UNIVERSITY OF SAN FRANCISCO
CHANGE THE WORLD FROM HERE

NLP on tweets: Analyzing twitter data about sexual abuse and assault

Prabha Nayak

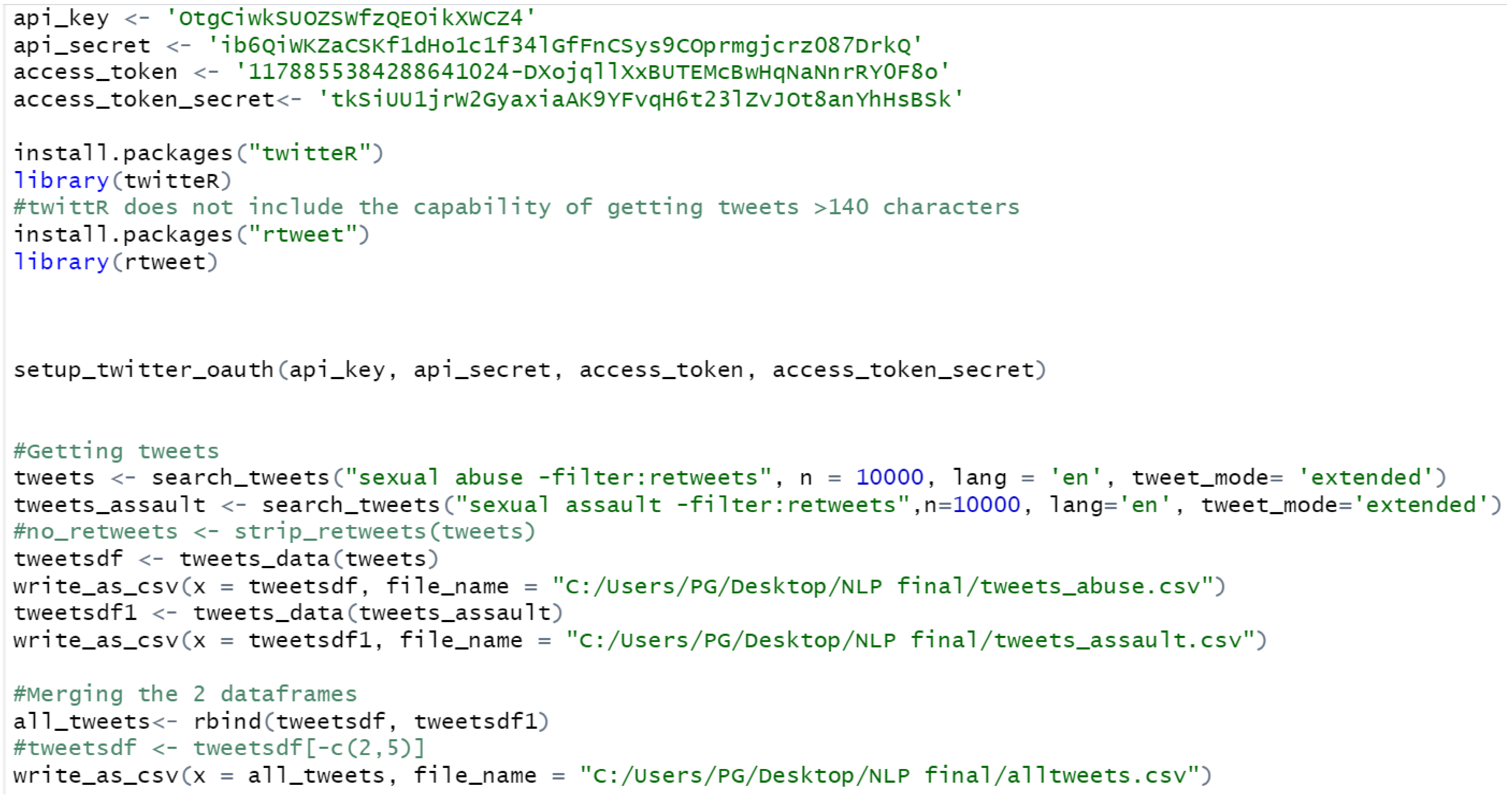
December 3, 2019

1. Accessing Twitter public API to obtain data

- Create a twitter account if you do not already have one.
- Go to https://apps.twitter.com/ and log in with your twitter credentials.
- Click "Create New App"
- Fill out the form, agree to the terms, and click "Create your Twitter application"
- In the next page, click on "API keys" tab, and copy your "API key" and "API secret".
- Scroll down and click "Create my access token“ and copy your "Access token" and "Access token secret".

