ElkfEd/IDC alias BART

Yannick Versley  Simone Ponzetto
Jason Smith  Vladimir Eidelman  Alan Jern
Massimo Poesio  Xiaofeng Yang  Alessandro Moschitti
Introduction

- Modularized and revamped version of EML-R system
- Full coreference including mention detection
- Open Source release planned for December
- Useful Java-only subset (e.g. for teaching)
Components

- Preprocessing
  aggregate data in MMAX2 annotation layers
- Mention extraction
  create markables from chunks/NEs
- Extract information about mentions
  mention type, semantic class ...
- Encode coref into classifier decisions and extract features
Components

- Preprocessing: aggregate data in MMAX2 annotation layers
- Mention extraction: create markables from chunks/NEs
- Extract information about mentions: mention type, semantic class ...
- Encode coref into classifier decisions and extract features

Modular pipeline architecture
Standoff annotation
Components

- Preprocessing
  *aggregate data in MMAX2 annotation layers*

- Mention extraction
  *create markables from chunks/NEs*

- Extract information about mentions
  *mention type, semantic class ...*

- Encode coref into classifier decisions
  extract features

Feature set and learners
(and learner settings!)
described in XML file
Machine Learning Infrastructure

- WEKA Machine Learning Toolkit
  - C4.5, RIPPER, other learning modules
- SVMlight-TK
  - SVMs, different kernels (linear, polynomial, etc.)
  - Tree kernels
  - Custom kernels
- MaxEnt-based classification and ranking
  - automatic feature combination
  - ranking: find best among a set of candidates
Flexibility in Preprocessing

- Chunker (YamCha) vs. Parser (Charniak)
- Stanford NER vs. Carafe
- Mentions from Chunks/NEs vs. Mentions from Mention Tagger (Carafe)
Resolution Algorithms

- Currently there:
  - Closest-first decoding (Soon et al., 2001)
  - Separate classifiers for pronouns/non-pronouns
  - Ranking-based resolution
  - Stacking ranking+classifier

- Wanted:
  - Classification + ranking
  - Global models
    (Daume&Marcu 2005, Culotta et al 2007, ...)

Quantitative / Qualitative Evaluation

- MUC scoring (per document, in total)
- Link-based scores, by type of anaphora (pronouns, appositions / copula, names, nominals)
- Qualitative evaluation inspect results in MMAX2
### Using different classifiers (MUC7)

<table>
<thead>
<tr>
<th>Learner</th>
<th>Recall</th>
<th>Precision</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>J48</td>
<td>53.3</td>
<td>70.3</td>
<td>60.6</td>
</tr>
<tr>
<td>SVMlight (linear)</td>
<td>48.4</td>
<td>72.0</td>
<td>57.9</td>
</tr>
<tr>
<td>MaxEnt (plain)</td>
<td>49.4</td>
<td>70.4</td>
<td>58.0</td>
</tr>
<tr>
<td>SVMlight (polynomial d=2)</td>
<td>51.9</td>
<td>70.9</td>
<td>59.9</td>
</tr>
<tr>
<td>MaxEnt (combination d=2)</td>
<td>53.1</td>
<td>69.4</td>
<td>60.2</td>
</tr>
</tbody>
</table>

- J48 easiest to use, but MaxEnt/SVMlight allow for greater choice of features
- Feature combination helps performance
- MaxEnt faster than SVMlight (332sec. vs. 500sec. testing time) but cannot use tree kernels
### Improving Preprocessing (ACE 2002 Bnews)

<table>
<thead>
<tr>
<th></th>
<th>Recl</th>
<th>Prec</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Feature set</td>
<td>59.4</td>
<td>52.2</td>
<td>55.6</td>
</tr>
<tr>
<td>Improved preprocessing</td>
<td>53.7</td>
<td>65.7</td>
<td>59.1</td>
</tr>
<tr>
<td>Extended feature set</td>
<td>56.1</td>
<td>76.3</td>
<td>64.7</td>
</tr>
</tbody>
</table>

- Extended feature set: uses more syntactic information (tree kernels, salience) and world knowledge
- Eliminating preprocessing errors is important
- Depending on the corpus/annotation guidelines, corpus-specific preprocessing can be necessary.
Modifying BART

- Trying out new Feature Sets
- Adding new Feature Extractors
- Adding new Preprocessing Components
- Different Resolution Methods
Inside BART: Preprocessing

- Documents are represented as instances of `MMAX2Discourse` (wrapped by `elkfed.mmax.Document`)
- Preprocessing information is stored on separate markable levels: Sentence, POS, Lemma, Chunk, NEs, Markable
- Chunk and NE markables are used to create markables on the Markable layer, then POS and Lemma info is added for the words
Inside BART: Preprocessing

- Pipeline components inherit from `PipelineComponent` or one of its descendants (`Parser`, `Tagger`)

- `AnnotationProcessor` or `TrainerProcessor` invokes the pipeline, which in turn calls the preprocessing components.
Inside BART: Mentions

Unannotated Text → ACE Mention Tagger → Merger → Preprocessing

Mention (with basic properties):
- Number
- Gender
- Mention Type
- Modifiers

Basic features
Syntactic features
Knowledge-based features

Decoder

SVM Classifier

Coreference Chains
Inside BART: Mentions

- A `MentionFactory` creates instances of the `Mention` class by:
  - Taking information from the Markable layer (each Markable markable becomes one mention)
  - Taking parses from the Parse annotation layer
  - Linking Mentions to their Utterance

- `Mention` is responsible for:
  - Keeping track of important information about the markable (gender, number, NE/not NE)
  - Linking to a node in the parse tree (no node will be present if we use a chunker)
Inside BART: Features
Inside BART: Features

- **PairInstance**
  - contains references to antecedent and anaphor Mentions
  - contains features of that pair (string match, alias, ...)

- **PairFeatureExtractor**
  - extracts one or several features
  - `describeFeatures`: give list of features and their type (`FeatureDescription`)
  - `extractFeatures`: takes a `PairInstance` and adds the features
Inside BART: Learning

- Learning is done by CorefTrainer subclasses, Annotation by CorefDecoder.
- CorefTrainer gets handed the mentions in the document and writes training examples to one or more InstanceWriter instances.
- CorefDecoder constructs testing examples and hands them to an OfflineClassifier constructed from the training examples. It returns a partition of Mentions.
<coref-experiment>
  <system type="soon">
    <classifiers>
      <classifier type="weka" model="idc0"
                  learner="weka.classifiers.trees.J48"
                  options=""/>
    </classifiers>
    <extractors>
      <!-- feature extractors -->
      <extractor name="FE_MentionType"/>
      ...
      <extractor name="FE_SentenceDistance"/>
    </extractors>
  </system>
</coref-experiment>