CS445 – Project 2

Project Title: Text Exploration

Written by:Alper Bingöl

Instructor: Reyyan Yeniterzi

Date: 10.12.2020

1 WordCloud

The very first function "*createWordCloud*" where frequencies of words in a given collection is represented. Some required setups should be done. These are not only for WordCloud but also for the whole project.

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopWords = set(stopwords.words('turkish'))
# Start with loading all necessary libraries
import numpy as np
import pandas as pd
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import matplotlib.pyplot as plt
% matplotlib inline
from matplotlib import pylab

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline

from nltk.tokenize import sent_tokenize, word_tokenize
import warnings

#warnings.filterwarnings(action = 'ignore')

import gensim
from gensim.models import Word2Vec

import pickle
```

Figure 1: Setup for the Project

Sidenote: All the figures in the report are generated according to "Tsample5000.pkl" dataset.

The function initially checks if stopwords are wanted or not. If yes, it generates WordCloud after eliminating stopwords which is downloaded from nltk.corpus. The stopwords list is for Turkish. The width and height of wordcloud is multiplied with 100 intentionally. Otherwise the cloud would be too small. Background color is chosen as white. Design arrangments could be done by the users purpose. One important specification is using mask as a background shape. One example could be found in the continuation of the text.The second check is for weighting. It is either TDIDF or just TF. The term weight is calculated according to that. Another function is used for term weight. In this extra function, called termweight, CountVectorizer and TfidfTransformer are used. In CountVectorizer an attribute called maxdf is used to eliminate too common words in the collection in order to increase the amount of relevance. maxdf is choses as 0.5. It is an hyperparameter.

1.1 Examples

Figure 2: mode="TFIDF", stopwords=False, maxdf = 0.5

Since the stopwords generated from nltk is not too much, the difference between is not huge at the first observation. But there are small differences between figure 2 and figure 3.

In the figure 4 the TFIDF values are not used. So the amount of words which includes information is less then figure 2 and 3.

Figure 3: mode="TFIDF", stopwords=True , maxdf = 0.5

Figure 4: mode="DF", stopwords=True , maxdf = 0.5

Figure 5: mode="TFIDF", stopwords=True , maxdf = 0.3

In the figure 5 maxdf is 0.3. The number of eliminated words is higher than other clouds. While eliminating common words, the cloud now includes more set of words where we can take off more meaningful information.

Figure 6: Example of a usage of mask in WordCloud

2 Zipf's Law

This figure is drawed using the ranks and frequencies of words in the collection. It shows that term frequency decreases rapidly as a function of rank. The formula of the law:

$$f_t = \frac{k}{r_t}$$

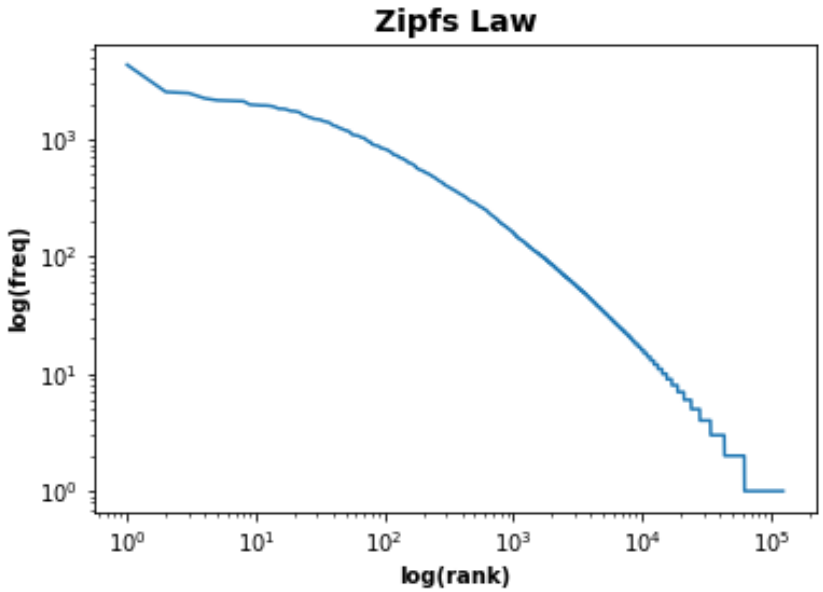

Figure 7: Zipf's plot

3 Heaps' Law

As the law indicates, the number of new words decreases as the size of the corpus increases. The formula of the law:

$$v = k \times n^{\beta}$$

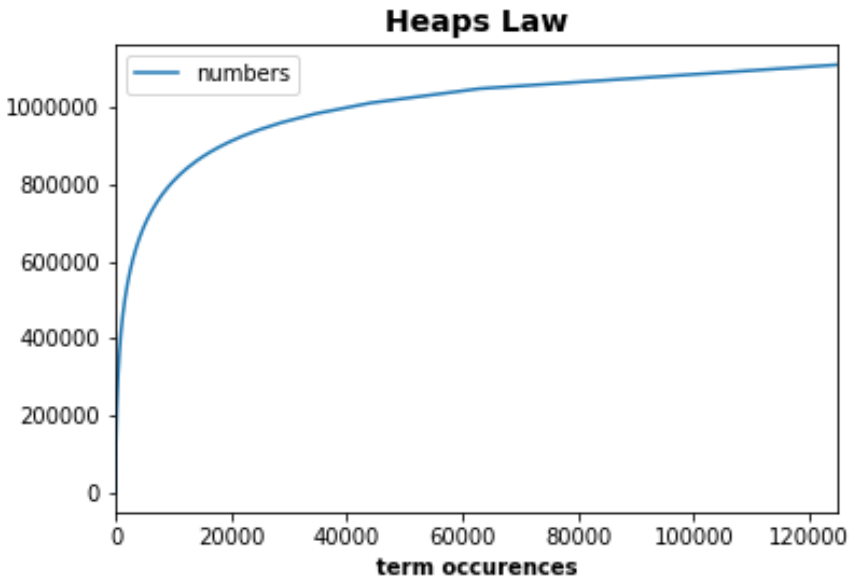

Figure 8: Heaps' plot

4 WordVectors

The function returns the models for trained word embeddings. The generated models are generated as the users desire. It is either CBOW or Skip-gram.

