Prof. Björn Gambäck
TA. Tollef Jørgensen

Lab 1 – Corpora and Text Processing

Deadline: 23:59, February 2, 2022

Guidelines

You are to submit a zipped folder with your name (first_last_tdt4310.zip) containing:

1. Source files (Python as .py or Notebooks .ipynb) with the following format: Lab[**Lab#**]_[**Exercise#**].[**py/ipynb**] (e.g., Lab1_1a.py)
2. Code should follow certain standards and software engineering principles. Modularity and reusability are to key factors. Unreadable and heavily repeated code may cause a resubmission. Refer to this summary of Clean Code for coding style: `https://gist.github.com/wojteklu/73c6914cc446146b8b533c0988cf8d29`
3. A concise report formatted as a pdf. Feel free to use Word/Pages, or any other means to create pdf files. A template for LaTeX is provided on blackboard if preferred. The report **must contain outputs from your code** along with answers to respective tasks.

A single Jupyter Notebook for the entire lab may suffice if you properly format it with markdown cells. In this case, you can submit it as "*your-name*-Lab1.ipynb".

Exercises

1 – Managing text and filtering

Using the following words:

```
["further", "Forward", "Foreign", "financE", "Forgive", "feature", "federal",
   "failurE", "Feeling", "finding", "freedom", "Foundry"]
```

(a) Print all words beginning with *fo*

(b) Print all words ending with *e*

(c) Reflect on how simple techniques like these can bring value to a project

(d) Converting words to lowercase is a frequent process in text cleaning. Can you think of any issues that arise by doing this?

2 – Reading a corpus

In NLTK, several corpora are available[1]. Load the Brown corpus and extract data (i.e. `words`) from at least two categories of your choice (or all)[2]. The goal is to find basic trends or identifiers for each category (i.e. spotting differences in the data). Do the following:

(a) Look at the top 5 most common words. Do you notice any similarities? Explain your findings

(b) Filter *stopwords* and repeat (a). Describe at least two new techniques you would use to further improve the result

(c) Implement at least one technique of your choice and repeat (a)

(d) Plot a feature you find interesting (e.g. lexical diversity). NLTK has several built-in methods for plotting (make sure you have Matplotlib installed!)

[1] `https://www.nltk.org/nltk_data/`

[2] Hint: you can find the categories with nltk.corpus.brown.categories()

3 - Building your own corpus

A real-world project needs updated data. Not all services provide proper APIs to access their data, and must thus be mined. In this task, you need to build a web scraper for your favorite news site, separating each article. Norwegian sites can also be used, you can find a list of stopwords here: `https://gist.github.com/kmelve/8869818`.
The following data should be extracted:

1. Headline
2. Ingress
3. Several sentences from the text body
4. Published date
5. Topic, if it is easily accessible. Several news sites allow you to browse news by category.
6. URL

Try to use at least 50 articles for somewhat interesting results. However, feel free to create your own huge dataset for future use! The data may be saved in any format of your choice (e.g., .txt, .json), as long as you are able to load the content into a Python program. If you managed to separate on topics, feel free to do the tasks *d-f* below separately for each topic (easy once you have the code for *one*)

(a) Build the corpus. Chapter 2, section 1.9 (NLTK) and Chapter 2 (ATAP) explain of how this can be done using NLTK (e.g., using a custom HTMLCorpusReader), but you are free to load the text as you prefer. As long as the code is readable!

(b) Clean the data using what you have just learned. There are likely several new problems to handle. Select one and explain how you would solve it – no need for code

(c) Separate the text body (sentences) into tokens (words) by splitting on spaces. Find and implement one improvement on this tokenization process

(d) Print the 10 most common words

(e) Create bigrams from the texts that do not contain stopwords. Print the 10 most common bigrams

(f) Find the most typical headline, i.e. the headline that contains the most common words

(g) Find the headline that contains the highest number of most frequently used words (from the common words in task d, but not limited to top 10). If you want to, you are free to explore more advanced techniques if desired, e.g., generating a new headline based on the frequency of words, etc.

Feedback

Some feedback on this type of exercise would be greatly appreciated. What do you think of open questions (e.g., discussions)? When tasks are simply "compute x, print y" for dataset, this encourages little freedom to experiment, which motivated the exercises you see above. This is (obviously) not a required field to answer.