MERC/IST 2013

Extracção e Análise de Dados na Web

Implementation Project

The goal is to develop a system to harvest, extract and analyze information from online news feeds. The focus will be on Portuguese political personalities and the opinions expressed about them.

1 Functional Requirements

Your system should be able to perform the following functions:

(a) **News Collection and Storage**

The system should poll public news feeds, identify and collect new news items, and store them in a common repository for further processing. The news collection should be performed on demand and new news items found should be added to the repository.

(b) **News Search**

The system should support text searches over the news repository. A user who inputs a set of keywords retrieves a list of news ranked by relevance, using any Information Retrieval model (e.g. BM25).

(c) **Extraction of Named Entities**

The system should be able to process the stored news and discover mentions to Portuguese political personalities. In a search (see item 2), the results should include, for each news item returned, the list of mentioned personalities.

(d) **Sentiment Analysis**

The application should be able to determine, for each entity found in each news item:

(a) if the news item contains opinions on that entity, and

(b) if it does, whether that sentiment is positive, negative or neutral.

When a search is performed, the result should include the sentiment on each personality in each returned news item.

(e) **Other Analysis**

Using the data that you have available (i.e., the news items and the results of items 3 and 4) implement any other analysis that you may find interesting. Use your imagination.

Example: discover who are currently the central people in Portuguese politics or discover how the opinions about someone have evolved over time.

2 Tools and Datasets

2.1 Examples of News Feeds

Several Portuguese newspapers have news feeds on the subject of politics. For example:

Diário de Notícias: `http://feeds.dn.pt/DN-Politica`

Jornal de Notícias: `http://feeds.jn.pt/JN-Politica`

You can use any other news feeds you may find.

> **Note:** Do not worry about duplicate news from different newspapers.

2.2 Possibly Useful Tools

Some of these tools may be useful for the required tasks.

Whoosh (`https://bitbucket.org/mchaput/whoosh/wiki/Home`) Text indexer and searcher.

feedparser (`http://packages.python.org/feedparser/`) Module for downloading and parsing syndicated feeds.

Beautiful Soup (`http://www.crummy.com/software/BeautifulSoup/bs4/doc/`) Library for pulling data out of HTML and XML files.

NLTK (`http://nltk.org/`) Toolkit for natural language processing. Contains modules to, among other things, parse texts into sentences and words, perform part-of-speech tagging, classify text, and do basic information extraction.

sqlite3 (available through the standard library) Python bindings for the SQLite database. SQLite is a C library that provides a lightweight disk-based relational database.

> **Note:** These are just examples. You may not need to use all of them (or any of them) and/or you may want to look for additional tools.

2.3 Possibly Useful Datasets

The following datasets may be useful for the required tasks.

SentiLex-PT 02 (`http://dmir.inesc-id.pt/project/SentiLex-PT_02_in_English`) Sentiment lexicon for Portuguese.

Personalities (available in Fenix) List of personalities mentioned in the news, using the SAPO news service (with the number of mentions).

> **Note:** As for the tools, you may or may not need to use them. Other useful datasets may be made available during the semester.

2.4 Some Recommendations

- The user interface is not important. Your application can, for example, work using only command line arguments. So, *focus on the required functionalities*. If you want to do a fancy interface, do it after everything else.
- For the more complex tasks, such as items 3 and 4, a good way of proceeding is to start with a simple solution. After that is working (even if the results are poor), then try more complex alternatives.
- Accurately discovering named entities and opinions is a hard task. Do not worry too much about the quality of the results. But keep this in mind: even if your solution does not perform well in most cases, *it should perform well in some cases*.
- Try to build a large dataset of news. The more news items you have, the easier it is to explore different cases and try alternative solutions.
- Feel free to reuse code. You can find many libraries for this type of tasks online. However, if you do use them, *make sure you know what they actually do*. You will have to explain it later.

3 Evaluation

The project will be evaluated according to the following criteria:

(a) **Difficulty:** complexity of the proposed solution and effort required to implement it.

(b) **Originality:** originality of the proposed solution. Solutions that are different from those of the remaining students will be valued.

(c) **Results:** quality of the obtained results.

(d) **Presentation:** quality of the written report, which should be clear, complete, and concise, and quality of the code, which should be readable and appropriately commented.

(e) **Discussion:** ability of the students to demonstrate and explain their work.

Each of the above items will be graded from 0 through 4. These grades will be relative to the average results achieved by the whole class. The final project grade is the sum of the grades for all items.

4 Important Dates and Deliverables

8/4–12/4: Project checkpoint. Students should demonstrate the current state of the project. The purpose of this checkpoint is advise the students on how to proceed. It will not be evaluated.

10/5: Project deadline.

13/5–24/5: Project demonstration and discussion.

Students should deliver, through Fenix, the following items:

(a) An archive file (*.zip* or *.tar.gz*) containing all the developed code;

(b) A written report (*PDF*) describing the work done.

The written report should contain:

- The identification of the students and their group number;
- One section for each of the functionalities described in Section 1.

  Each section should contain a *brief* description of the solution adopted for the corresponding functionality. *Do not insert code*—just textually describe the tools and algorithms used. Nevertheless, remember to *indicate which module/class/function of your code implements the solution.* The description of the solution should be clear but succinct—*do not clutter your text with obvious implementation details.*
- One section showing some experimental results.

  Show a printout of your results, for cases you find interesting. Do not clutter the results with useless or repetitive information. If possible, show an analysis using the usual evaluation measures (precision, recall, accuracy, etc.)
- One section containing a critical analysis of your solution (success cases, failure cases, and their explanation) and proposing possible future improvements.

The report should be written on A4 pages, using 12pt font. There is no page limit, but the report should be concise. It can be written in *English* or *Portuguese*. A printed version of the report should be handed to prof. Pável Calado, after being submitted through Fenix.