```
def create_WordVectors(Docs,dim,mode,size):


    counts = count(Docs)
    aa=counts.to_frame()
    cumulative_sum = aa.cumsum()
    cumulative_sum.rename(columns = {0:'numbers'}, inplace = True)
    cumulative_sum=cumulative_sum.reset_index()
    cumulative_sum.rename(columns = {"index":'words'}, inplace = True)
    unique_words=cumulative_sum['words']
    unique_list = unique_words.tolist()

    if mode == "cbow":
        model = gensim.models.Word2Vec(unique_list, min_count = 1, size = dim, window = size)

    else:
        model = gensim.models.Word2Vec(unique_list, min_count = 1, size = dim, window = size, sg = 1)

    return model
```

Figure 9: WordVectors Fucntion

5 References

https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/.X9KBsdgzYvj

https://towardsdatascience.com/simple-wordcloud-in-python-2ae54a9f58e5

https://amueller.github.io/word$_c loud/auto_e xamples/frequency.html$

https://www.datacamp.com/community/tutorials/wordcloud-python

https://scikitlearn.org/stable/modules/generated/sklearn.feature$_e xtraction.text.TfidfVectoriz$

https://scikitlearn.org/stable/modules/generated/sklearn.feature$_e xtraction.text.CountVectoriz$

https://matplotlib.org/api/$_a s_g en/matplotlib.pyplot.html$

https://pypi.org/project/wordcloud/

https://www.nltk.org/api/nltk.lm.html

https://pypi.org/project/gensim/