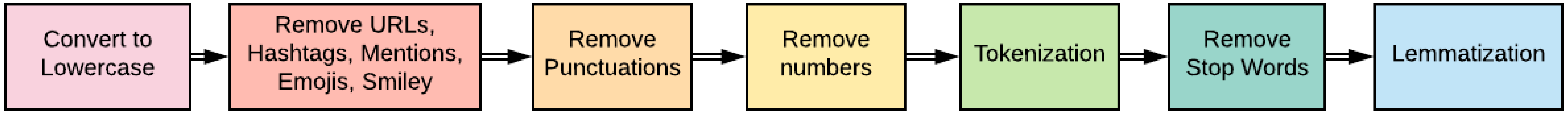
Number of tweets gathered = 19862

Number of features = 68



```r
api_key <- 'OtgCiwkSUOZSWfzQEOikXWCZ4'
api_secret <- 'ib6QiWKZaCSKf1dHo1c1f34lGfFnCSys9COprmgjcrzO87DrkQ'
access_token <- '1178855384288641024-DXojqllXxBUTEMcBwHqNaNnrRYOF8o'
access_token_secret<- 'tkSiUU1jrW2GyaxiaAK9YFvqH6t23lZvJOt8anYhHsBSk'

install.packages("twitteR")
library(twitteR)
#twittR does not include the capability of getting tweets >140 characters
install.packages("rtweet")
library(rtweet)



setup_twitter_oauth(api_key, api_secret, access_token, access_token_secret)


#Getting tweets
tweets <- search_tweets("sexual abuse -filter:retweets", n = 10000, lang = 'en', tweet_mode= 'extended')
tweets_assault <- search_tweets("sexual assault -filter:retweets",n=10000, lang='en', tweet_mode='extended')
#no_retweets <- strip_retweets(tweets)
tweetsdf <- tweets_data(tweets)
write_as_csv(x = tweetsdf, file_name = "C:/Users/PG/Desktop/NLP final/tweets_abuse.csv")
tweetsdf1 <- tweets_data(tweets_assault)
write_as_csv(x = tweetsdf1, file_name = "C:/Users/PG/Desktop/NLP final/tweets_assault.csv")

#Merging the 2 dataframes
all_tweets<- rbind(tweetsdf, tweetsdf1)
#tweetsdf <- tweetsdf[-c(2,5)]
write_as_csv(x = all_tweets, file_name = "C:/Users/PG/Desktop/NLP final/alltweets.csv")
```

