L4X: Information extraction

Robin Kurtz, Marco Kuhlmann

Goal	Use freely available NLP tools to implement a simple system for information extraction.
Preparations	Read the chapter on Information Extraction in the book by Jurafsky and Martin (2017), linked from the course website.
Report	Submit the requested code.

01 In this assignment you will learn how to use freely available NLP tools to extract **semantic triples** from running text. A semantic triple is a triple composed of a subject, a predicate, and an object. In this assignment, the predicate is represented by a verb, and the subject and the object are represented by **named entities** (see Jurafsky and Martin, 2018). Here is an example of a semantic triple:

bilen – tillhörde – racingföraren Juan Manuel Fangio

Note that named entities in general are made up of more than just one word, making the extraction of semantic triples somewhat harder than the extraction of word–word dependencies that you considered in Lab L4.

02 Your task is to build a system that reads in (tokenised) Swedish newswire text and

- tags the text with parts-of-speech and named entities;
- parses the POS-tagged text to dependencies;
- combines the outputs of the preceding steps and extracts all semantic triples.

We provide you with the newswire text as well as a pre-trained tagger. To solve the task you will have to run the tagger and the parser on the data, and implement the extractor.

03 The newswire text from which you will extract the semantic triples can be found in

`/courses/729G17/labs/l4x/data/webbnyheter-2000.plain`

The text consists of 2 000 sentences, with one token per line and a blank line after each sentence.

04 To tag the newswire text with part-of-speech tags and named entities you will use Stagger[1], a freely available tagger for Swedish developed at Stockholm University. Stagger comes with a pre-trained model trained on the Stockholm–Umeå Corpus (SUC). The tagger and the model can be found in

`/courses/729G17/labs/l4x/stagger/`
`/courses/729G17/labs/l4x/swedish.bin`

Before using Stagger, make sure to read its `README` file.

05 When using Stagger you will most likely encounter an out-of-memory error. To increase the maximum memory allocated to Java, add the argument `-Xmx2G` (or similar) to your `java` call.

06 The output from the tagger is in the CoNLL format that should be familiar to you from lab L4, except that there is no dependency information. Instead there will be two columns that hold information about named entities in the form of **IOB-tags**; see the advanced material for Lecture 3 for more details on this.

07 To parse the tagged text you can use MaltParser (lab L4) together with a suitable pre-trained model. (Think about which of the models that you trained for lab L4 is most suitable for this task.)

08 The output that you will receive from Stagger has two additional columns that you will have to remove before you can use MaltParser. Use the following shell command:

```
cut -f1-10 /path/to/stagger-output > /path/to/parser-input
```

You may want to have a look at the manual of *cut* (type 'man cut' in the terminal).

09 For the triple extraction, you will have to implement a Python script `student.py` which can be run as follows:

```
python3 student.py /path/to/stagger-output /path/to/parser-output
```

The output produced by your script should print one semantic triple per line, with subject, predicate and object separated from each other with a tab character:

```
bilen    tillhörde    racingföraren Juan Manuel Fangio
```

[1] `http://www.ling.su.se/english/nlp/tools/stagger`

10 Start by extracting dependency triples consisting of a verb, a subject, and an object from the parser output. Both subject and object should be either a noun or a proper noun. Consult the documentation of the Swedish Treebank for details about which part-of-speech tags and dependency relations to use for this task.

`http://stp.lingfil.uu.se/~nivre/swedish_treebank/`

The result of this step should be triples such as

bilen – tillhörde – racingföraren

11 To go from dependency triples to semantic triples, you should replace the subject and the object word with the largest named entity that they occur in. Thus in the above example, the single (object) word *racingföraren* is replaced with the enclosing entity

racingföraren Juan Manuel Fangio

You should only return triples where at least one word (either subject, or object, or both) are actually part of a named entity. Note that a named entity may may consist of a single word.