CE807 Lab 7
Coreference

March 2 (revised March 7th)

In this lab we are going to experiment with coreference, using both Python libraries and tools outside Python.

All the materials for this lab, including this script, can be found at

http://csee.essex.ac.uk/staff/poesio/Teach/807/Labs/Lab7

Copy all the material in that folder in a directory in your file space.

1. Off-the-shelf coreference resolvers: The Stanford coreference system

There are several publically available coreference resolution systems, some of them available as part of complete pipelines.

One of the most widely used such systems is the Stanford coreference system mentioned in Lecture 7 and part of the Stanford CORENLP information extraction pipeline that we already discussed in Lab 5, available from (and described at)

http://stanfordnlp.github.io/CoreNLP/

Stanford CoreNLP includes a POS tagger, parser, NER system. coreference resolver, and sentiment analyser, and supports Arabic, Chinese, English, French, German and Spanish.

ToDo: read the description of the Stanford CoreNLP pipeline at the page above.

The Stanford Deterministic coreference resolution system is described at

http://nlp.stanford.edu/software/dcoref.shtml

This system implements the multi-pass sieve coreference resolution algorithm described in the lectures.

ToDo: read the description of the Stanford coreference system at the page above

The demo of the Stanford CORE NLP pipeline that we saw in Lab 5, at:

http://corenlp.run/

Can be used to try out coreference resolution as well. Try it for instance on the discourse:

Prime Minister Theresa May travelled to Washington in 2017. She met with President Trump. She invited him to London. They got along very well.

ToDo: experiment with the demo, trying other examples. See if you can break it.

The coreference system can be downloaded with the rest of the Stanford CORE NLP pipeline from a number of sites, including

http://stanfordnlp.github.io/CoreNLP/

ToDo: Download a version of Stanford CORE and uncompress it in a folder.

The latest version of the Stanford CORE NLP pipeline is version 3.7. You can find this at the site above or from the Lab pages. However, the Python wrapper to Stanford CORE discussed below only works for version 3.4.1, so it's probably easiest to download version 3.4.1 directly (this is also the last version that supports Java 6 and 7). You can find it on the lab pages or at

http://nlp.stanford.edu/software/stanford-corenlp-full-2014-08-27.zip

While you wait for the file to download (it's pretty big, 386Mb compressed, 500+Mb uncompressed) read the instructions on how to use the system at:

http://stanfordnlp.github.io/CoreNLP/cmdline.html

As discussed in the page above, the basic way to run the pipeline is from the command line, specifying `edu.stanford.nlp.pipeline.StanfordCoreNLP` as the Java library and using the option –annotators to specify which components of the pipeline to use and the option –file to specify the input file to process.

If you're running the command line from the directory in which you have unzipped the package, then the following command runs the pipeline including all the modules up to and including coreferences on the file input.txt

```
java -cp "*" -Xmx4g edu.stanford.nlp.pipeline.StanfordCoreNLP -
annotators tokenize,ssplit,pos,lemma,ner,parse,dcoref -file input.txt
```

The pipeline writes its output on a file called <inputfile>.xml—in this case. input.txt.xml.

ToDo: examine this output file and see if you can understand how the output of the various modules is represented, including in particular the output of the deterministic coreference resolver.

ToDo: create a new input file containing the example discourse above and run the Stanford Pipeline on it.

In the next Section, we will see one way for using the Stanford Deterministic Coreference System from Python using a wrapper.

2. The Stanford Coreference System in Python

The NLTK Python library does not include a coreference resolver, but there is a Python wrapper for the Stanford CORE system, called `stanford-corenlp-python`. The package is available from github at:

https://github.com/dasmith/stanford-corenlp-python

The package requires the packages `pexpect` *and* `unidecode`. Also, the package requires an older version of the Stanford CORE, 3.4.1, so you have two options.

