An Open Architecture for Natural Language Processing/MT

Rajeev Sangal
Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
sangal@iiit.ac.in

10 March 2011
Outline

1. Technical design principles
2. System overview
   - Pipeline architecture
3. System design principles
4. SSF and Dashboard
System building principles:

2.1 Dependency structure - Not phrase structure
2.2 Hybrid processing - statistical and rule-based processing
2.3 Separating engines from data
2.4 Transfer vs. interlingua
2.5 Common representation - Shakti Standard Format (SSF)
1.1 Dependency Structure

- Bags - Un-analyzed chunks
  - Store local level analysis
- Dependency structure (at sentence level)
  - Build tree structure with Paninian relations
- Use feature structures to store information
  - Ex. morph features after morph analysis, word sense after WSD analysis, etc.
1.2 Hybrid Processing

Combines

- Rule-based approach. Example:
  - Transfer grammar rules
  - Rules for target language generation, etc.

- Statistical techniques. Example:
  - Part-of-speech (POS)
  - Word sense disambiguation (WSD), etc.
1.3 Separating Engines from Data

- Separate the engines from language data
  - Engines are programs, and are language independent
  - Data - language dependent
- Means different groups can work in parallel and prepare them
1.4 Transfer vs. Interlingua

Transfer approach, but

■ Transfer among group of languages
■ Common representation
  ■ Dependency trees with Paninian analysis
■ Multi-dict (like interlingua)
1.5 SSF - Shakti Standard Format

- Allows representation of sentences in the form of trees
- Each node in a tree can have features
- Multiple trees possible
- Provision for representation of discourse relations
2. System Overview – Pipeline architecture

Source Language (SL)

Web source
(doc, html,...)

Text Collector

Raw Text

Text–Cleaner

Text Pre–processor

Standard with Metadata

a.cml

Tokenizer

SSF
2. System Overview – Pipeline architecture

- Normalizer spelling Corrector
- Sandhi Splitter
- Morph Analyzer
- POS tagger
- Chunker
- Pruning
2. System Overview – Pipeline architecture

- Head computation module
- Vibhakti computation module
- Discoursing Processing
- Anaphora Resolver
- Discourse connective Handler
- Named Entity Recognizer
- Clause Boundary Identifier
- Parser (Simple/Full)

* a.hcm
* a.vcm

*Rajeev Sangal*

Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
sangal@iiit.ac.in

An Open Architecture for Natural Language Processing/MT
2. System Overview – Pipeline architecture

Lexical Sense Disambiguation

Syntax Transfer  Lexical Transfer  Transliteration

SL to TL transfer

Built Agreement Module
  Engine → Agreement Rules

Local Word Splitter

Word Generator
  Engine ← Morph data

Translation Delivery
  Target Language (TL)
3 Shakti System Design Principles

4.1 Modularity
4.2 Simplicity of organization
4.3 Robustness - Dealing with failure to analyze
4.4 Transparency
3.1 Modularity

- MT task broken into small sub-tasks
  - Each task linguistically meaningful and independent
- Currently has about 20 modules of which
  - Source language analysis - 12
  - Transfer grammar/lex component - 5
  - Target language generation - 3
3.2 Simplicity of organization

- Pipe-line flow
  - Although more complex data flow structure possible
- Common representation used by all modules
  - Shakti Standard Format (SSF)
3.3 Robust: Designed to deal with failure

- If a module fails to perform its analysis, next module operates on partial analysis
- A module can deal with 2-3 levels of analysis (not yet implemented inside modules)
  - If a more detailed level of analysis not available, works at less detailed level
- Shakti standard format (SSF) allows seamless shifting between levels (design of SSF crucial for good design)
3.4 Transparency

- Developer friendly
  - Inputs and outputs of all modules available readily
  - Great for debugging
  - Standard readable textual representation
  - Profiling for making system faster
The following sentence:

- Children are watching some programmes on television in the house.

contains the chunks (enclosed by double brackets),

- ((Children)) [[are watching]] ((some programmes)) ((on television)) ((in the house))

All the chunks are noun phrases, except for one ['are watching'] which is a verb group.
Mark the part-of-speech tag for each word