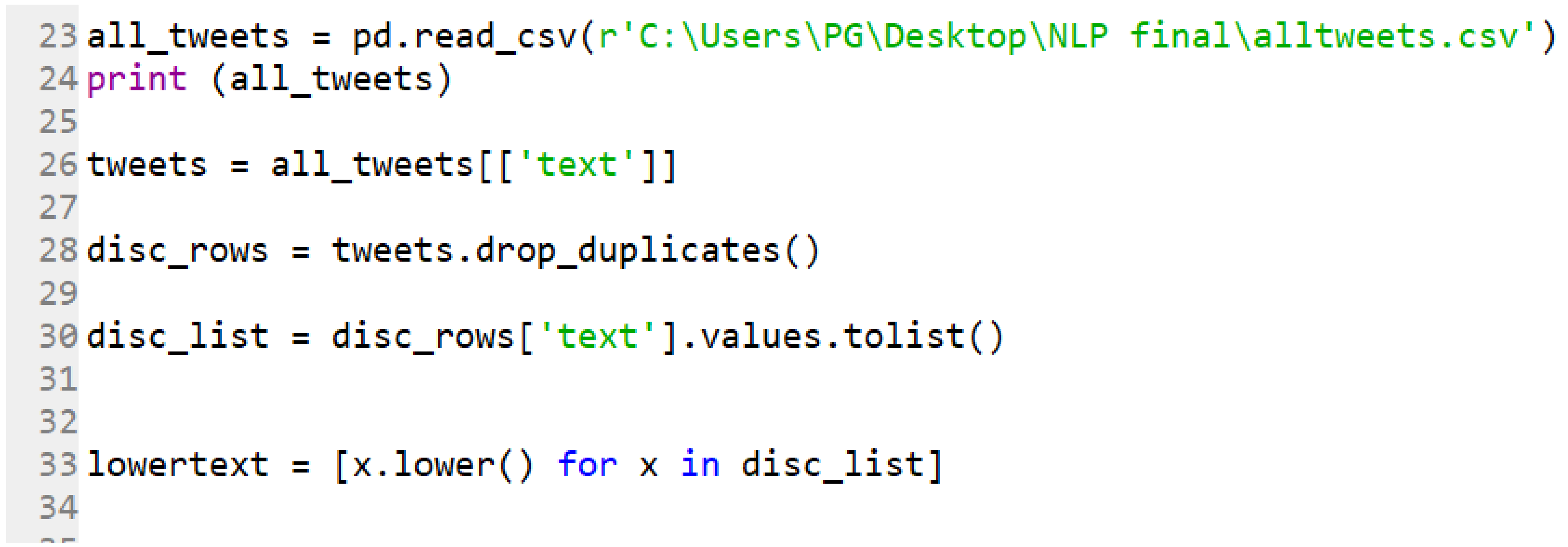
NLP Pipeline



2. Preprocessing of data

a. Convert to lowercase

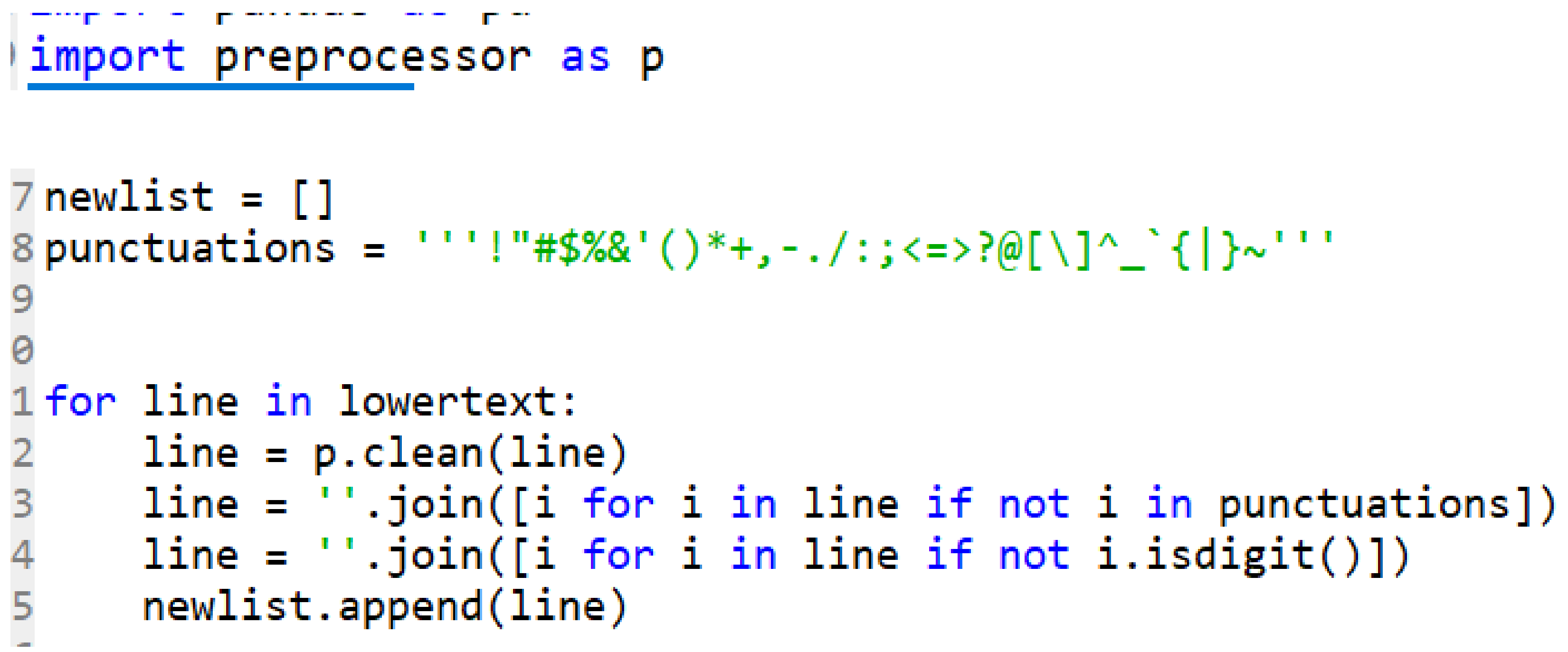
- Column ‘text’ extracted from the all_tweets dataframe
- Text converted to lowercase using lower()



```
all_tweets = pd.read_csv(r'C:\Users\PG\Desktop\NLP final\alltweets.csv')
print (all_tweets)

tweets = all_tweets[['text']]

disc_rows = tweets.drop_duplicates()

disc_list = disc_rows['text'].values.tolist()


lowertext = [x.lower() for x in disc_list]

