Introduction to TectoMT

Zdeněk Žabokrtský, Martin Popel

Institute of Formal and Applied Linguistics
Charles University in Prague

CLARA Course on Treebank Annotation, December 2010, Prague
Outline

- PART 1
  - What is TectoMT?
  - TectoMT’s architecture
  - Overview of TectoMT’s tools and applications

- PART 2 - demo
What is TectoMT?

- multi-purpose NLP software framework
- created at UFAL since 2005

main linguistic features
- layered language representation
- linguistic data structures adopted from the Prague Dependency Treebank

main technical features
- highly modular, open-source
- numerous NLP tools already integrated (both existing and new)
- all tools communicating via a uniform OO infrastructure
- Linux + Perl
- reuse of PDT technology (tree editor TrEd, XML...)
Why “TectoMT”?

- **Tecto..**
  - refers the (Praguian) tectogrammar
  - deep-syntactic dependency-oriented sentence representation
  - developed by Petr Sgall and his colleagues since 1960s
  - large scale application in the Prague Dependency Treebank

- **.....MT**
  - the main application of TectoMT is Machine Translation

- however, not only “tecto” and not only “MT” !!!

- re-branding planned for 2011: TectoMT → Treex
What is not TectoMT?

- TectoMT (as a whole) is not an end-user application
  - it is rather an experimental lab for NLP researchers

- however, releasing of single-purpose stand-alone applications is possible
Motivation for creating TectoMT

- First, technical reasons:
  - Want to make use of more than two NLP tools in your experiment? Be ready for endless data conversions, need for other people's source code tweaking, incompatibility of source code and model versions...
  - ⇒ unified software infrastructure might help us in many aspects.

- Second, our long-term MT plan:
  - We believe that tectogrammar (deep syntax) as implemented in Prague Dependency Treebank might help to (1) reduce data sparseness, and (2) find and employ structural similarities revealed by tectogrammar even between typologically different languages.
Main Design Decisions

- Linux
- Perl as the core language

- set of well-defined, linguistically relevant layers of language representation

- neutral w.r.t. chosen methodology ("rules vs. statistics")

- emphasis on modularity
  - each task implemented by a sequence of blocks
  - each block corresponds to a well-defined NLP subtask
  - reusability and substitutability of blocks

- support for distributed processing
Data Flow Diagram in a typical application in TectoMT

INPUT DATA FILES → input format converter → MEMORY REPRESENTATION OF SENTENCE STRUCTURES → output format converter → OUTPUT DATA FILES

scenario:

block 1 → block 2 → block 3 → … → block n

non-Perl tool X → non-Perl tool Y
Hierarchy of data-structure units

- **document**
  - the smallest independently storable unit (~ xml file)
  - represents a text as a sequence of bundles, each representing one sentence (or sentence tuples in the case of parallel documents)

- **bundle**
  - set of tree representations of a given sentence

- **tree**
  - representation of a sentence on a given layer of linguistic description

- **node**

- **attribute**
  - document's, node's, or bundle's name-value pairs
Tree types adopted from PDT

- **tectogrammatical layer**
  - deep-syntactic dependency tree

- **analytical layer**
  - surface-syntactic dependency tree
  - 1 word (or punct.) ~ 1 node

- **morphological layer**
  - sequence of tokens with their lemmas and morphological tags
Trees in a bundle

- in each bundle, there can be at most one tree for each "layer"
- set of possible layers = \{S,T\} x \{English,Czech,...\} x \{M,A,T,P, N\}
- S - source, T-target (analysis vs. synthesis, MT perspective)
- M - morphological analysis
- P - phrase-structure tree
- A - analytical tree
- T - tectogrammatical tree
- N - instances of named entities

Example: SEnglishA - tectogrammatical analysis of an English sentence on the source-language side
Hierarchy of processing units

- **block**
  - the smallest individually executable unit
  - with well-defined input and output
  - block parametrization possible (e.g. model size choice)

- **scenario**
  - sequence of blocks, applied one after another on given documents

- **application**
  - typically 3 steps:
    - 1. conversion from the input format
    - 2. applying the scenario on the data
    - 3. conversion into the output format
Blocks

- technically, Perl classes derived from TectoMT::Block
- either method `process_bundle` (if sentences are processed independently) or method `process_document` must be defined
- several hundreds blocks in TectoMT now, for various purposes:
  - blocks for analysis/transfer/synthesis, e.g.
    - `SEnglishW_to_SEnglishM::Lemmatize_mtree`
    - `SEnglishP_to_SEnglishA::Mark_heads`
    - `TCzechT_to_TCzechA::Vocalize_prepositions`
  - blocks for alignment, evaluation, feature extraction, etc.
- some of them only implement simple rules, some of them call complex probabilistic tools
- English-Czech tecto-based translation currently composes of roughly 140 blocks
Tools available as TectoMT blocks

to integrate a stand-alone NLP tool into TectoMT means to provide it with the standardized block interface

already integrated tools:

- taggers
  - Hajič's tagger, Raab&Spoustová Morče tagger, Rathnaparkhi MXPOST tagger, Brants's TnT tager, Schmid's Tree tagger, Coburn's Lingua::EN::Tagger

- parsers
  - Collins' phrase structure parser, McDonalds dependency parser, Malt parser, ZŽ's dependency parser

- named-entity recognizer
  - Stanford Named Entity Recognizer, Kravalová's SVM-based NE recognizer

- miscel.
  - Klimeš's semantic role labeller, ZŽ's C5-based afun labeller, Ptáček's C5-based Czech preposition vocalizer, ...
Other TectoMT components

- "core" - Perl libraries forming the core of TectoMT infrastructure, esp. for memory representation of (and interface to) to the data structures
- numerous file-format converters (e.g. from PDT, Penn treebank, Czeng corpus, WMT shared task data etc. to our xml format)
- TectoMT-customized Pajas' tree editor TrEd
- tools for parallelized processing (Bojar)
- data, esp. trained models for the individual tools, morphological dictionaries, probabilistic translation dictionaries...
- tools for testing (regular daily tests), documentation...
Languages in TectoMT

- full-fledged sentence PDT-style analysis/transfer/synthesis for English and Czech
  - using state-of-the-art tools

- prototype implementations of PDT-style analyses for a number of other languages
  - mostly created by students
  - Polish, French, German, Tamil, Spanish, Esperanto...
English-Czech translation in TectoMT
English-Czech translation in TectoMT

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

**TRANSFER**
- t-layer
  - query
  - dictionary
  - use HMTM

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

**Rule based** & **Statistical** blocks
Real Translation Scenario

SEnglishW_to_SEnglishM::
Tokenization
Normalize_forms
Fix_tokenization
TagMorce
Fix_mtags
Lemmatize_mtree

SEnglishM_to_SEnglishN::
Stanford_named_entities
Distinguish_personal_names

SEnglishM_to_SEnglishA::
McD_parser
Fill_is_member_from_deprel
Fix_tags_after_parse
McD_parser REPARSE=1
Fill_is_member_from_deprel
Fix_McD_topology
Fix_nominal_groups
Fix_is_member
Fix_atree
Fix_multiword_prep_and_conj
Fix_dicendi_verbs
Fill_afun_AuxCP_Coord
Fill_afun

SEnglishT_to_TCzechT::
Clone_tree
Translate_LF_phrases
Translate_LF_joint_static
Delete_superfluous_nodes
Translate_F_try_rules
Translate_F_add_variants
Translate_F_rerank
Translate_L_try_rules
Translate_L_add_variants
Translate_LF_numerals_by_rules
Translate_L_filter_aspect
Transform_passive_constructions
Prune_personal_name_variants
Remove_unpassivizable_variants
Translate_LF_compounds
Cut_variants
Rehang_to_eff_parents
Rehang_shared_attr
Detect_formeme
Detect_voice
Fix_imperatives
Fill_is_name_of_person
Fill_gender_of_person
Add_cor_act
Add_adjectives
Add_relclause_to_postposit
Add_dicendi_closer_to_dsp
Add_PersPron_next_to_verb
Add_enough_before_adj
Fix_money
Recompute_deepord
Find_gram_coref_for_refl_pron
Neut_PersPron_gender_from_antec
Override_pp_with_phrase_translation
Valency_related_rules
Fill_clause_number
Move_text_coref_to_gram_coref

TCzechT_to_TCzechA::
Clone_atree
Distinguish_homonymous_mlemmas
Reverse_number_noun_dependency
Init_morphcat
Fix_possessive_adjectives
Mark_subject

TCzechA_to_TCzechW::
Concatenate_tokens
Ascii_quotes
Remove_repeated_tokens
Parallel analysis

- data needed for training the transfer phase models
- Czech-English parallel corpus CzEng
- 8 mil. pairs of sentences with automatic PDT-style analyses and alignment
Summary of Part I

- TectoMT (→Treex)
  - environment for NLP experiments
  - multipurpose, multilingual
  - PDT-style linguistic structures
  - Linux+Perl, open-source
  - modular architecture (several hundreds of modules)
  - capable of processing massive data
  - will be released at CPAN