- ((Children_NNS)) [[are_VBP watching_VBG]] ((some_DT programmes_NNS)) ((on_IN television_NN)) ((in_IN the_DT house_NN))
### SSF: Example

<table>
<thead>
<tr>
<th>Addr</th>
<th>Lex</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((</td>
<td>NP</td>
</tr>
<tr>
<td>1.1</td>
<td>children</td>
<td>NNS</td>
</tr>
<tr>
<td>2</td>
<td>((</td>
<td>VG</td>
</tr>
<tr>
<td>2.1</td>
<td>are</td>
<td>VBP</td>
</tr>
<tr>
<td>2.2</td>
<td>watching</td>
<td>VBG</td>
</tr>
<tr>
<td>3</td>
<td>((</td>
<td>NP</td>
</tr>
<tr>
<td>3.1</td>
<td>some</td>
<td>DT</td>
</tr>
<tr>
<td>3.2</td>
<td>programmes</td>
<td>NNS</td>
</tr>
<tr>
<td>4</td>
<td>((</td>
<td>PP</td>
</tr>
<tr>
<td>4.1</td>
<td>on</td>
<td>IN</td>
</tr>
<tr>
<td>4.1.1</td>
<td>((</td>
<td>NP</td>
</tr>
<tr>
<td>4.1.1.2</td>
<td>television</td>
<td>NN</td>
</tr>
</tbody>
</table>
### SSF: Example - Morph Features

<table>
<thead>
<tr>
<th>Addr</th>
<th>Lex</th>
<th>Category</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>children</td>
<td>NNS</td>
<td>&lt;fs af=child,n,m,p,3,0,,&gt;</td>
</tr>
<tr>
<td>2</td>
<td>((</td>
<td>VG</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>are</td>
<td>VBP</td>
<td>&lt;fs af=be,v,m,p,3,0,,&gt;</td>
</tr>
<tr>
<td>2.2</td>
<td>watching</td>
<td>VBG</td>
<td>&lt;fs af='watch,v,m,s,3,0,,' aspect=PROG&gt;</td>
</tr>
<tr>
<td>3</td>
<td>((</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>some</td>
<td>DT</td>
<td>&lt;fs af=some,det,m,s,3,0,,&gt;</td>
</tr>
<tr>
<td>3.2</td>
<td>programmes</td>
<td>NNS</td>
<td>&lt;fs af=programme,n,m,p,3,0,,&gt;</td>
</tr>
<tr>
<td>4</td>
<td>((</td>
<td>PP</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>on</td>
<td>IN</td>
<td>&lt;fs af=on,p,m,s,3,0,,&gt;</td>
</tr>
<tr>
<td>4.1.1</td>
<td></td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>4.1.2</td>
<td>television</td>
<td>NN</td>
<td>&lt;fs af=television,n,m,s,3,0,,&gt;</td>
</tr>
</tbody>
</table>
He gave me a book in the garden this morning.

[gave]
  |   
  |   
  |   

[he]    [book]    [me]

Bags shown with head enclosed in square brackets.
He gave me a book in the garden this morning.

The dependency relations are shown as follows:

- **he** role=\text{subj}:\text{give} \quad drel='k1:give'
- **gave**
- **me** role=\text{obj}:\text{give} \quad drel='k4:give'
- **book** role=\text{obj2}:\text{give} \quad drel='k2:give'
- **garden** role=\text{prep\_in}:\text{give} \quad drel='k7s:give'
- **this\_morning** role=\text{prep\_r\_adv}:\text{give} \quad drel='k7t:give'
Conclusions

1. Technical design principles
   2.1 Dependency structure - Not phrase structure
   2.2 Hybrid processing - statistical and rule-based processing
   2.3 Separating engines from data
   2.4 Transfer vs. interlingua
   2.5 Common representation - Shakti Standard Format (SSF)

2. System overview

3. System design principles
   4.1 Modularity
   4.2 Simplicity of organization
   4.3 Robustness - Dealing with failure to analyze
   4.4 Transparency

4. Shakti Standard Format - Powerful representation scheme