```

2. Preprocessing of data

b. Remove URL, Punctuations and numbers

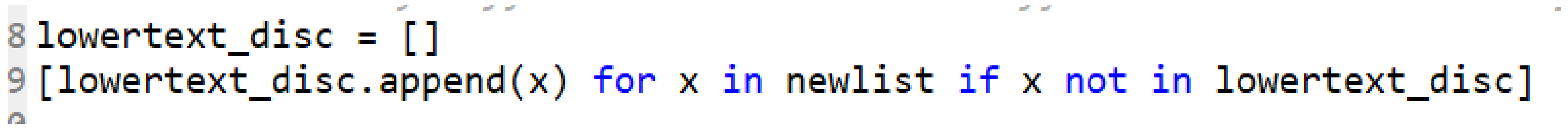


```
import preprocessor as p

newlist = []
punctuations = '''!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~'''


for line in lowertext:
    line = p.clean(line)
    line = ''.join([i for i in line if not i in punctuations])
    line = ''.join([i for i in line if not i.isdigit()])
    newlist.append(line)
```

2. Preprocessing of data

c. Remove duplicates

- Each tweets ends with a distinct URL.
- So removing duplicates preferred after removing the URLs from tweets.

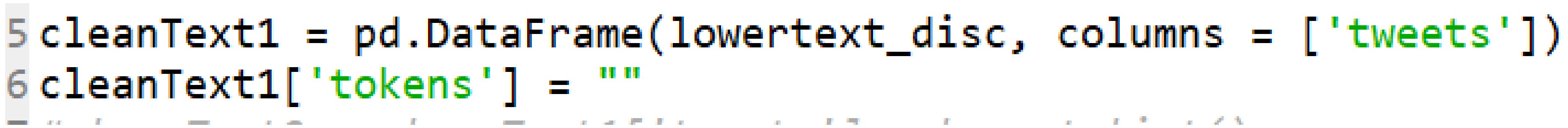


```
lowertext_disc = []
[lowertext_disc.append(x) for x in newlist if x not in lowertext_disc]
```

