UNIVERSITY OF TÜBINGEN, DEPARTMENT OF GENERAL LINGUISTICS

Homework 2: Lexical Complexity

Xiaobin Chen, Zarah Weiss, and Detmar Meurers

Due November 15th, 2019, 10am

1 MEASURING LEXICAL COMPLEXITY OF GRADED TEXTS

In this assignment, you will implement a text complexity analyzer that calculates the lexical complexity of graded texts. You will implement a few measures from each of the three lexcial complexity categories: lexical density, lexical sophistication, and lexical variation.
For lexical density, implement the N_{lex}/N measure discussed in the lecture. For lexical sophistication, use the SUBTLEXus frequency list and choose two indexes from the list to implement a lexical sophistication calculator. For lexical variation, implement the Type-Token Ratio measure.
Compare the calculated results across reading levels and summarize your findings in a few sentences.

1.1 Input and Output Requirements

Your program should take the following input arguments (in this order):

1. An input folder containing multiple subfolders. The names of the subfolders are the reading levels of the texts within them. Each subfolder contains multiple plain text files to be processed. Each text file contains one reading text.
2. The name of the output csv file to which the results should be saved

Your program should output a single csv file (as specified by the user in the input) with the following columns:

File the file name

Reading.level the reading level of the file, which is the same as the name of the folder the file is in

Lexical.density the lexical density value (N_{lex}/N) of the text

Mean.frequency.<index1> the mean frequency of a frequency index of your choice from SUBTLEXus (see resources)

Sd.frequency.<index1> the standard deviation of a frequency index of your choice from SUBTLEXus (see resources)

Mean.frequency.<index2> the mean frequency of another frequency index of your choice from SUBTLEXus (see resources)

Sd.frequency.<index2> the standard deviation of another frequency index of your choice from SUBTLEXus (see resources)

TTR the Type-Token Ratio of the text

In short, for the sophistication measures, choose two indexes from SUBTLEXus (`https://www.ugent.be/pp/experimentele-psychologie/en/research/documents/subtlexus`) and calculate the mean frequency of all the words (and their SD) in a text.

1.2 Resources

You may write your code either in Java or in Python 3. Use one of the following APIs for sentence segmentation and tokenization:

- **OpenNLP** version 1.9.1 using the corresponding off-the-shelf model for English[1]
- **Stanford CoreNLP** version 3.9.2 using the corresponding off-the-shelf model for English[2]
  - Note: **don't** use the model English (KBP)
- **NLTK** version 3.4.5[3]
  - Use the nltk.tokenize.punkt sentence segmentizer and
  - the nltk.tokenize.word_tokenize tokenizer with the pretrained English model.
- **spaCy** version 2.2[4]
  - use the English model en_core_web_sm

Since these tools may differ in the segmentation they produce, we will evaluate the output of your code against results that we obtain using the same tool using the model and version specified here. If you choose to use another tool, we will match the results against the one that yields the most similar results to yours.

1.3 Submission

Submit your executable code via moodle in the format specified under *General Submission Requirements* at the end of this document. The submission deadline is

- **November 15th, 2019 at 10am**

1.4 Hints

- Familiarize yourself with the SUBTLEXus frequency list and get to know the differences between the frequency indexes.
- Convert the frequency list file into a format more easily readable by your programming language.
- Think about how you deal with words that are not included in the list.

[1] `https://opennlp.apache.org/download.html`
[2] `https://stanfordnlp.github.io/CoreNLP/download.html`
[3] `https://www.nltk.org/install.html`
[4] `https://spacy.io/`

2 USE YOUR CODE ON THE GIVEN DATA SET

Run your code on the following test data and include the resulting csv under the name **results-task2.csv** before you submit.

- Test data in **corpus.zip**
- The SUBTLEXus frequency list **SUBTLEX-US_frequency_list_with_PoS_and_Zipf_information.xlsx** (you may want to convert the file to a format more easily processable by your programming language)

The data is a subset of the Newsela data, a leveled news data set.[5]

3 KNOW YOUR CODE

You need to be able to explain your pipeline in a few words in front of the class. We will choose a student at random to elaborate on their code. Relevant information are for example:

- Which components did you use in your pipeline?
- In which order are they executed?

4 GENERAL SUBMISSION REQUIREMENTS

4.1 Submission

Homework can only be submitted via moodle before the end of the respective deadline. Your submission must always include a README file named **README.txt** including the following information:

- Full names and email addresses of all students who contributed to and want credit for this homework.
- The programming language use used
- A full list of the NLP APIs you used
- The name of your main file that should be called for execution of your code.
- A preliminary summary of your findings when comparing the lexical complexity of the texts with different reading levels. One or two paragraphs should suffice.

You may also submit other files (e.g. plots, animations, etc.) that you think will help us understand your points.

4.2 Programming Languages

- We assume that you program in either Java or Python 3. If you have a good reason to use another programming language, talk to us about it early as possible.
- If you use Java, make sure to use Maven to handle your dependencies (`https://maven.apache.org/`). Submit an executable jar file as well as your source code.
- If you use Python 3, make sure to use a virtual environment to handle your dependencies. If you are not familiar with virtual environments, you may finde the following links useful:
  - `https://www.geeksforgeeks.org/python-virtual-environment/`
  - `https://docs.python-guide.org/dev/virtualenvs/`
- Make sure that your code can be executed. If we cannot run your code, it counts as a failure.

[5] `https://newsela.com/`