td>
<td>sg,...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**form**

**lemma**

**afun (edge label)**

**morphological tag (PoS)**

**t_lemma**

**functor (semantic role)**

**formeme (morphosyntactic abstraction)**

**grammatemes for number, negation, tense**
Treex architecture
data units

DOCUMENT

sentence 1

BUNDLE

Zone en_src

W-layer

Peter does not love Mary.

A-layer

Peter
do
love
Mary

T-layer

Peter
love
Mary

sentence 2

BUNDLE

Zone cs_src

W-layer

Petr nemiluje Marii.

A-layer

Petr
nemiluje
Marii

T-layer

Petr
milovat
Marie

...
Treex architecture
data units

DOCUMENT

sentence 1

BUNDLE

Zone en_src

W-layer
Peter does not love Mary.

A-layer

T-layer

BUNDLE

Zone cs_trans

W-layer
Petr nemiluje Marii.

A-layer

T-layer

form

lemma

afun (edge label)

morphological tag (PoS)

t_lemma

functor (semantic role)

formeme (morphosyntactic abstraction)

grammatemes for number, negation, tense
Internals – Design decisions

- Perl (wrappers for binaries, Java,...)
- Linux (some applications platform-independent)
- OOP (Moose)
- Open source (dual GNU GPL & Perl Artistic)
- Neutral w.r.t. methodology (statistical, rule-based)
- Multilingual
- Open standards (Unicode, XML)
Treex Core Classes
• Treex::Core::Document
• Treex::Core::Node
• Treex::Core::Scenario
• Treex::Core::Block

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

Visualzation
• 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
(+ Makefiles etc.)

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

Treex Blocks

In-house Tools

Third-party Tools

Visualization

Readers and Writers

Download

Shared part

SVN

Versioned part
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.
Sériovou výrobu firma rozjecha loni v listopadu.
Firma začala výrobu řady poslední listopad.
TrEd visualization

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

Named entity: normalized name=Novojičínsko, type=gro (oblast - okolí města)
package Tutorial::Solution::Svo2Sov;

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