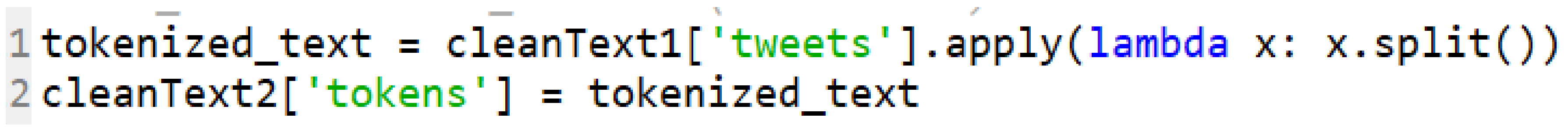
2. Preprocessing of data

d. Tokenization

- Tokenization using the split() method

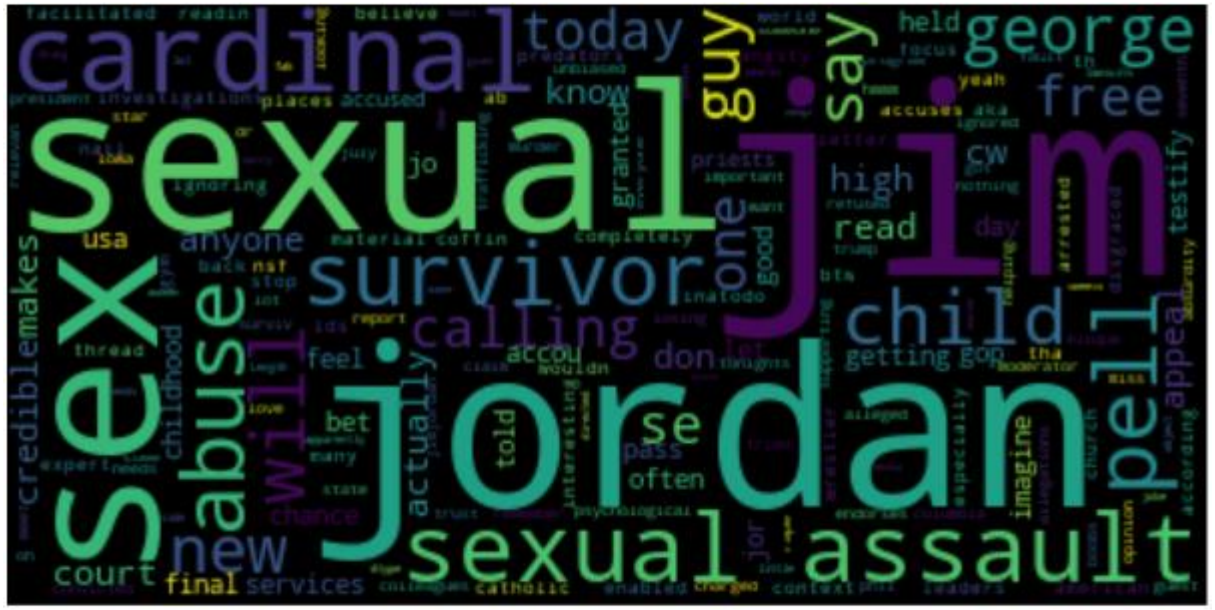


```
cleanText1 = pd.DataFrame(lowertext_disc, columns = ['tweets'])
cleanText1['tokens'] = ""
```



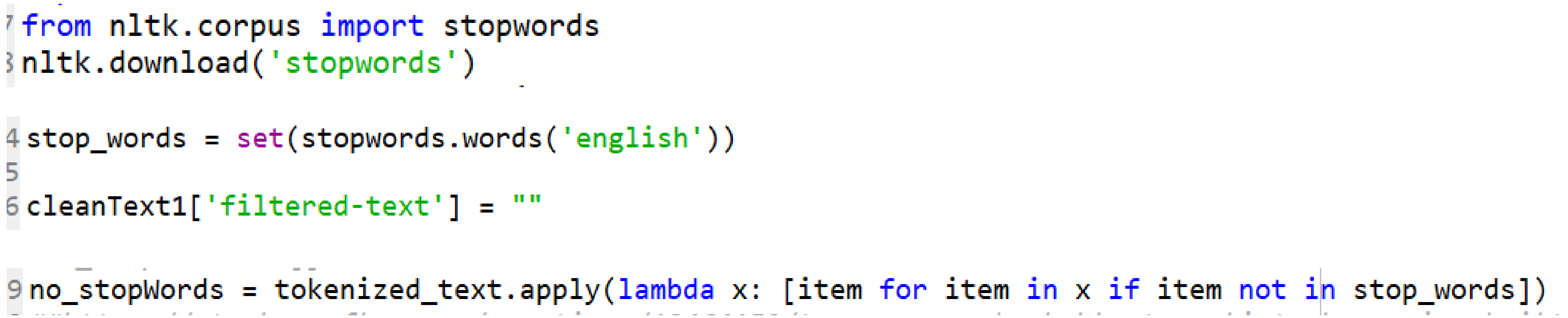
```
tokenized_text = cleanText1['tweets'].apply(lambda x: x.split())
cleanText2['tokens'] = tokenized_text
```

