MERC/IST 2015

Extracção e Análise de Dados na Web

Implementation Project

The goal of this work is to develop a system to harvest, extract and analyze information from online news feeds. The focus will be on Portuguese entities and on the relationships between 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 can input a set of keywords and the system should retrieve 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 named entities. In a search, the results should include, for each news item returned, the list of mentioned entities.

(d) **Discovery of Relationships**

The system should be able to discover relationships between pairs of entities. The definition of *relationship* is left open. For example:

- You can consider that there is a relationship between two entities if they are mentioned in the same news item; or
- You can consider that there is a relationship between two entities if they are mentioned in the same sentence; or
- You can find actual semantic relationships between entities, such as "company X *is-located* in location Y".

Your effort in addressing this issue will be valued.

The user should be able to search for a specific named entity and, if found, the system should show all other entities that have a relationship to it.

(e) **Other Analysis**

Using the data that you have available implement any other analysis that you may find interesting. For example, you may try to discover most mentioned entities in the news, or which entity is most related to other entities, or how mentions to an entity change over time. Originality will be valued.

2 Tools and Datasets

2.1 Dataset

You can find several online news feeds, such as `http://www.dn.pt/rss/` or `http://www.jn.pt/rss/`.

Possibly useful are also the datasets *DBpediaEntities-PT* (`http://dmir.inesc-id.pt/project/DBpediaEntities-PT_01_in_English`), *DBpediaRelations-PT* and (`http://dmir.inesc-id.pt/project/DBpediaRelations-PT_01_in_English`). These contain, respectively, a list of entities and relationships extracted from DBPedia. In DBpedia we can also find a much more extensive list of relationships here: `http://data.dws.informatik.uni-mannheim.de/dbpedia/2014/pt/mappingbased_properties_en_uris_pt.nt.bz2`.

If needed, you can use any other dataset you may find.

2.2 Tools

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

Whoosh (`https://bitbucket.org/mchaput/whoosh/wiki/Home`) Text indexer and searcher. Useful if you want to process text, such as movie synopsis.

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.

scikit-learn (`http://scikit-learn.org/stable/`) a library containing tools for data mining and data analysis

Note that 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 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.
- 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 entities and relationships 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. *You should try to get the best results possible.*
- 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) **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.

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

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

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

7/4–8/4: 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.

15/5: Project deadline.

25/5–29/5: 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 containing a *brief* description of the solution adopted. *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.
- 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 one of the teachers, after being submitted through Fenix.