IST 2017/2018

Processamento e Recuperação de Informação

Lab 01: Python for Text Processing

1

Let us start with some some warm up exercises.

1.1

Implement the *quicksort* algorithm[1] in Python. Define a function that receives a list of objects and sorts the list in place. If needed, use the following pseudocode as a guide.

```
Quicksort(A as array, low as int, high as int)
    if (low < high)
        pivot_location = Partition(A,low,high)
        Quicksort(A,low, pivot_location - 1)
        Quicksort(A, pivot_location + 1, high)

Partition(A as array, low as int, high as int)
    pivot = A[low]
    leftwall = low
    for i = low + 1 to high
        if (A[i] < pivot) then
            leftwall = leftwall + 1
            swap(A[i], A[leftwall])
    swap(A[low],A[leftwall])
    return (leftwall)
```

1.2

Implement a script that reads a list of numeric values from a file (containing one value per line) and prints the same values in ascending order. Use the quicksort function previously defined.

1.3

Implement a script that reads a text file, containing natural language text, and prints each word it contains and the number of times the word occurs.

[1] `https://en.wikipedia.org/wiki/Quicksort`

1.4

Implement a script that reads two text files and counts the number of words in common.

2

Now to try out some useful libraries.

2.1

The Python extension package named `nltk`[2] provides a set of tools that are useful for processing natural language text. For example, you can use the following methods:

- `nltk.sent_tokenize(d)`, which splits a document d into a list of sentences;
- `nltk.word_tokenize(s)`, which splits a sentence s into a list of words;
- `nltk.pos_tag(w)`, which tags the words in list w according to their part-of-speech (i.e., tag words according to morphosyntactic classes such as noun, verb, adjective, ...);

Use the *nltk* package to solve word-counting problems 1.3 and 1.4.

2.2

Again using the `nltk` package, count how many words of each syntactic class (noun, verb, etc.) occur in a document.

2.3

Scikit-learn is a machine learning library for Python[3], which also contains many useful functions to deal with textual information. For example, you can use the following classes:

- `sklearn.feature_extraction.text.CountVectorizer`, which transforms a list of texts into a vector of word counts;
- `sklearn.feature_extraction.text.TfidfVectorizer`, which transforms a list of texts into a vector of TF-IDF values;

Using these classes, solve the word-counting problems 1.3 and 1.4.

Notice that these vectorizers work by first learning the vocabulary (using method `fit`) and then transforming the documents into vectors (using method `transform`).

Also notice that the method `transform` returns a *sparse matrix*, defined in the `numpy`[4] package. These can be accessed using the notation `m[line,column]`.

[2] `http://www.nltk.org`
[3] `http://scikit-learn.org/stable/`
[4] `http://www.numpy.org/`

2.4

Again using *Scikit-learn* transform two documents into TF-IDF vectors. Use those vectors to compute the cosine similarity between the documents.