For this lab

- Download BART:
  - http://www.bart-coref.org , get the tarball
- Download data:
  - http://www.bart-coref.org/labs, get `lab_data_130521.zip`

BART: a Toolkit for Multilingual Anaphora Resolution

Olga Uryupina
CiMeC, University of Trento

Massimo Poesio
University of Essex

21.05.2013

Outline

- Coreference with BART
- First experiments: BART as a black box
- MMAX2 and BART
- Re-training your model
- Preprocessing Pipelines
  - Preprocessing for English
  - Preprocessing for other languages
- Language Plugin
- Developing new features and models
- Other formats

Coreference Resolution

One reason **Lockheed Martin Corp.** did not announce a full acquisition of Loral Corp. on Monday, according to Bernard Schwartz, Loral's chairman, was that **Lockheed** could not meet the price he had placed on Loral's 31 percent ownership of Globalstar Telecommunications Ltd.

Coreference Resolution

One reason Lockheed Martin Corp. did not announce a full acquisition of **Loral Corp.** on Monday, according to Bernard Schwartz, **Loral**'s chairman, was that Lockheed could not meet the price he had placed on **Loral**'s 31 percent ownership of Globalstar Telecommunications Ltd.

Coreference Resolution

One reason Lockheed Martin Corp. did not announce a full acquisition of Loral Corp. on Monday, according to **Bernard Schwartz**, Loral's **chairman**, was that Lockheed could not meet the price **he** had placed on Loral's 31 percent ownership of Globalstar Telecommunications Ltd.

Coreference Resolution

Goal: identify “entities” or “chains”, IDENTITY

- {Lockheed Martin Corp., Lockheed}
- {Loral Corp., Loral, Loral}
- {Bernard Schwartz, chairman, he}
- {Monday}
- ..

Input: raw text + set of “mentions” (“markables”)

Prerequisites

- Set of mentions – Mention Detector needed
- Linguistic Information – different layers of linguistic knowledge (PoS, morphology, parse trees, semantic labels,..)

Why do we need it?

- Information Extraction
- IR/QA
- Machine Translation
- Summarization
- Reverse task – generating anaphoric expression, e.g. to improve coherence

Directions for research in the field

- Feature engineering – improving linguistic representation of the problem
- Modeling
- New languages

All these things can be done with BART – and you don't need to concentrate on the rest!

BART

Baltimore Anaphora Resolution Toolkit

- Prototype: Ponzetto & Strube (2006)
- 1st version: John Hopkins Workshop (2007)
- Current state: multiple features, several models, 3 languages supported, different input/output formats, scoring metrics
- Evalita-2009 – the only system able to perform CR in Italian
- SemEval-2010 – state-of-the-art performance for En, It; best for De
- CoNLL-2011 – two groups participating with BART, both show results within the top cluster

BART's strong sides

- Modularity
- LanguagePlugin
- Extensive scoring/testing facilities
- Supports various input formats (including raw text – can be run from scratch)
- A lot of solutions already implemented

Getting started with BART

- Versley Y., Ponzetto S., Poesio M., Eidelman V., Jern A., Smith J., Yang X. and Moschitti A. (2008) BART: A modular toolkit for coreference resolution. ACL-08 demo.
- http://www.bart-coref.org
- Contact CiMeC group: Massimo Poesio, Olga Uryupina

First steps with BART

If you just need coreference..

- Go to bart-coref.org
- Download the treeball
- Unpack it
- Launch the BART server
- Send requests to process your documents
    -

First steps with BART: details

- Launch the BART server

```
(1a) source setup.sh (Linux, Mac)
(1b) set CLASSPATH=.;dist/BART.jar;libs/* (W)
(2) java -Xmx1024m elkfed.webdemo.BARTServer
```

- Send requests to process your documents
  - Web demo: in your browser, go to

```
http://localhost:8125/index.jsp
```

First steps with BART: details

- POST: in your terminal, prepare text1.txt, try:

```
cat text1.txt | POST
http://localhost:8125/BARTDemo/ShowText/process/
```

- “no POST” error?

```
lwp-request -m POST -c 'text/html;
charset=ISO8859-15'
http://localhost:8125/BARTDemo/ShowText/process/
txt/tab  < test1.txt
```

What does it do?!