Word Cloud generation before



2. Preprocessing of data

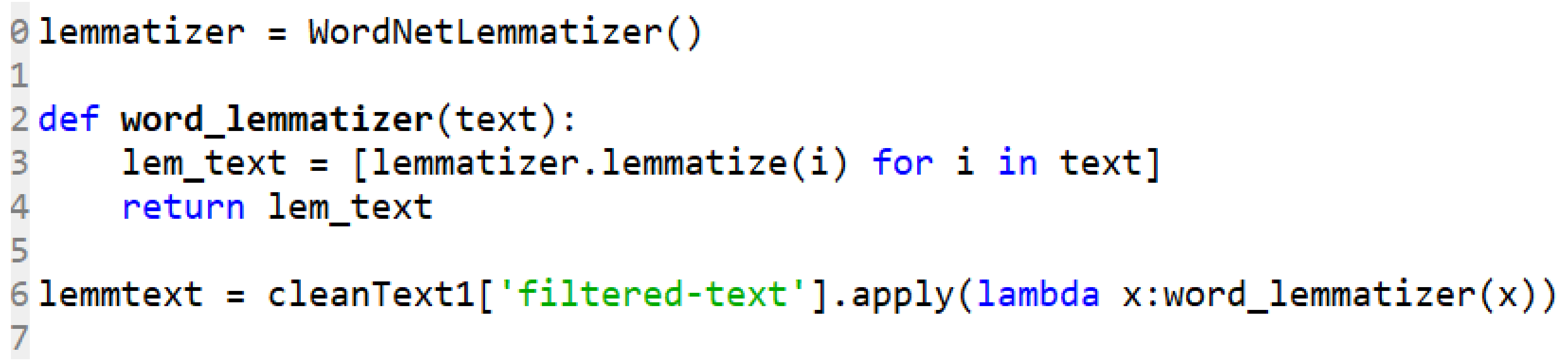
e. Removing Stop Words



```
from nltk.corpus import stopwords
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

cleanText1['filtered-text'] = ""

no_stopWords = tokenized_text.apply(lambda x: [item for item in x if item not in stop_words])
```

2. Preprocessing of data

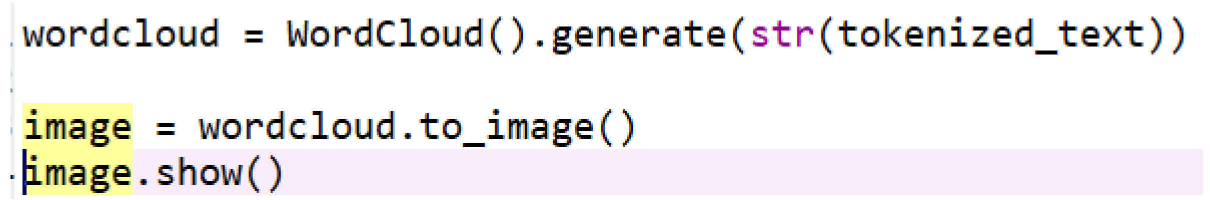
f. Lemmatization



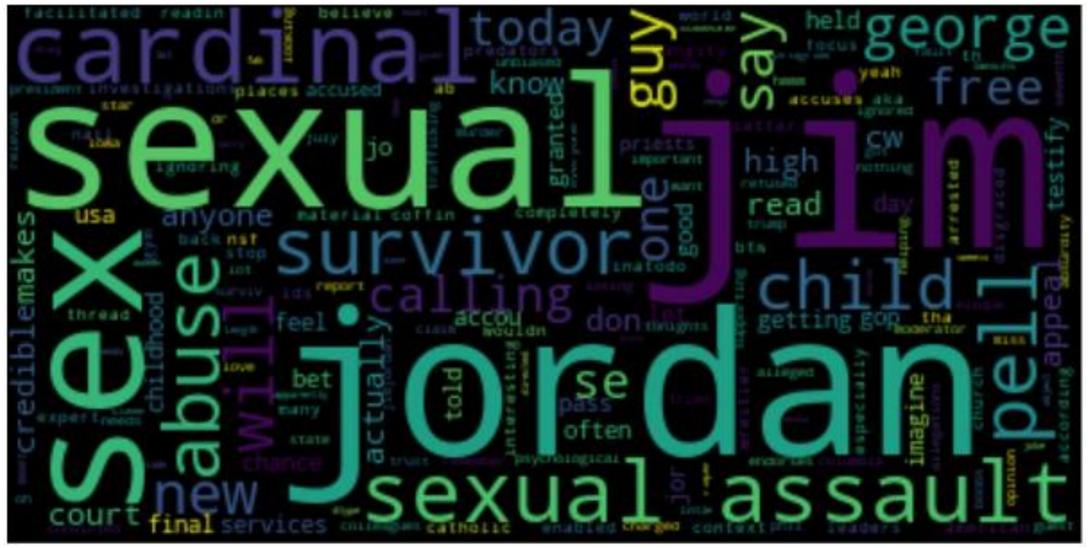
```python
lemmatizer = WordNetLemmatizer()

def word_lemmatizer(text):
    lem_text = [lemmatizer.lemmatize(i) for i in text]
    return lem_text

lemmtext = cleanText1['filtered-text'].apply(lambda x:word_lemmatizer(x))

