Treex NLP Framework

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

October 9th 2013, Prague, NIF Workshop
Outline

Applications

multi-purpose NLP framework
**Treex**

Architectures

Treebanks

Background
Applications

lemmatization
tagging
parsing
deep parsing

machine translation
TectoMT

online interface
Treex::Web

named entity r.
coreference
alignment
SMT preproc.

multi-purpose NLP framework
Treex
Applications: analysis tools

**lemmatization**
- base forms of words (linguistically adequate, finer than stemming)

**tagging**
- part-of-speech tags, morphological analysis, disambiguation

**parsing**
- dependency or constituency (phrase-based) trees
- tectogrammatical trees, meaning of a sentence

**deep parsing**
- tectogrammatical trees, meaning of a sentence

Languages supported:
- English
- Czech
- German
- Arabic
- Spanish
- French
- Tamil
- Russian
- Hindi
- Esperanto
- Polish
- Vietnamese
- Urdu
- Finish
- Latin
- Greek
- etc.
Applications: TectoMT

- linguistically motivated machine translation
- developed since 2005 at ÚFAL by Zdeněk Žabokrtský et al.
- so far only English to Czech
- WMT 2012: evaluated as equally good as Moses system
- WMT 2013: in combination with Moses better than Google Translate
- speed: 0.8 seconds per sentence (after loading models)
- parallelization support (SGE with 100 cores, 68 M sentences, 10 days)
Applications: TectoMT

- deep-syntactic (tectogrammatical) transfer
- translation process divided to more than 140 “blocks“
- combining statistical and rule based blocks

**ANALYSIS**
- tectogrammatical layer
  - fill formems
  - grammatemes
  - build t-tree
  - mark edges to contract
- analytical layer
  - analytical functions
  - parser (McDonald's MST)
- morphological layer
  - lemmatization
  - tokenization

**TRANSFER**
- query dictionary
- use HMTM

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

Source language (English) | Target language (Czech)

Applications: TectoMT
Applications: TectoMT

**MORPHOLOGY:**
- ResegmentSentences
- Tokenize
- NormalizeForms
- FixTokenization
- TagMorce
- FixTags
- Lemmatize

**NAMED ENTITIES:**
- StanfordNamedEntities
- DistinguishPersonalNames

**A-LAYER:**
- MarkChunks
- ParseMST
- SetIsMemberFromDeprel
- RehangConllToPdtStyle
- FixNominalGroups
- FixIsMember
- FixAtree
- FixMultiwordPrepAndConj
- FixDicendiVerbs
- SetAfunAuxCPCoord
- SetAfun

**TRANSFER:**
- CopyTtree
- TrLFPhrases
- TrLFJointStatic
- DeleteSuperfluousTnodes
- TrFTryRules
- TrFAddVariants
- TrFRelrank
- TrLTryRules
- TrLAddVariants
- TrLFNumeralsByRules
- TrLFilterAspect
- TransformPassiveConstructions
- PrunePersonalNameVariants
- RemoveUnpassivizableVariants
- TrLFCompounds
- CutVariants
- FixTransferChoices
- ReplaceVerbWithAdj
- DeletePossPronBeforeVlastni
- TrLFemaleSurnames
- AddNounGender
- MarkNewRelClauses
- AddRelpronBelowRc
- ChangeCorToPersPron
- AddPersPronBelowVfin
- AddVerbAspect
- FixDateTime
- FixGrammatemesAfterTransfer
- FixNegation
- MoveAdjsBeforeNouns
- MoveGenitivesRight
- MoveRelClauseRight
- MoveDicendiCloserToDsp
- MovePersPronNextToVerb
- MoveEnoughBeforeAdj
- MoveJesteBeforeVerb
- FixMoney
- OverridePpWithPhraseTr
- FindGramCorefForReflPron
- NeutPersPronGenderFromAntec
- ValencyRelatedRules
- SetClauseNumber
- TurnTextCorefToGramCoref

**SYNTHESIS TO A-LAYER:**
- CopyTtree
- DistinguishHomonymous
- ReverseNumberNounDep
- InitMorpcat
- FixPossessiveAdjs
- MarkSubject
- ImposePronZagr

**SYNTHESIS TO TEXT:**
- ConcatenateTokens
- ApplySubstitutions
- DetokenizeUsingRules
- RemoveRepeatedTokens
- NormalizePunctuationForWMT
Applications: Treex::Web

online interface
Applications: NER

- Support for nested named entities ("Jeseník nad Odrou": city vs. river)
- Hierarchical classes (g=geographical entity, gr=region, gu=city, gs=street,...)
Background

multi-purpose NLP framework

Treex

PDT layers of annotation

Prague Markup Language
PDT layers of language description
implemented in Prague Dependency Treebank

- **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
PDT layers of language description implemented in Prague Dependency Treebank

- **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)
  - ...

