CE706 - Information Retrieval
Assignment 1: Indexing for Web Search

Udo Kruschwitz

5th February 2018

Plagiarism

You are reminded that this work is for credit towards the composite mark in CE706, and that the work you submit must therefore be your own. Any material you make use of, whether it be from textbooks, the Web or any other source must be acknowledged as a comment in the program, and the extent of the reference clearly indicated.

The Context of your Task

Here are some of the requirements included in a current job ad for an *NLP Data Scientist* at Thomson Reuters[1]:
Ability to solve Classical NLP Problems including:

- *...*
- *Named Entity Recognition*
- *Part of Speech Tagging*
- *Co-Reference Resolution / Named Entity Relevance Scoring*
- *...*

Among the skills and experience required, the job description includes:

- *SciKit Learn, NLTK, spaCy, Stanford CoreNLP, gensim, TextBlob*
- *Python, R, Java, Matlab, Scala, Groovy*
- *NoSQL, ElasticSearch, SPARQL, RDF, Semantic Web (e.g OWL)*
- *Open Source Software Development e.g GitLab, GitHub*

That looks exactly like the profile I expect you to have when you finish your MSc!

The Task

Your task is to apply your IR skills to build a processing pipeline that turns a Web site into structured knowledge (thus enhancing your chances of getting the job outlined above). Your system should take HTML pages as input, process them using the kind of techniques that we have been looking at in the module, and output an index of terms identified in the documents.

This assignment comes in stages. Marks are given for each stage. You may choose not to attempt some stages. You might also implement a system that does not strictly follow the stages but will work in the same way. The stages are as follows:

[1] `http://jobs.thomsonreuters.com/ShowJob/Id/89693/NLP-Data-Scientist/`

- **Input/Output (10%)** The system must be able to read Web pages (a small number will do here and they can be stored locally) and produce appropriately formatted output. The Web pages should be processed one at a time using the steps outlined below.
- **HTML Parsing (10%)** Before the text can be analyzed it is necessary to get rid of the HTML tags. The result will be plain text. Note that if you simply delete all HTML tags, you will lose information such as meta tag keywords. Therefore, I strongly suggest that you use some tool to perform this task.
- **Pre-processing: Sentence Splitting, Tokenization and Normalization (10%)** The next step should be to transform the input text into a normal form of your choice.
- **Part-of-Speech Tagging (10%)** The input should be tagged with a suitable part-of-speech tagger, so that the result can then be processed in the next steps.
- **Phrase Detection (10%)** Once you have assigned POS tags to words you can identify noun phrases within the document (e.g. *"computer science"*). You should apply a suitable selection method, using POS tags, that allows you to find sensible phrases (e.g. a noun followed by a noun as in the above example).
- **Ranking (10%)** Not all selected nouns and phrases are equally important. Use a weighting formula (e.g. *tf.idf*) to assign a weight to each noun or phrase selected from an indexed document.
- **Entity Extraction (10%)** Apply a named-entity-recogniser (NER tagger) to your text to identify entities such as person names, locations, organisations, dates etc. Such entities are typically capitalised noun phrases.
- **Engineering a Complete System (10%)** The final system should have control over all the individual components so that there is a single call and all the above steps will be performed.

You will have noticed that the percentages above only add up to 80%. This is because one of the important aspects of the project is that your work should be well documented and your code well commented. **20% of your mark will come from this.** You should submit:

- A description of your implementation: what the code does, and the software you used
- Unedited and commented output from a run of the code submitted using these two Web pages:
  `http://csee.essex.ac.uk/staff/udo/index.html`
  `http://irsg.bcs.org/ksjaward.php`
  (feel free to submit other runs *as well*, i.e. using Web pages of your own choice)
- Output produced by each stage of the processing pipeline for each of the two files, i.e. in the suggested staged architecture outlined above this would be output produced by the HTML parser, followed by the output of the sentence splitter, the tokenizer etc. This could for example be a simple text file per processing step or just the complete concatenated output.
- A *short* discussion of your solution focussing on functionality implemented and possible improvements and extensions.

You may work in pairs. If you do, you each need to submit the same report (please include information about which two reports should be treated as a pair). Both members of a pair will get the same mark unless there is reason to do otherwise.

You can implement your system either on the Linux or the Windows machines. Java, Python, C/C++, Perl and shell scripts are good choices for this project, but you are by no means restricted to those languages. Identify suitable open-source tools that help you building your pipeline.

Submission

The assignment, which counts for **20%** of the overall mark, should be submitted as a single *zip file* via the electronic submission system by **Friday, 23 February 2018, 11:59 (mid-day)**. *The guidelines about late assignments are explained in the students' handbook.*