NLP Coursework 3: Coreference resolution

Monday, March 19th.

As seen in the lectures, coreference is a fairly advanced type of information extraction which requires input from a number of other NLP interpretive tasks, from tokenization to NER. As such, playing around with coreference gives you a good idea of what NLP methods can reliably extract from text and how that information can be used.

In this assignment you will experiment with a number of coreference resolution systems, starting by simply using them and then progressively delving in more deeply in what they do. You will do by working your way through this document.

Throughout this document you will be handed out **Tasks**, each of which is worth a number of points. Some of these tasks involve simply reading some documentation; others are more practical. You will have to submit a report containing your solution to these tasks. (The document explains what you need to submit for each task.)

Deadline: March 29th, 23:59:59

PART A – 60 points

1. Coreference with the Stanford CORE NLP System

1.1 The Stanford CORE NLP package

There are a variety of off-the shelf coreference packages. Of these, possibly the best known is the Stanford Coreference System. This system is part of the Stanford CORE NLP package system available from

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

which is one of the most widely used NLP pipelines, so it's worth spending some time learning about it.

*Task 1 (3 pts): Read the description of the Stanford CORE system at the page above. In particular, read the **Annotators** tab describing the NLP modules in the system. In your report, answer the following questions:*

a. *What types of interpretation (annotators) can the system carry out?*
b. *What languages can it handle?*
c. *How can it be used?*

There is an online demo of the Stanford CORE system at:

http://corenlp.run/

*Task 2 (4): Run the demo on an example text **of your own**, and make sure you understand the output of the system. Include in the report a screenshot of the output, and discuss it: e.g., explain what type of interpretation is produced at each stage.*

Task 3 (2): As you should know by now, the system can also process other languages. Read the description of 'Human Languages Supported' and try a text in a language other than English. (Get example sentences from the Web if you can only speak English.)

1.2 Installing the Stanford CORE NLP and testing it

The Stanford CORE NLP system can be downloaded from the github above, or from the Stanford site.

Task 4 (8): download and install version 3.9.1 of the package (if not already installed!!) from

http://nlp.stanford.edu/software/stanford-corenlp-full-2018-02-27.zip

The Stanford CORE NLP is a Java package, so in order to run it, you need a version of Java; for version 3.9.1, you need at least version 8 (see the description page). If you don't have Java version 8, you can install it as follows:

---

1. Download the appropriate version of Java Runtime Environment 8 from http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
2. Extract the files using "tar xf jre_donwload".
3. Type in the following commands to set the JAVA_HOME and PATH variables:
    1. export JAVA_HOME=path to extracted jre file
    2. export PATH="$JAVA_HOME/bin:$PATH"

To confirm you've installed the right version, you can type in "java -version" and check that it is indeed version 8.

The package is pretty large (~500Mb). As you're waiting for it to load, you can go and get a coffee ☺ Once it's downloaded, unzip it, and then change directory into the package directory.

The simplest way to run Stanford CORE NLP is by using the script `corenlp.sh` that you will find in the package directory. This script starts an interactive shell in which you can type some input in natural language to be processed by the system.

Task 5 (3): Start the script. (NB: it takes a while to start as it has to load a number of packages). The interactive shell is ready when you see the prompt NLP>. Try typing in at least two sentences and analyse the results of tokenization, dependency parsing, and NER tagging. Copy the system's output in your report.

The script uses a default configuration of annotators. In order to select different types of annotators you need to dig a bit deeper. Open the script `corenlp.sh` in an editor of your choice. You'll see that the key part of the script is a call to the `java` command. You can run the script from the command line yourself, as in the following example:

```
>java -cp "*" -Xmx8g edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators
tokenize,ssplit,pos,lemma,ner,parse,dcoref -file input.txt
```

The key ingredients of this command are:

- `edu.stanford.nlp.pipeline.StanfordCoreNLP` is the name of the package
- the `-annotators` command specifies the levels of interpretation you require: in this case, tokenization, sentence splitting, POS tagging, lemmatization, NER tagging, parsing, and deterministic coreference
- the `-file` option specifies the input.

The other options specify as follows:

- -cp specifies the classpath – the folders where java is supposed to find libraries. Specifying "*" (**in quotes**) loads all the jar files in the current directory.

- -Xmx8g option specifies the size of the memory. *This is quite important* – the package requires a lot of memory!!

The system outputs a file in .xml format containing the required information.

Task 6 (3): read the documentation for command line use at:
https://stanfordnlp.github.io/CoreNLP/cmdline.html
and answer the following questions:

- *How can you process a list of files?*
- *What is a .properties file?*

Task 7 (5): Create a .txt file containing the text:
Queen Mary University is located in London. It is a great university.
And run the package from the command line, and using the annotators in the example above, on that .txt file. When the system has finished running, open the .xml file and try to understand the output. Include the output in your report and explain it.

It is possible to tell the system to produce output in a format other than XML.

Task 8 (3): Read the instructions in the command line page regarding producing output in other formats, and modify the call above so that the system output is produced in CONLL format.

The best way to specify some of the options above, and more, is to create a *properties file*. The Stanford Core pipeline looks for a file called `StanfordCoreNLP.properties` in the class path, and modifies its behaviour accordingly. For instance, instead of specifying the annotators in the command line, you can specify them in the properties file, as follows:

```
annotators = tokenize, ssplit, pos, lemma, ner, parse, dcoref
```

Task 9 (5): Create a StanfordCoreNLP.properties file specifying the annotators as above, and then call the system without specifying the annotators. Include in the report the output of this call.

1.3 Experimenting with the coreference components of Stanford Core

The Stanford CoreNLP package contains three coreference resolvers: deterministic, statistical, and neural. In this first part of the assignment, we will start with the best known and most widely used of these systems, the deterministic coreference resolver.

Task 10 (10): Read the description of the deterministic coreference resolver at http://nlp.stanford.edu/software/dcoref.shtml and, for more details, the Lee et al 2013 paper describing the system at
http://www.mitpressjournals.org/doi/pdf/10.1162/COLI_a_00152
Answer the following question

- *What does it mean to say that the deterministic coreference resolver is based `on a multi-pass sieve?*

Task 11 (6): Test the limits of the system: i.e., try to find interesting examples of coreference that the system can process correctly, and others that it can't (for full credit: at least one of each)

The deterministic coreference resolver is not based on Machine Learning methods so it can't be 'trained' for a different domain. However, it is possible to modify its behaviour by creating or modifying the StanfordCoreNLP.properties file. The properties that can be set to modify the behaviour of the deterministic coreference resolver are discussed in http://nlp.stanford.edu/software/dcoref.shtml

Task 12 (8): Modify the .properties file to require the deterministic coreference resolver to use the list of animacy and gender information, and compare the results obtained before and after the change.