- PDT layers of language description implemented in Prague Dependency Treebank

[Diagram showing dependency tree structures]
layers of language description
implemented in Treex

- Mostly backward compatible adaptations (adding attributes)
  - formeme \((n:2, n:k+3, v:\text{že}+v\text{fin}, v:rc, \text{adj:attr})\)
  - attributes for clauses, is_passive \((\rightarrow \text{diathesis})\),...

- 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
Prague Markup Language

- developed since 2005, used for PDT 2.0 (2006)
- many similarities with NIF: universal NLP,...
- XML-based (no RDF), PML Schema (→ Relax NG)
- special support for trees and roles
- one schema for each layer of PDT, one for Treex
- used in TrEd (tree editor) and PML-TQ (tree query)
- much less popular than simple CoNLL format
- too universal, too complex, only Perl implementation
- not enough advertising?
- standard: format vs. API?
Treebanks

multi-purpose NLP framework

Treex

Czech-English deep treebank
CzEng

Czech-English deep treebank
PCEDT

treebanks for 30 languages
HamleDT
CzEng
(Czech-English treebank)

- automatically parsed deep parallel treebank
- 15 M sentences; 200 M tokens

with Treex
PCEDT
(Prague Czech-English Dependency Treebank)

- manually parsed deep parallel treebank
- 50 k sentences; 1.2 M tokens

pre-annotated with Treex
HamleDT

- 30 manually annotated dependency treebanks
- 30 languages; 400 k sentences; 6 M tokens
- harmonized into a common style (guidelines)

using Treex
Treex architecture

multi-purpose NLP framework

*Treex*

- motivation
- architecture
- internals
- examples
Motivation

Goals of Treex

- elegant integration of in-house and third-party NLP tools
- modularity, reusability, cooperation, efficient development
- ability to easily modify and add code in a full-fledged programming language (Perl)
- unified object-oriented interface for accessing data structures
My idea about NIF pipeline

- **input files**
- **NIF files**
- **tool 1**
- **tool n**

1. Parse NIF to NIF
2. Tool 1
3. Tool n
Treex architecture

- **input files**
- **document reader**
- **block 1**
- **block n**
- **In-memory document representation (OOP API)**
- **document writer**
- **output files**
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:
Blocks can be easily substituted with an alternative solution.

**Scenario A**

<table>
<thead>
<tr>
<th>W2A::EN::Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2A::EN::Tokenize</td>
</tr>
<tr>
<td>W2A::EN::TagMorce</td>
</tr>
<tr>
<td>W2A::EN::Lemmatize</td>
</tr>
<tr>
<td>W2A::EN::ParseMST</td>
</tr>
</tbody>
</table>

**Scenario B**

<table>
<thead>
<tr>
<th>W2A::SegmentOnNewlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2A::EN::TagLinguaEn</td>
</tr>
<tr>
<td>W2A::EN::Lemmatize</td>
</tr>
<tr>
<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

<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>M-layer</th>
<th>M-layer</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Peter do not love Mary.</em></td>
<td><em>Petr milovat Marie.</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A-layer</th>
<th>A-layer</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Peter do not love Mary.</em></td>
<td><em>Petr milovat Marie.</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T-layer</th>
<th>T-layer</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Peter love Mary.</em></td>
<td><em>Petr milovat Marie.</em></td>
</tr>
</tbody>
</table>

### Diagram

- **DOCUMENT**: sentence 1
- **BUNDLE**
- **ZONE**: en_src
- **W-layer**: *Peter does not love Mary.*
- **M-layer**: *Peter do not love Mary.*
- **A-layer**: *Peter do not love Mary.*
- **T-layer**: *Peter love Mary.*

- **DOCUMENT**: sentence 2
- **BUNDLE**
- **ZONE**: cs_src
- **W-layer**: *Petr nemiluje Marii.*
- **M-layer**: *Petr milovat Marie.*
- **A-layer**: *Petr milovat Marie.*
- **T-layer**: *Petr milovat Marie.*

- **DOCUMENT**: ... sentence N
- **BUNDLE**
- **ZONE**: ...
- **W-layer**: ...
- **M-layer**: ...
- **A-layer**: ...
- **T-layer**: ...
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

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

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

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

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

**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

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

**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
TrEd visualization

word alignment on the morphological layer
TrEd visualization

word alignment on the tectogrammatical layer
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;
}

1;

John loves Mary more than cherry. 
SUBJ VERB OBJECT

John  Mary   loves more than cherry. 
SUBJ OBJECT VERB

SVO → SOV
Thank you

Cooperation is welcomed.

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

http://ufal.mff.cuni.cz/treex
http://quest.ms.mff.cuni.cz/treex-web

http://ufal.mff.cuni.cz/tectomt
http://ufal.mff.cuni.cz/hamledt
http://ufal.mff.cuni.cz/czeng
http://ufal.mff.cuni.cz/pcedt2.0
http://ufal.mff.cuni.cz/jazz/PML
http://euler.ms.mff.cuni.cz/