Homework 4:
NLTK: Corpus Linguistics

Robert Litschko[*]
Symbolische Programmiersprache

Due: Thursday November 17, 2022, 12:00 (noon)

In this exercise you will:

- practice object oriented programming
- use NLTK to analyze text and perform basic corpus linguistics
- implement a language guesser

Exercise 1: Object-oriented programming II [3 points]

For this exercise we will use the solution of last weeks exercise as a starting point. Please implement your solution in the `hw04_nltk/document.py` file.

This part of the homework will be graded using unit tests by running:

```
python3 -m unittest -v test_document.py
```

Implement the following methods:

1. Inheritance: Modify the class `PDFDocument` to make it inherit methods and attributes from `TextDocument`.
2. Override the constructor in `PDFDocument`: It should accept a `docid` and a `filepath` variable (string) that points to the location of a pdf file on disk[1]. You should first use the `load_pdf()` function provided by us to extract the content of the pdf file[2] and then pass the text and the document id to the parent constructor.

[*]Credit: Exercises are based on previous iterations from Katerina Kalouli.
[1]For example: `/home/usr/myfile.pdf`
[2]For this to work you need to install PyPDF with '`pip install PyPDF2`', refer to last weeks exercise for more information on installing python packages.

3. Aggregation: Create a class `Author` with the attributes `firstname`, `lastname` and `age`.[3] Extend the constructor of `PDFDocument` to by adding an additional parameter and instance attribute `author`.

Exercise 2: Lexical information [7 points]

Take a look at `hw04_nltk/analyze.py`. In this exercise you will have to implement some methods in class Analyzer, that can analyze any text. In this exercise we will use `analyze.py` to compute basic statistics on `ada_lovelace.txt`.

This part of the homework will be graded using unit tests by running:

```
python3 -m unittest -v test_analyzer.py
```

Implement the following methods:

1. __init__(self, path) - should read the file text, create the list of of words (use `nltk.word_tokenize` to tokenize the text), and calculate frequency distribution of words (use `nltk.FreqDist`)
2. numberOfTokens(self) – should return the number of tokens in the text
3. vocabularySize(self) – returns the size of the vocabulary.
4. lexicalDiversity(self) – returns the lexical diversity of the text.
5. getKeywords(self) - returns words as possible key words, that are longer than seven characters and occur more than seven times (sorted alphabetically)
6. numberOfHapaxes(self) – returns the number of hapaxes in the text
7. avWordLength(self) – returns the average word length of the text.

Exercise 3: Language Guesser [4 points]

Implement a language guesser that determines the language it thinks the text is written in. The decision should be based on the frequency of individual words in each language. Take a look at the file `hw04_nltk/model_lang.py`. In this exercise you will have to complete some methods to make it work (you can also take a look at the last slides of our Tuesday session to get some more help).

This homework will be graded using unit tests by running:

```
python3 -m unittest -v test_lang_guesser.py
```

Complete the following tasks:

[3]You only need to implement the constructor.

1. Complete the class method `build_language_models(self)`. This method should return a conditional frequency distribution where:
   a) the languages are the conditions
   b) the values are the frequency distribution of lower case words in corresponding language
2. Complete the class method `guess_language(self,language_model_cfd, text)`:
   a) it should calculate the overall score of a given text based on the frequency of words accessible by `language_model_cfd[language].freq(word)`.
   b) it should return the most likely language for a given text according to the scores