It runs a precompiled coreference model:

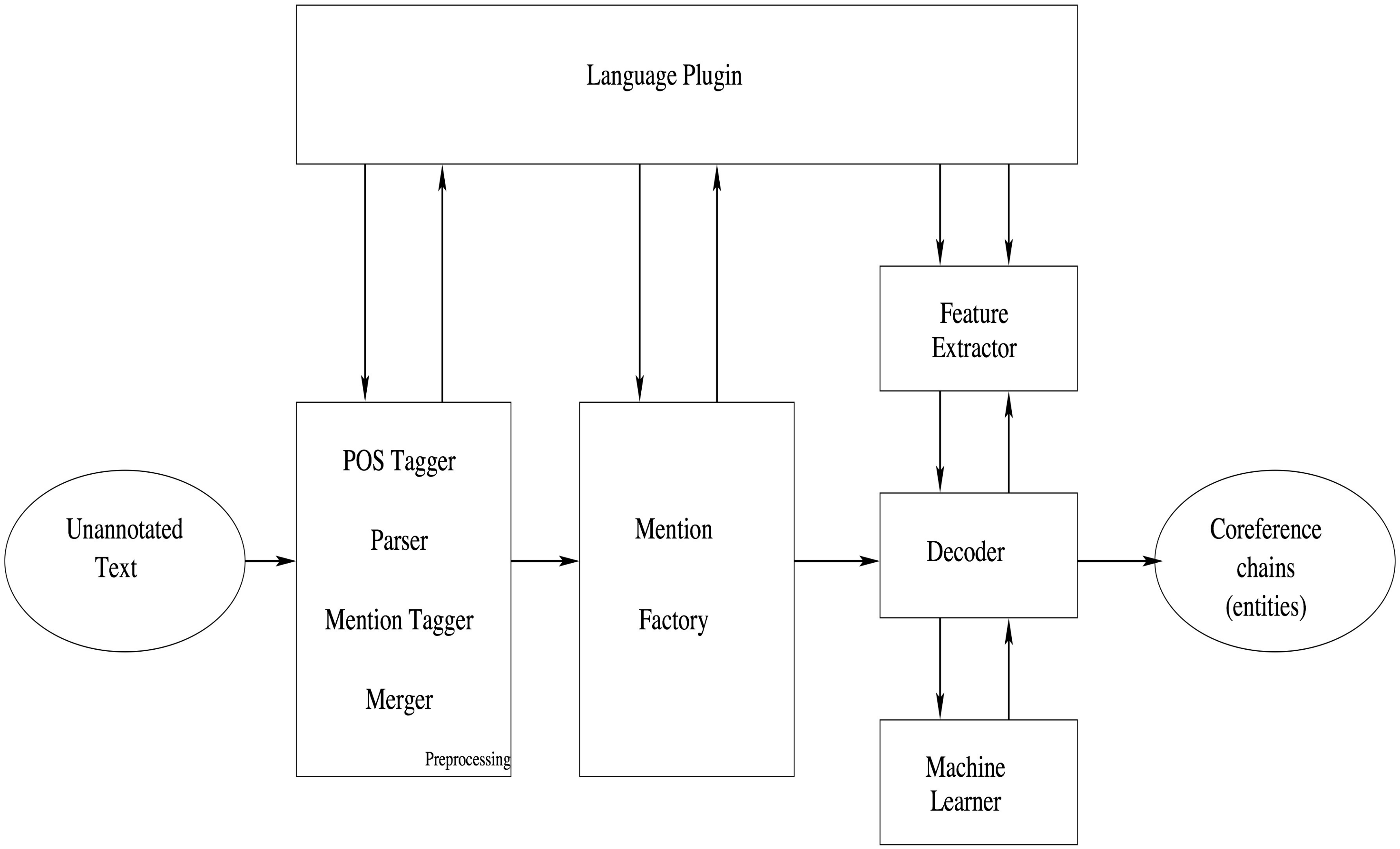
- Default setting for preprocessing: Berkley parser + Stanford NER => markables
- Default setting for coreference: MaxEnt, mention-pair, sort of Soon et al (2001) features

Intermediate steps:

```
webdemo_temp/Basedata
webdemo_temp/markables
```

Inside the black box

- Preprocessing
- Coreference Resolution
- Re-formatting


Language Plugin
Unannotated Text
POS Tagger
Parser
Mention Tagger
Merger
Preprocessing
Mention Factory
Feature Extractor
Decoder
Machine Learner
Coreference chains (entities)

