Treex: Modular NLP Framework

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

LATE Lunch, DFKI
November 4th 2010, Saarbrücken, Germany
Outline

• Motivation
• Layers of language description in PDT
• Treex architecture
• Treex internals
• Future plans
• Conclusion and examples
Motivation for creating Treex

Originally a framework for a linguistically motivated MT system

- called TectoMT (both the MT system and framework)
- deep syntactic (tectogrammatical) transfer
- started with English to Czech direction
- translation process divided to ~ 90 “blocks”
- combining statistical and rule-based blocks

Goals:

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

Now used for many other projects, not limited to MT nor tectogrammatics:

- automatic alignment & annotation of a parallel treebank (CzEng)
- support for manual annotations (PEDT)
- lemmatization, tagging, parsing
- named entity recognition, information retrieval, coreference
- preprocessing for phrase-base MT
  - change word order, append determiners to nouns,...
  - add deep-syntactic features as an input for factored translation
- conversions, evaluations, etc.
4 layers of language description implemented in Prague Dependency Treebank (PDT)

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

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

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

- **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

- **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)
  - ...

\[
\text{\includegraphics[width=\textwidth]{treetree.png}}
\]
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
  - conversion of the input to Treex internal format (XML)
  - possibly split into more files
  - applying a scenario to the files (loaded in memory)
  - conversion to the desired output format
Blocks can be easily substituted with an alternative solution.
Blocks can be easily substituted with an alternative solution.

Scenario A
Sentence_segmentation_simple
Penn_style_tokenization
TagMxPost
Lemmatize_mtree
McD_parser

Scenario B
Each_line_as_sentence
Tokenize_and_tag
Lemmatize_mtree
Malt_parser
Treex architecture
data units

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

• **Bundle**
  • corresponds to one sentence
  • “bundle of trees”

• **Tree**
  • direction (S=source, T=target)
  • language (Arabic, Czech, English, German,...)
  • layer of language description (M, A, T)
Treex architecture
data units

<table>
<thead>
<tr>
<th>English layers</th>
<th>Czech layers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SEnglishW</strong></td>
<td><strong>SCzechW</strong></td>
</tr>
<tr>
<td><em>Peter does not love Mary.</em></td>
<td><em>Petr nemiluje Marii.</em></td>
</tr>
<tr>
<td><strong>SEnglishM</strong></td>
<td><strong>SCzechM</strong></td>
</tr>
<tr>
<td><em>Peter do not love Mary.</em></td>
<td><em>Petr milovat Marie.</em></td>
</tr>
<tr>
<td><strong>SEnglishA</strong></td>
<td><strong>SCzechA</strong></td>
</tr>
<tr>
<td><em>Peter do not love Mary.</em></td>
<td><em>Petr milovat Marie.</em></td>
</tr>
<tr>
<td><strong>SEnglishT</strong></td>
<td><strong>SCzechT</strong></td>
</tr>
<tr>
<td><em>Peter love Mary.</em></td>
<td><em>Petr milovat Marie.</em></td>
</tr>
</tbody>
</table>

**BUNDLE**

**DOCUMENT**

sentence 1

sentence 2

...
Treex architecture
data units

DOCUMENT

sentence 1

<table>
<thead>
<tr>
<th>English layers</th>
<th>Czech layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEnglishW</td>
<td>SCzechW</td>
</tr>
<tr>
<td>Peter does not love Mary.</td>
<td>Petr nemiluje Marii.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEnglishM</th>
<th>SCzechM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter do not love Mary</td>
<td>Petr milovat Marie</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEnglishA</th>
<th>SCzechA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter do not love Mary</td>
<td>Petr milovat Marie</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEnglishT</th>
<th>SCzechT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter love Mary</td>
<td>Petr milovat Marie</td>
</tr>
</tbody>
</table>