```

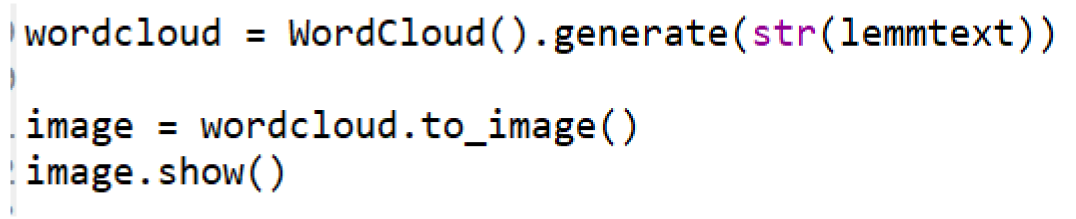
3. Analysis

a. Word Cloud generation

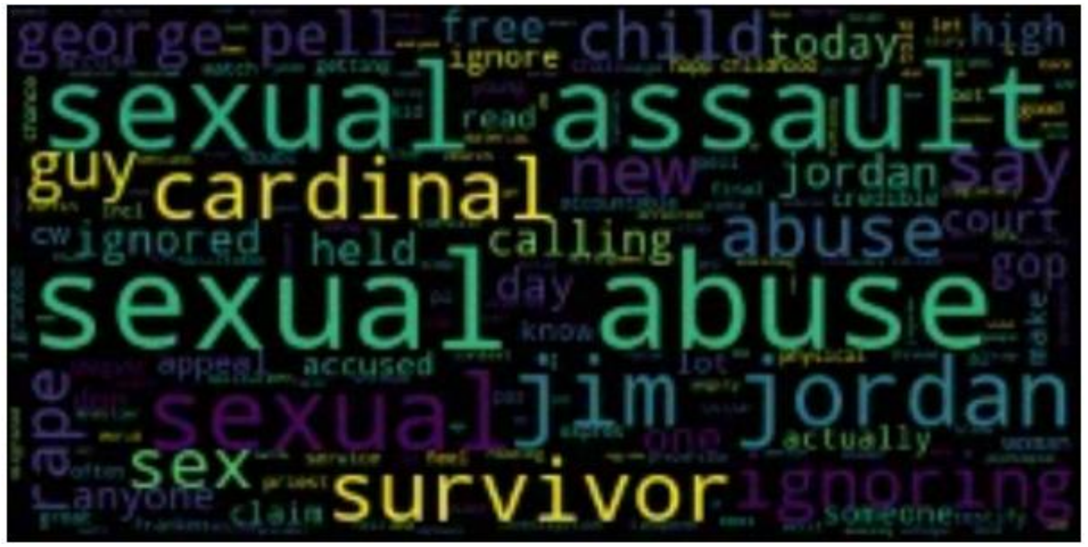


```
wordcloud = WordCloud().generate(str(tokenized_text))

image = wordcloud.to_image()
image.show()
```



