Homework 5:
NLTK: Preprocessing

Robert Litschko*
Symbolische Programmiersprache

Due: Thursday November 24, 2022, 12:00 (noon)

In this exercise you will:

- gain some hands-on experience on @classmethods decorators
- use NLTK to do some basic preprocessing of texts

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

```
python3 -m unittest -v hw05_preprocessing/test_preprocessor.py
```

Exercise 1: Creating @classmethods [2 points]

Take a look at `hw05_preprocessing/preprocessor.py`. In this exercise you will have to modify the two @classmethods so that *from_nltk_book* can be used to initialize a preprocessor when the text is from the nltk book and *from_text_file* can be used to initialized a preprocessor when the text is read from a file:

1. `from_nltk_book(cls)`: **[1 point]**
   - This function should read the inaugural text from the nltk book into a string.
   - In its return statement, the function should call the constructor of the class with two parameters: the text string and the text_id "inaugural" (this function needs to return an object of type `Preprocessor`).
2. `from_text_file(cls, path, text_id)`: **[1 point]**
   - The function shoudl read the text from the file located at `path` into a string,
   - In its return statement, it should call the constructor of the class with two parameters by passing the `text_id` parameter and the loaded file text (this function needs to return an object of type `Preprocessor`).

*Credit: Exercises are based on previous iterations from Katerina Kalouli.

3. Go to the `__main__` section and create exacly two instances of the `Preprocessor` class. One instance should be created through the text file ada_lovelace.txt (use `text_id`="ada_lovelace" when you call `from_text_file`) and the other one through the inaugural text included in the nltk book.[1] **[optional]**

Exercise 2: Preprocessing Texts [8 points]

Take a look at `hw05_preprocessing/preprocessor.py`. In this exercise you will have to implement some methods in class Preprocessor in order to apply some basic preprocessing on your texts. Implement the following methods:

1. `get_no_of_sents(self)` should split the input text into sentences and return the number of sentences in the text. **[1 point]**

2. `tokenize_text(self)` This function should split the text into tokens and return the tokenized text. Use an nltk method for that, equivalent to s.split(), but not the WhiteSpaceTokenizer. **[1 points]**

3. `lemmatize_token(self, token)` should lemmatize the given token, after converting it to lowercase, and return the lemmatized form. **[2 points]**

4. `get_20_most_frequent_words(self)` should return the 20 most frequent words of the text after removing the stopwords and all non-alphanumeric characters (in other words, after removing all punctuation). Note that the stopwords in NLTK are stored in their lemmatized form, so you will have to make sure you check whether the *lemma* of each word is contained in the stopwords list. Make sure you use the method `tokenize_text()` that you implemented above. Note that even after removing stopwords, the words "was" and "has" will still be in the list of most frequent words – this is because NLTK does not lemmatize these words correctly and thus does not recognize them as stopwords. **[2 points]**

5. `get_originality_score(self, most_freq_words)` should return the originality score of a text. This score can be measured by counting how many words contained in the swadesh list of nltk are also contained in the list with the 20 most frequent words of the text. For example, if 10 words can be found in both lists, the originality score is 10. The list with the 20 most frequent words should be given to the method as a parameter. **[2 points]**

6. Go back to the `Preprocessor` objects you created earlier (Exercise 1.3). For each instance, complete the following tasks below (note that each instance should be initialized with the appropriate @classmethod). **[optional]**
   - Print out the numbers of sentences of the text.
   - Print out the 20 most frequent words of the text.
   - Print out the originality score of the text.

[1] We provided you with a copy of ada_lovelace.txt in your homework folder, please use this file.