Vincent Kríž & Barbora Hladká

RExtractor: a Robust Information Extractor

Recent developments in natural language processing and corpus Linguistics, 11.9.2015
MFF UK

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University in Prague
Czech Republic

kriz@ufal.mff.cuni.cz
http://ufal.mff.cuni.cz/~kriz
Motivation

- large collections of documents
- efficient browsing & querying
- typical approaches
  - full-text search
  - metadata search
- semantic interpretation of documents → suitable DB & query language → user-friendly browsing & querying
Case study on legislative domain

Legal texts
- specialized texts operating in legal settings
- they should transmit legal norms to their recipients
- they need to be clear, explicit and precise

Sentences
- simple sentences are very rare
- usually long and very complex

Legal texts are “generally considered very difficult to read and understand” (Tiersma, 2010)
Scenario

- **Cooperation between**
  - Information Extraction
  - Semantic Web
Scenario

• Extracting knowledge base
  – set of entities and relations between them
  – linguistic analysis (RExtractor)

• Knowledge base representation
  – Linked Data Principles
  – Resource Description Framework (RDF)
Scenario

• Extracting knowledge base
  – set of entities and relations between them
  – linguistic analysis (RExtractor)

• Knowledge base representation
  – Linked Data Principles
  – Resource Description Framework (RDF)
RExtractor Architecture

Conversion Component
- HTML
- TXT
- PDF
  → XML

NLP Component
- Segmentation
- Tokenization
- Tagger
- Morphology
- Parser

Relation Extraction
- PML-TQ
- Database of queries

Entity Detection
- PML-TQ
- Database of entities
RExtractor Architecture
RExtractor Architecture
NLP Component

- **Prague Dependency Treebank** framework
  - [http://ufal.mff.cuni.cz/pdt3.0](http://ufal.mff.cuni.cz/pdt3.0)

- **Tools**
  - segmentation & tokenization
  - lemmatization & morphology
  - syntactic parsing
Accounting units create fixed items and reserves according to special legal regulations.
RExtractor Architecture
Entity Detection Component

- **Database of Entities**
  - entities specified by domain experts

  - tree queries better than regular expressions
    - coordination
    - several word forms in inflective languages
Accounting units create fixed items and reserves according to special legal regulations.
Accounting units create fixed items and reserves according to special legal regulations.
RExtractor Architecture

Conversion Component
- HTML
- TXT
- PDF

NLP Component
- Segmentation
- Tokenization
- Tagger
- Morphology
- Parser

Relation Extraction
- PML-TQ
- Database of queries

Entity detection
- PML-TQ
- Database of entities
Relation Extraction Component

- **Database of Queries**
  - queries formulated by domain experts
  - their formulation in the form of PML-TQ queries on dependency trees

- **RDF ready output**
  - triples \((subject, predicate, object)\)
  - each position
    - is annotated in a text \((text \ chunk)\)
    - has a specific **ontological concept** \((RDF \ Class)\)
Relation Extraction Component

- Accounting units' obligations

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>hasToCreate</td>
<td>Something</td>
</tr>
<tr>
<td>Accounting</td>
<td>create</td>
<td>fixed items</td>
</tr>
<tr>
<td>units</td>
<td>create</td>
<td>reserves</td>
</tr>
<tr>
<td>Accounting</td>
<td>create</td>
<td>reserves</td>
</tr>
<tr>
<td>units</td>
<td>according to</td>
<td>regulations</td>
</tr>
<tr>
<td></td>
<td>legal Atr</td>
<td>special Atr</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>units</td>
<td>and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coord</td>
<td></td>
</tr>
</tbody>
</table>

Subject: Accounting units
Predicate: create
Object: fixed items

Predicate: create
Object: reserves

Predicate: according to
Object: regulations

Predicate: hasToCreate
Object: Something

Predicate: and
Object: Coord
The proposal for the entry into the register shall be submitted by the operator.
Relation Extraction Component

Types of relations

- **Definitions**
  - entities are defined or explained

- **Obligations**
  - an entity is obligated to do something

- **Rights**
  - an entity has right to do something
Relation Extraction Component

Results

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>R</th>
<th>O</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td># of queries</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>Goldstandard</td>
<td>97</td>
<td>308</td>
<td>62</td>
<td>467</td>
</tr>
<tr>
<td>Extracted</td>
<td>70</td>
<td>255</td>
<td>41</td>
<td>366</td>
</tr>
<tr>
<td>True positive</td>
<td>53</td>
<td>206</td>
<td>36</td>
<td>295</td>
</tr>
<tr>
<td>False negative</td>
<td>44</td>
<td>102</td>
<td>26</td>
<td>172</td>
</tr>
<tr>
<td>False positive</td>
<td>17</td>
<td>49</td>
<td>5</td>
<td>71</td>
</tr>
<tr>
<td>Precision (%)</td>
<td>75.7</td>
<td>80.8</td>
<td>87.8</td>
<td>80.6</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>54.6</td>
<td>66.9</td>
<td>58.1</td>
<td>63.2</td>
</tr>
</tbody>
</table>
Relation Extraction Component

Error analysis

<table>
<thead>
<tr>
<th>Error</th>
<th># of errors</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser</td>
<td>145</td>
<td>59.7%</td>
</tr>
<tr>
<td>Query</td>
<td>93</td>
<td>38.3%</td>
</tr>
<tr>
<td>Entity</td>
<td>5</td>
<td>2.1%</td>
</tr>
</tbody>
</table>

Results

- errors in automatic parsing
- query design
Conclusion

- general pipeline for **extraction** and **representation** of information that is presented in raw texts
  - processes input texts by linguistically-aware tools
  - extracts entities and relations from dependency trees
    - Linked Data principles
- **Legal documents** as a pilot domain