BART internal representation

- MMAX2 format
- Each document is represented as a collection of XML “levels”:

  - sentence

  - PoS

  - lemma

  - chunk

  - ..
- Mention extraction – create the “markable” level

Internal representation – Example

- example_words.xml:

```
<?xml version="1.0"?>

<!DOCTYPE words SYSTEM "words.dtd">

<words>

<word id="word_1">CNN19981215</word>

<word id="word_2">.2130.0051</word>

<word id="word_3">NEWS</word>

<word id="word_4">STORY</word>

<word id="word_5">12/15/1998</word>

<word id="word_6">21:30:51.39</word>

<word id="word_7">The</word>

<word id="word_8">U.N.</word>

<word id="word_9">chief</word>
```

Internal representation – Example

- example_sentence_level.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE markables SYSTEM "markables.dtd">

<markables xmlns="www.eml.org/NameSpaces/sentence">

<markable id="markable_1348" span="word_130..word_137" orderid="7"  imported="false"
mmax_level="sentence" />

<markable id="markable_1341" span="word_1..word_6" orderid="0"  imported="false"
mmax_level="sentence" />

<markable id="markable_1346" span="word_83..word_98" orderid="5"  imported="false"
mmax_level="sentence" />
```

Internal representation – Example

- example_parse_level.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE markables SYSTEM "markables.dtd">

<markables xmlns="www.eml.org/NameSpaces/parse">

<markable id="markable_58" span="word_99..word_129" tag="(S1 (S (PP (IN In) (NP (DT the) (NN
report))) (, ,) (NP (NNP Butler)) (VP (VBZ says) (SBAR (S (NP (NP (NNP Iraq) (POS 's)) (NN conduct)) (VP
(VBD ensured) (SBAR (IN that) (S (NP (DT no) (NN progress)) (VP (AUX was) (ADJP (JJ able) (S (VP (TO
to) (VP (AUX be) (VP (VBN made) (PP (IN in) (NP (NP (DT the) (NNS fields)) (PP (IN of) (NP (NN
disarmament) (CC or) (NN accounting))))) (PP (IN for) (NP (NP (NNP Iraq) (POS 's)) (VBN prohibited)
(NNS weapons)))))))))))))))) (. .)))"  mmax_level="parse" />
```

Internal representation – Example

- example_markable_level.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE markables SYSTEM "markables.dtd">

<markables xmlns="www.eml.org/NameSpaces/parse">

<markable id="markable_0" span="word_8" wiki1="United_Nations" sentenceid="1"
wiki2="United_Nations" label="organization" min_ids="word_8" type="enamex" lemmata="u.n."
isprenominal="false" pos="nnp" mmax_level="markable" />

<markable id="markable_1" span="word_8..word_11" wiki1="Inspector" sentenceid="1" wiki2="Inspector"
label="person" min_ids="word_11" type="chunk" lemmata="u.n. chief weapon inspector"
isprenominal="false" pos="nnp nn nns nn" mmax_level="markable" />

<markable id="markable_2" span="word_17..word_18" wiki1="United_Nations" sentenceid="1"
wiki2="United_Nations" label="organization" min_ids="word_18" type="enamex" lemmata="the u.n."
isprenominal="false" pos="dt nnp" mmax_level="markable" />
```

Running BART on MMAX2

- Check your configuration:

```
config/config.properties
```

- Just launch..

  Full circle

```
java -Xmx4000m
  -Delkfed.corpus=YOURCORPUS
  elkfed.main.XMLExperiment
```

  Just applying the model

```
java -Xmx4000m
  -Delkfed.corpus=YOURCORPUS
  elkfed.main.XMLAnnotator
```

Configuring BART

config/config.properties

- Set paths to your training/testing data
- Set language (default=english)
- Set runPipeline and Pipeline parameters
- Many other parameters

What does it do?

Using the default settings (cf. config):

- prepares the training data,
- learns classifier(s),
- runs on the test data, output “response”
- evaluates

Evaluation results: STDIN

Response: adding response level to MMAX2 test

Re-train your model with new settings

You can specify the experimental settings in your exp xml:

```
java -Xmx4000m -Delkfed.corpus=YOURCORPUS
elkfed.main.XMLExperiment YOURSETTINGS.xml
```

Experimental settings: models

- Mention-pair model (Soon et al.)
- Splitting
- Mention-ranking model
- Entity-mention models (“Semantic trees”) – under development
- Dummy (for debugging/development) – BART 2.0

Experimental settings: ML

- SVM: SVMLight, needs a special API for a realistic speed (not distributed with BART)
- MaxEnt: in-house implementation; parameters: feature combination count (1,2,3) and a regularization parameter
- WeKa: interface to the WeKa package, standard parameters

Experimental settings: features

- Total: around 45 “feature extractors” (BART2.0)
- Each feature extractor corresponds to a set of related features
- Features for each split type etc can (and should) be specified separately

Experimental settings: presets

- Examples for different pre-sets:

  ```
  src/elkfed/main/*xml
  ```

- Examples for different baselines (soon, all, ng; maxent, weka, svm; ranking) – BART 2.0

Evaluation

- Internal scoring
  - MUC
  - Anaphor-based
  - CEAF*2 (BART 2.0, requires lp_solve)
- External scoring
  - CoNLL scorer
  - Modified BIO-CoNLL scorer (BART 2.0)

Preprocessing

- What if you don't have all the MMAX levels?
  - English (available within the distribution)
  - German (needs external components)
  - Italian (needs external components)

Preprocessing: English

- Raw text: go for the “black-box” scenario:
  - Create an MMAX project with the web demo
  - Store it somewhere
  - Test different models
- Incomplete MMAX (at least tokens)
  - Run a “Preprocessing Pipeline”
- XML annotation (MAS-XML) – BART 2.0
- CoNLL format – BART 2.0

Preprocessing pipelines

- Input: tokens + ?
- Output: missing levels + set of markables =

  MMAX project ready for a BART XMLExperiment

Preprocessing Pipelines

- Chunker pipeline (for basic Nps):
  - run YamCha + Stanford Pos
  - run (Stanford) NER
  - merge the outputs to create markables
- Parser pipeline (for basic NPs):
  - run (Berkley) parser
  - run (Stanford) NER
  - merge the outputs to create markables

Preprocessing Pipelines

- CARAFE pipeline (for external EMD, including gold), BART 2.0
  - store precompiles mention boundaries on the "enamex" level
  - inherit markables from enamex
- CoNLL pipeline, BART 2.0:
  - Similar to ParserPipeline
  - Produce Maximal Nps
  - Rely on precompiled Parse Trees

Preprocessing Pipelines

- Create your own pipeline:
  - Keep precompiled levels
  - Straightforward, but needs java programming
  - Depends on available precompiled levels

Preprocessing: run

In `config.properties`, change `runPipeline` to `true` (don't forget to change it back to false once you're done)

```
java -Xmx4000m -Delkfed.corpus=YOURCORPUS
elkfed.main.PreProcess
```

This will preprocess the whole corpus (both train and test)

Language Plugin

Language-specific information is outsourced to the language plugin

Supported languages:

- English
- German
- Italian

Unsupported languages: via Abstract Language Plugin

Coreference for Samoan

- Setting up a baseline system:
  - Create a corpus in the MMAX format, try to specify as many parameters of markables as possible
  - Just run
- Setting up a more language-aware system (requires java programming):
  - Create the Samoan Language plugin
  - Overwrite language-specific functions (e.g., head finder, number and gender calculation modules etc)

Running BART on tabular format

- Import your data in the tabular format
- Run pipeline to finalize your mention extraction (e.g. add missing properties and levels)
- Launch BART as usual
- Export results

Tabular formats for coreference

- Preliminary export format
  - `java -Xmx4000m elkfed.mmax.tabular.TabularExport DIR DOCID`
- SemEval (brackets and numbers), BART2.0
- BIO and BIOM, BART 2.0 – encodes entity and mention type, MIN

Why do we need it?

To run the CoNLL scorer!

Developing new models and features

- Large collection of feature extractors already implemented
- New features can be implemented straightforwardly, but require java programming
  - check (better in BART 2.0) `src/elkfed/coref/features/pairs`
  - Use Dummy Decoder for feature analysis (BART 2.0)

Developing new models and features

- Implemented models:
  - mention-pair, enity-mention (Semantic Trees), ranking
  - BART 3.0: ranking, ILP, graph-based (Cai & Strube)
- Implementing a new model:
  - Well.. that's tricky
  - Check `src/elkfed/coref/algorithms` for a start

Thank you!
Good luck with BART
And let us know how it works for Samoan