Homework 3:
Representing Simple Documents

Robert Litschko*
Symbolische Programmiersprache

Due: Thursday November 10, 2022, 12:00 (noon)

In this exercise you will:

- Implement a simple document class.
- Get experience using the `unittest` framework.

You can monitor your progress by calling (from the `src` direcory:)
`python3 -m unittest hw03_docs/test_documents.py`

Exercise 1: TextDocument class [10 points]

1. Implement the helper method `normalized_tokens` that takes a string and returns a list of tokens[1] (converted to lower case).
2. Complete the constructor for `TextDocument`. You need to add `word_to_count`, a dictionary that maps every word to the number of its occurrences in this document.
3. Complete the class method `from_file`, that creates a document by reading a file, and calls the constructor with the text read from the file (and the filename as its id).
4. Implement the `__str__` method. It should return a string representation that is at most 25 characters long. If the original text is longer than 25 characters, the last 3 characters of the short string should be *"..."*. For example, the document text:
   "Dr. Strangelove is the U.S. President's advisor."
   Should yield the `str` representation:
   "Dr. Strangelove is the..."

*Credit: Exercises are based on previous iterations from Katerina Kalouli.
[1]For the purpose of this exercise you can split the input string by whitespaces. Alternatively, you can also use nltk's `word_tokenize` function (see below).

5. Complete the function `word_overlap` that determines the number of words that occur in both of the documents (`self` and `other_doc`) at the same time. Every word should be considered only once, irrespective of how often it occurs in either document (i.e. we consider word *types*)[2]. In other words this should return the size of the intersection of the word sets for both documents.

Using NLTK (Optional)

You are able to solve this homework without any external Python-packages. However, the `nltk` package is a widely used text processing library that implements a range of common operations for you. We will see more on NLTK starting from lecture 4, but provide information on how you can already install it in this homework.

You are welcome to install and explore it on your own for solving the above tasks.[3] If you work on the cip pool computers, nltk should already be installed. To use the `word_tokenize` function in nltk, you may have to download the resource `'punkt'`:

1. open the Python interactive shell:
   `python3`
2. then execute the following commands:
   ```
   >>> import nltk
   >>> nltk.download('punkt')
   ```

If you use your own computer:

- **Unix (with Python3):**
  `sudo apt-get install python3-pip`
  `sudo pip3 install -U nltk`
  Test the installation:
  `python3`
  `>>>import nltk`

If you use a virtual environment:

- **Unix venv (with Python3):**
  `sudo apt install python3-venv` (on debian/ubuntu)
  `cd path/my_group/src`
  `python3 -m venv venv`
  `source venv/bin/activate`
  `pip3 install -U nltk`
  Test the installation:
  `python3`
  `>>>import nltk`

[2] More on the distinction of tokens and types in lecture 4

[3] We will use nltk in future lectures and exercises. It's therefore highly encouraged that you familiarize yourself with the package.

- **Anaconda:**
  `conda activate myenv`
  `conda install -c anaconda nltk` (or `pip install nltk`)
  Test the installation: `python`
  `>>>import nltk`
- **Windows:** `http://www.nltk.org/install.html`
- **PyCharm:** View > Tools Windows > Python Packages
- **The handling of external Python-packages is a crucial skill!** If you encounter difficulties, ask fellow students or the tutors.