KED2020 Exercise 3: Python NLP

Alex Flückiger

2020

Requirements

Your submission needs to comply with the following requirements:

- Deadline: 14 May 2020, 23:59 Hand in your solutions via the respective exercise module on OLAT. Keep in mind to submit on time, as the module is only open until midnight.
- Submission: a single, executable Python script.
- Naming schema: *SURNAME*_KED2020_ex_3.py
  Replace *SURNAME* with your surname.
- Your Python script should look similar to the template provided here. Add all the commands that you have used in this exercise to this script. Additionally, write a short comment for each command that explains what it does.
- Find solutions individually. Ask friends or Google whenever you feel lost. In terms of programming, Google may be your best friend. When you struggle to understand a concept or have technical issues, feel free to post in the forum on OLAT.

1 Corpus of 1 August Speeches by Swiss Federal Councilors

In practice, you need to work with a collection of text documents, which is also called a corpus. In this exercise, you will analyze the yet unpublished dataset of historical 1 August speeches by Swiss Federal Councilors.

You aim to compare the vocabulary across two groups of documents, specifically, speeches given before the turn of the millennium and those given after. For this, follow these steps:

1. Make sure that your local copy of the Github repository `KED2020` is up-to-date with `git pull`. You find the dataset as `.csv` file here: `materials/dataset_speeches_federal_council_2019.csv`
2. Open the editor *Spyder* and set the working directory to your local copy of the repository (or you need to change the file path): `YOUR_PATH/KED2020/`
3. Import the necessary modules in your script.
4. Create a corpus object with the function provided on the next page. You can copy it from `/scripts/KED2020_10.py` rather than from this PDF.
5. Create two subcorpora from the main corpus by filtering according to the following criteria (meta attributes `Sprache`, `Jahr`)[1]
    1. all German speeches hold before 2000
    2. all German speeches hold as from the year 2000
6. Print out the number of documents for both of the subcorpora.
7. Export the lowercased unigram vocabulary for both of subcorpora to a file. Bonus: Set an additional argument in the function to get the relative frequency: `weighting="freq"`
8. Compare both vocabularies. How did the language change between these periods? Which are terms have been the most popular in which period (absolute frequency does not matter)? Summarize your impression in a few lines.

Alternative: Gender instead of History

Are you rather tired of history and more interested in gender? You can alternatively divide the corpus by gender. Nevertheless, you should limit to German speeches hold from 1999[2] and on (meta attributes `Geschlecht`, `Sprache`, `Jahr`).

[1] To combine multiple criteria, simply use the logical `and` operator: `filter_func = lambda doc: doc._.meta.get("year") > 1900 and doc._.meta.get("year") < 2000`

[2] In 1999, Ruth Dreifuss was the first woman who gave a speech in the role of a Swiss Federal President. Until 2004, only speeches by Swiss Federal Presidents are included in the dataset. From this point, speeches by Swiss Federal Councilors are covered as well. Thus, it would be not sensible to consider documents from the past for which there are no female records.

```python
def get_texts_from_csv(f_csv, text_column):
    """
    Read dataset from a csv file and sequentially stream the rows,
    including metadata.
    """

    # read dataframe
    df = pd.read_csv(f_csv)

    # keep only documents that have text
    filtered_df = df[df[text_column].notnull()]

    # iterate over rows in dataframe
    for idx, row in filtered_df.iterrows():

        # read text and join lines (hard line-breaks)
        text = row[text_column].replace('\n', ' ')

        # use all columns as metadata, except the column with the actual text
        metadata = row.to_dict()
        del metadata[text_column]

        yield (text, metadata)

f_csv = '../materials/dataset_speeches_federal_council_2019.csv'
texts = get_texts_from_csv(f_csv, text_column='Text')

corpus_speeches = textacy.Corpus(de, data=texts)
```

2 Test your script

This task is a simple sanity check. You don't need to include it in your submission.

Your deliverable has to be a runnable script comprising all the commands to accomplish the tasks above. To test your script, run it with the commands below. Once you start the script again, it executes all the commands, one after another. Everything should be reconstructed accordingly. If not, correct the script so that it runs through without any issue.

```
# run script
python script_name.py
```

3 Feedback

Please answer the following questions at the end of your script. Start your answers with the `#` symbol to make them comments that are ignored when running the script.

1. Do you have any questions concerning the exercise or the commands?
2. How long did it take to solve this exercise? Give a fair estimation.