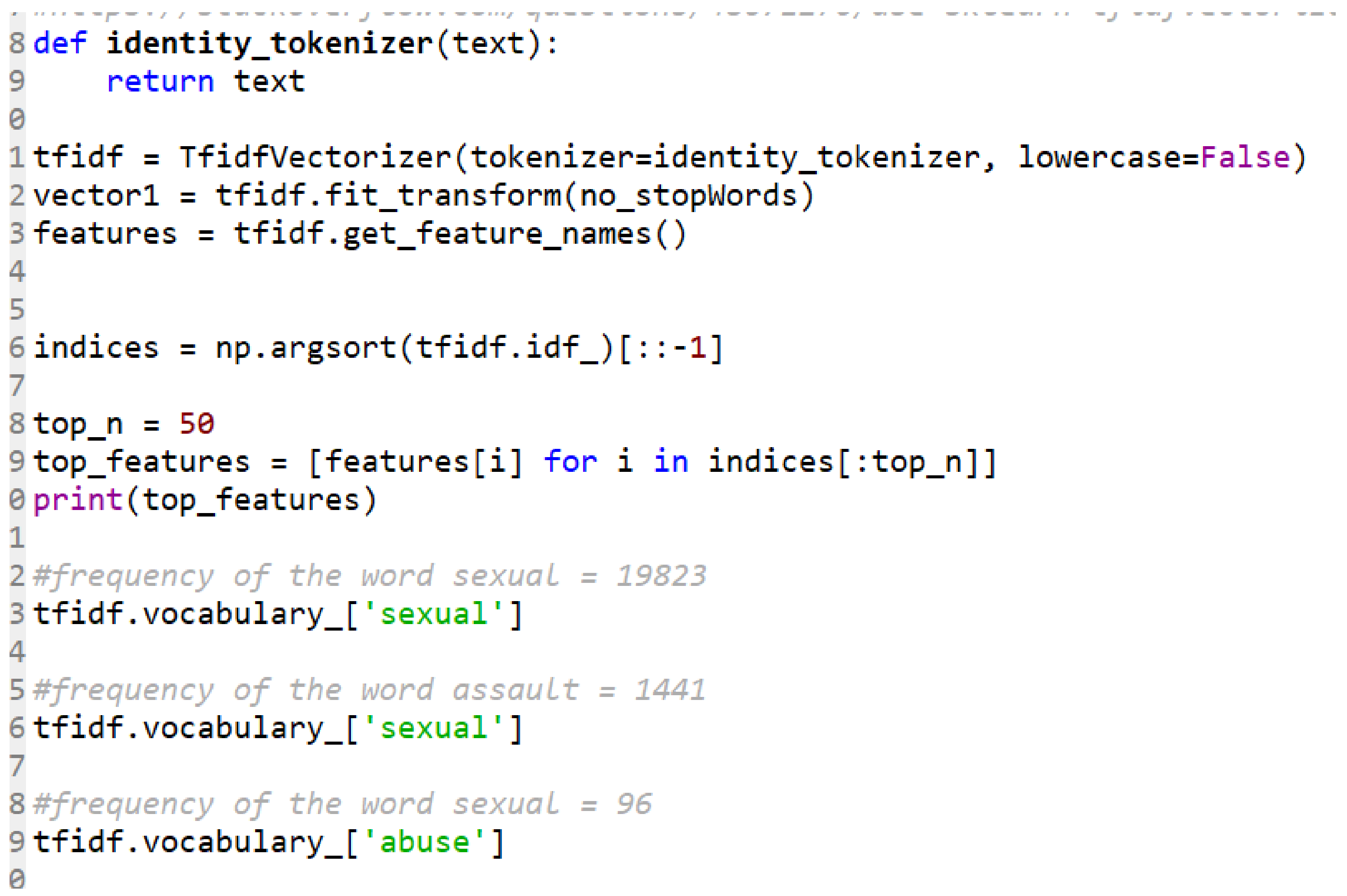


```
wordcloud = WordCloud().generate(str(lemmtext))

image = wordcloud.to_image()
image.show()
```



3. Analysis

a. TF-IDF



```
def identity_tokenizer(text):
    return text

tfidf = TfidfVectorizer(tokenizer=identity_tokenizer, lowercase=False)
vector1 = tfidf.fit_transform(no_stopWords)
features = tfidf.get_feature_names()


indices = np.argsort(tfidf.idf_)[::-1]

top_n = 50
top_features = [features[i] for i in indices[:top_n]]
print(top_features)

#frequency of the word sexual = 19823
tfidf.vocabulary_['sexual']

#frequency of the word assault = 1441
tfidf.vocabulary_['sexual']

#frequency of the word sexual = 96
tfidf.vocabulary_['abuse']

```

Next Steps

- Try n-grams
- Train-test split
- Fit training data to a model
- Fit the test data to the model to obtain results

References

1. https://stackoverflow.com/questions/34860982/replace-the-punctuation-with-whitespace
2. https://codereview.stackexchange.com/questions/115954/writing-lists-to-csv-file
3. https://gis.stackexchange.com/questions/72458/exporting-list-of-values-into-csv-or-txt-file-using-arcpy
4. https://stackoverflow.com/questions/13464152/typeerror-unhashable-type-list-when-using-built-in-set-function/13464194
5. https://towardsdatascience.com/nlp-for-beginners-cleaning-preprocessing-text-data-ae8e306bef0f
6. https://stackoverflow.com/questions/48671270/use-sklearn-tfidfvectorizer-with-already-tokenized-inputs
7. https://medium.com/fintechexplained/nlp-text-processing-in-data-science-projects-f083009d78fc
8. https://www.geeksforgeeks.org/generating-word-cloud-python/

Thank you!