ToDo:

(i) (easiest version) download the package from github following the instructions there, also installing `pexpect` *and* `unidecode`

pip install pexpect unidecode --user
git clone git://github.com/dasmith/stanford-corenlp-python.git
cd stanford-corenlp-python

at this point if you already downloaded version 3.4.1 of Stanford CORE, just move the directory inside the stanford-corenlp python directory

mv <name of directory where 3.4.1 is> .

or download it now, either from the lab page or from Stanford

wget http://nlp.stanford.edu/software/stanford-corenlp-full-2014-08-27.zip
unzip stanford-corenlp-full-2014-08-27.zip

(ii) if you have downloaded version 3.7,

download the package from github as above,, also installing `pexpect` *and* `unidecode`

then make the 3.7 folder `stanford-corenlp-full-2016-10-31`

a subfolder of the folder containing `stanford-corenlp-python`

but now replace the files `corenlp.py` *and* `client.py` *with the versions from the Lab page (you will probably still need to fiddle with the code though).*

The wrapper sets up a server process running the Stanford CORE pipeline; a client gets output by sending requests to the server. The server is started by the command

```
python corenlp.py
```

You obtain output by sending requests to the server as shown in the file `client.py`

The output of `client.py` is a JSON dictionary containing two keys:

- `sentences`: this lists for each sentences the tokens in that sentence with their POS and NE tags, and their constituency and dependency parse
- `coref`: this lists the coreference links identified by the Stanford Deterministic Coreference System.

(study the example at the Stanford-corenlp-python page)

ToDo: modify client.py to send to the server the request to process the text:

Prime Minister Theresa May travelled to Washington in 2017. She met with President Trump. She invited him to London. They got along very well.

and try to understand the output of the script.

3. A state-of-the-art coreference resolver in Python

CORT (Martschat&Strube, 2015) is a state-of-the-art coreference resolver implemented in Python. It uses a general-purpose framework for representing coreference problems based on *latent variables*. It uses the Stanford CORE pipeline to preprocess text.

CORE can be downloaded from PyPi, and a lot of information / packages / models are available from its GitHub page:

https://github.com/smartschat/cort

3.1 Downloading and basic use

CORT can be installed in Python by running from the command line

```
pip3 install cort
```

ToDo: do that.

In order to run CORT, you need the Stanford CORE pipeline (the version you already installed is fine). Then you need a *model* learned from a corpus. The easiest way to use CORT is to run it using pre-trained models. You can find the models, and instructions on how to run CORT, at the page:

https://github.com/smartschat/cort/blob/master/COREFERENCE.md

ToDo: Read the sections 'Running Cort', 'Model Downloads and Results' and 'Features' at the page above, and download at least two of the models—e.g., model-pair-train+dev.obj (mention pairs) and model-latent-train+dev.obj (latent variables)

After doing this, you can run CORT on raw text from the command line using the command `cort-predict-raw`. The command requires you to specify (i) the input file (example.txt in the example below), (ii) the model, (iii) the directory where StanfordCORE is found, (iv) the feature file (standard_features.txt below) and the suffix to be added to the name of the input file to create the name of the output file.

```
cort-predict-raw -in example.txt \
    -model model-latent-train+dev.obj \
    -extractor cort.coreference.approaches.mention_ranking.extract_substructures \
    -perceptron cort.coreference.approaches.mention_ranking.RankingPerceptron \
    -clusterer cort.coreference.clusterer.all_ante \
    -corenlp /Software/StanfordCoreNLP/stanford-corenlp-full-2016-10-31 \
    -features standard_features.txt \
    -suffix out
```

ToDo:
(i) create a file called, e.g., example.txt, and put in it some text with some examples of coreference (for example the running example about May and Trump)
(ii) create a feature file. You could start by copying the standard features from https://github.com/smartschat/cort/blob/master/COREFERENCE.md
(iii) run the command above.

3.2 Training new models

CORT also includes command-line functions to train new models and test them, but in order to understand this we'll need to discuss CORT first. Meanwhile, you can read the documentation at the pages above.

References

- The Stanford Core NLP documentation at http://stanfordnlp.github.io/CoreNLP/
- The Stanford Deterministic Coreference Resolver page at http://nlp.stanford.edu/software/dcoref.shtml
- The stanford-corenlp-python documentation at https://github.com/dasmith/stanford-corenlp-python
- The CORT documentation at https://github.com/smartschat/cort
- Sebastian Martschat and Michael Strube (2015). **Latent Structures for Coreference Resolution**. *Transactions of the Association for Computational Linguistics*, 3, pages 405-418.