sentence 2

...
### Treex architecture data units

#### BUNDLE

<table>
<thead>
<tr>
<th>English layers</th>
<th>Czech layers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SEnglishW</strong></td>
<td><strong>TCzechW</strong></td>
</tr>
<tr>
<td><em>Peter does not love Mary.</em></td>
<td><em>Petr nemiluje Marii.</em></td>
</tr>
<tr>
<td><strong>SEnglishM</strong></td>
<td><strong>TCzechM</strong></td>
</tr>
<tr>
<td><em>Peter</em> <em>do</em> <em>not</em> <em>love</em> <em>Mary.</em></td>
<td><em>Petr</em> <em>milovat</em> <em>Marie.</em></td>
</tr>
<tr>
<td><strong>SEnglishA</strong></td>
<td><strong>TCzechA</strong></td>
</tr>
<tr>
<td><em>Peter</em> <em>Sb</em> <em>AuxV</em> <em>Neg</em> <em>love</em> <em>Pred</em> <em>Mary</em> <em>Obj</em></td>
<td><em>Petr</em> <em>Sb</em> <em>milovat</em> <em>Pred</em> <em>Marie</em> <em>Obj</em></td>
</tr>
<tr>
<td><strong>SEnglishT</strong></td>
<td><strong>TCzechT</strong></td>
</tr>
<tr>
<td><em>Peter</em> <em>ACT</em> <em>love</em> <em>PRED</em> <em>Mary</em> <em>PAT</em></td>
<td><em>Petr</em> <em>ACT</em> <em>milovat</em> <em>PRED</em> <em>Marie</em> <em>PAT</em></td>
</tr>
</tbody>
</table>
### Treex architecture data units

**English layers**

<table>
<thead>
<tr>
<th>SEnglishW</th>
<th>Peter does not love Mary.</th>
</tr>
</thead>
</table>

**Czech layers**

<table>
<thead>
<tr>
<th>TCzechW</th>
<th>Petr nemiluje Marii.</th>
</tr>
</thead>
</table>

**BUNDLE**

<table>
<thead>
<tr>
<th>SEnglishM</th>
<th>Peter do n love Mary.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TCzechM</th>
<th>Petr milovat Marie</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SEnglishA</th>
<th>Peter do n love Mary.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TCzechA</th>
<th>Petr milovat Marie</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SEnglishT</th>
<th>Peter love Mary</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TCzechT</th>
<th>Petr milovat Marie</th>
</tr>
</thead>
</table>

**DOCUMENT**

- sentence 1
  - English layers
    - SEnglishW: Peter does not love Mary.
    - SEnglishM: Peter does not love Mary.
    - SEnglishA: Peter do n love Mary.
    - SEnglishT: Peter love Mary.
  - Czech layers
    - TCzechW: Petr nemiluje Marii.
    - TCzechM: Petr milovat Marie.
    - TCzechA: Petr milovat Marie.
    - TCzechT: Petr milovat Marie.

- sentence 2
  - BUNDLE

- ... sentence N
  - BUNDLE
Internals – Design decisions

- Perl (wrappers for binaries, Java,...)
- Linux (some applications platform-independent)
- OOP (ClassStd, 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

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
- Tree_tagger
- McD_parser
- Mark_passives

Applications
- scenarios
- + format conversions

Format Convertors (to & from tmt)
- plain text
- HTML & various XML
- corpora PDT, PennTB, EMILLE, PADT, CoNLL, vertical

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

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

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

Treex Blocks
- Tree_tagger
- McD_parser
- Mark_passives

Applications
- scenarios
- + format conversions

Format Convertors (to & from tmt)
- plain text
- HTML & various XML
- corpora PDT, PennTB, EMILLE, PADT, CoNLL, vertical

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

- Reimplementation of core components
- CPAN release
- Adding new languages more easily
- Improved parallelization support
- Faster code, smaller files, ...
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
TrEd visualization

named entities

Tři utonuli jsou z Jeseníku nad Odrrou na Novojičínsku a jedna žena utonula v Novém Jičíně, Žilině.
package Tutorial::SVO_to_SOV_solution;
use Moose;
extends 'Treex::Core::Block';

sub process_bundle { 
    my ( $self, $bundle ) = @_; 
    my $a_root = $bundle->get_tree('SEnglishA'); 

    foreach my $a_node ( $a_root->get_descendants() ) { 
        if ( $a_node->get_attr('m/tag') =~ /^V/ ) {  # verb found 
            foreach my $child ( $a_node->get_echildren() ) { 
                if ( $child->get_attr('afun') eq 'Obj' ) {  # object found 
                    # Move the object and its subtree so it precedes the verb 
                    $child->shift_before_node($a_node); 
                } 
            } 
        } 
    } 
} 
1;
Thank you

Cooperation is welcomed.

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