Data Analysis - lab 7

ISEP – December 17th 2019

Instructions: Prepare a report including the source code and the results. Deposit your report on Moodle and don't forget your binome's name or to make 2 deposits if you did not work alone.

Libraries

This lab may require the following libraries : nltk, math, wikipedia and textblob. The nltk library may be used to download text corpuses and sub-packages, e.g.:

```
import nltk

nltk.download('stopwords')
from nltk.corpus import stopwords

nltk.download('gutenberg')
from nltk.corpus import gutenberg
```

A Text processing with NLTK

The Natural Language Toolkit, or NLTK for short, is a Python library written for working and modeling text. It provides good tools for loading and cleaning text that we can use to get our data ready for working with machine learning and deep learning algorithms.

1. Download the file *metamorphosis_ clean.txt* from Moodle, open it with a text editor and read it briefly.

```
import nltk
from nltk import sent_tokenize
from nltk.corpus import stopwords

filename = 'data.txt'
file = open(filename, 'rt')
text = file.read()
file.close()

sentences = sent_tokenize(text)
words=word_tokenize(sentences[0])
nltk.pos_tag(words)

text_words = text.split()

print(stopwords.words('english'))
en_stops = set(stopwords.words('english'))
```

2. We are now going to use the nltk library to tokenize this text (you may use the partial code above):

   - Load the text in Python using the commands *open* and *file.read*.

- Use the command *sent_tokenize()* to start slicing your text into sentences. Display the 1st and 11th sentence.
- Use the command *word_tokenize()* to slice your first sentence into words. Then use *nltk.pos_tag()* to display the function of each word.
- Use the command *text.split()* to display the first 100 tokenized words of the text.
- Since we are going to use english language, we want the nltk library to be able to detect stopwords in english in order to remove them: Set up nltk to detect english stopwords and display them.
- Remove all stop words from the text and print the new text.

3. Load the coll2000 corpus, a dataset from the wall street journal.

```
nltk.download('conll2000')
from nltk.corpus import conll2000
```

- Using a loop and the *sents()* function of *coll2000*, display the 6 first sentences of this dataset.
- Likewise use, *tagged_sents()* to tag them.

4. **Bonus:** If you have access to pip or pip3, install *gensim_sum_ext* and use the code below to summarize the text from question 2:

```
from gensim.summarization import summarize
print (summarize(text))
from gensim.summarization import keywords
print (keywords(text))
```

B WordNet

WordNet is a lexical database for the English language, which was created by Princeton University, and is part of the NLTK corpus. You can use WordNet alongside the NLTK module to find the meanings of words, synonyms, antonyms, and more. We are going to go through some examples.

1. First, you are going to need to import WordNet:

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
from nltk.corpus import wordnet as wn
```

2. The function *synsets()* from wordnet makes it possible to find synonyms for a word and even to print their definition.

- Using *synsets()*, print all the elements of synsets for the word "Love". Comment.
- Use the properties *definition()*, *lemmas()* and *examples()* of your first synonym. What do these do ?
- Using wordnet documentation, find a way to get all the antonyms of the word "Love".

- Print all synonyms and antonyms for the word "Love".

The WordNet corpus reader gives access to the Open Multilingual WordNet, using ISO-639 language codes. (you may need this: nltk.download('all')).

3. Print all available language in wordnet.

4. Print lemma name of the first synset of the word "love" in Japanese.

   Several similarity measures are defined inside WordNet: path_similarity, lch_similarity, wup_similarity, etc.

5. Retrieve the sets of synonyms for the words "cat", "dog" and "car". Then, compute, the similarity between all pairs of sets using the 3 similarity functions proposed above. Comment.

```
cat = wn.synset('cat.n.01')
dog = wn.synset('dog.n.01')
car = wn.synset('car.n.01')
```

C TextBlob

TextBlob aims to provide access to common text-processing operations through a familiar interface. You can treat TextBlob objects as if they were Python strings that learned how to do Natural Language Processing.

1. Create your first Textblob object using the following code:

```
from textblob import TextBlob
Textblob_object = TextBlob("Python is a beautiful high-level, general-purpose programming language!")
print(Textblob_object)
```

   - Use the *tags* property of your newly created object. What does it do ?
   - Use the *noun_phrases* property to extract the nouns from the sentence and display the results.

   The sentiment property returns a named tuple of the form Sentiment (polarity,subjectivity). The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

2. Consider the sentence "Paris is so beautiful, I like Paris, people are so nice !".

   - Create a textblob object with this sentence.
   - Using the *sentiment* property, analyze its polarity and subjectivity. Comment.

   Let us consider this text from Wikipedia : "During the Iron Age, what is now metropolitan France was inhabited by the Gauls, a Celtic people. Rome annexed the area in 51 BC, holding it until the arrival of Germanic Franks in 476, who formed the Kingdom of Francia. The Treaty of Verdun of 843 partitioned Francia into East Francia, Middle Francia and West Francia.

West Francia, which became the Kingdom of France in 987, emerged as a major European power in the Late Middle Ages, following its victory in the Hundred Years' War (1337–1453). During the Renaissance, French culture flourished and a global colonial empire was established, which by the 20th century would become the second largest in the world. The 16th century was dominated by religious civil wars between Catholics and Protestants (Huguenots). France became Europe's dominant cultural, political, and military power in the 17th century under Louis XIV. In the late 18th century, the French Revolution overthrew the absolute monarchy, establishing one of modern history's earliest republics and drafting the Declaration of the Rights of Man and of the Citizen, which expresses the nation's ideals to this day."

3. Use the *words* or *sentences* properties to break this text into words or sentences respectively.

4. Print the sentiment analysis for each sentence.

5. Use the *translate()* function to make an attempt at translating this text into french.

D TF-IDF analysis with textblob and wikipedia

Using the Wikipedia package and the textblob package, we are going to analyze text corpuses from Wikipedia so that we can compute different statistics on word freqencies inside these corpuses.

1. Use the following steps to write a function that computes the Term Frequency Inverse Document Frequency (you will need the math and textblob libraries):

   - Using the properties *words* and *words.count()* of blobtext, write a function **tf** that takes into parameter a word and a blobtext and returns the word frequency inside the textblob.
   - Write a function **n_containing** that takes into parameters a word and a textblob list and returns how many of them contain this word.
   - Write a function **idf** that takes into parameters a word and a textblob list and returns the inverse document frequency for the word passed as parameter.
   - Write a function **tfidf** that takes into parameter, a word, a textbloc and a textblob list. This function must return the Term Frequency Inverse Document Frequency.

2. Create a textblob list on the following topics : "France", "Elton John", "Python", "Fox", "Isaac Newton", "Zinedine Zidane". To do so, you may use the example below:

```
1 import wikipedia
2 from textblob import TextBlob
3
4 bloblist = []
5 bloblist.append(TextBlob(wikipedia.summary("Isaac Newton")))
```

3. For all texts, remove the stopwords, change everything to lower case, and create a bag of words containing all other words.

4. Merge your list of words and compute the tfidf of all words for all documents in the corpus.