

Lab 1 TDT4310

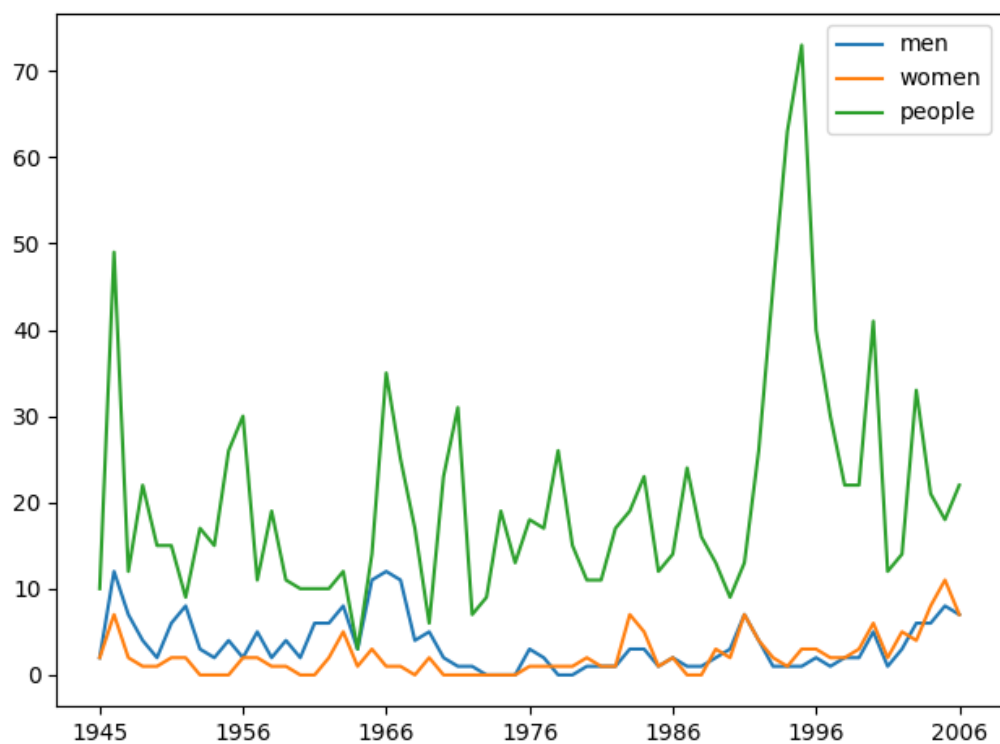
Vebjørn Ohr

February 05 2021

Exercise 1

Exercise 2

To find the frequencies the count function of Python lists where used on each of the texts of the corpus.



Exercise 3

Task a)

The pig latin algorithm was implemented by iterating through all characters of the word until a vowel appeared. The characters iterated thorough was then moved to the back, and 'ay' appended.

Task b)

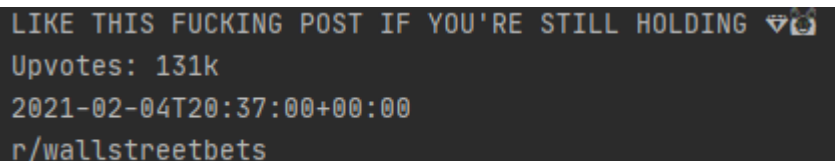
For converting text the word_tokenize function of NLTK was used and then the function from task a) was used on each token.

Task c)

It turned out to be quite tricky to decode Pig Latin as you don't know how many consonants were moved behind. It seems like a dictionary would be needed, and even then some words can be translated to the same word in pig latin, meaning you need to know the context as well.

Exercise 4

Selenium was used for getting HTML data and BeautifulSoup was used for handling it. The posts were found by using the select functionality of BeautifulSoup, using a css selector for finding elements with the class name thing". Promotions were skipped as they don't contain any of the desired information. For each post the title, number of upvotes, time of posting, and subreddit were found using a similar approach. Which selector to use was found by inspecting the html code in the browser. Two of the posts gathered are shown below.



```
LIKE THIS FUCKING POST IF YOU'RE STILL HOLDING 📉📊  
Upvotes: 131k  
2021-02-04T20:37:00+00:00  
r/wallstreetbets
```

```
Gun Rights Rally #VirginiaRally  
Upvotes: 43.2k  
2021-02-04T12:01:47+00:00  
r/PoliticalHumor
```

Exercise 5

The corpus is made by extracting the full text from all the tweets for each search term. The text from each search term is saved to a single .txt file, meaning each search term has its own file with all relevant tweets.

The PlaintextCorpusReader class of NLTK is then used to create a NLTK corpus from the files. NLTK also has a built in Twitter framework for building a twitter corpus, but I had some problems implementing this with the given json files.

Task a)

The corpus was tokenized using the Python list split function. This was chosen instead of the NLTK tokenizer to keep Twitter hashtags. This also meant that some other symbols and empty strings needed to be removed. I made the function able to both filter the entire corpus, or a text file of the corpus as this was needed for later tasks. The NLTK English stopwords was used to remove stopwords by iterating through the list of tokens and only including non-stopwords. Clearly there are other languages present other than English in the tweet texts, but this was not taken into consideration for this implementation.

Task b)

For finding the most frequent words, the NLTK FreqDist function was used on the corpus.

```

TOP 10 WORDS FOR THE ENTIRE CORPUS:
  anime 160
  artificial 157
  music 148
  #ai 142
  intelligence 141
  backend 141
  like 114
  using 103
  new 103
  database 97

```

Task c)

The same method as in task b) was used, but now iterating through each text file of the corpus. The results for two of the search terms are included.

```

TOP 10 WORDS FOR anime.txt
  anime 154
  de 50
  like 32
  que 31
  le 26
  want 22
  en 20
  scene 20
  e 19
  @paran@rml: 19

```

```

TOP 10 WORDS FOR artificial intelligence.txt
  artificial 152
  intelligence 135
  machine 46
  ai 32
  (ai) 29
  smart 27
  uses 27
  via 26
  table 25
  tennis 25

```

Task d)

To find the most common hashtags I iterated through all files in the corpus and gathered all tokens starting with a hashtag. I then again used FreqDist to find the most common ones for each file, and added them to a dictionary, summing those that were already there. The dictionary was then sorted, and the top 10 hashtags returned.

```
        #ai 142
    #machinelearning 69
        #analytics 64
#artificialintelligence 56
        #ml 49
        #nlp 49
    #opportunity 46
    #datascience 45
    #100daysofcode 40
        #jobs 40
```