Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Text, Web and Social Media Analytics Lab

Prof. Dr. Diana Hristova

Exercise 3. Text Representation (1)

In this exercise we will derive the document representation of the preprocessed (stemmed) newsgroups dataset. We will apply sklearn as well as gensim to derive the Bag-of-words document representation. Those two packages are the standard packages for deriving the Bag-of-words document representation. We will calculate the following representations for each package:

- Absolute frequencies
- Relative frequencies
- TF-IDF frequencies

Finally, we will derive N-grams for the dataset.

1. Import the following packages:
    a. pickle
    b. pandas
    c. from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
    d. from sklearn.preprocessing import Binarizer
    e. from gensim.corpora import Dictionary
    f. from gensim.models import TfidfModel

2. Load the stemmed dataset from Exercise 2 by first mounting the drive and then using pickle.load(). Print the first entry of the data frame to assure that it was imported correctly.

3. **Bag-of-words (sklearn):**
    a. Apply the following transformations to the stemmed data using the *fit_transform()* function. What is each of them doing? What are the pros and cons of each approach?
        i. *CountVectorizer(max_df=0.95, min_df=0.05)*
        ii. *TfidfVectorizer(max_df=0.95, min_df=0.05,use_idf=False, norm='l1')*
        iii. *TfidfVectorizer(max_df=0.95, min_df=0.05, smooth_idf=False)*

        How many features are left in the dictionary in iii. and what are their names (Hint: Use *get_feature_names()*)? Do the features make sense? Are they different to those in i. and ii.? What happens if you change *max_df* and *min_df*?
    b. Apply *Binarizer()* to the result from a. i. using again *fit_transform().* What does the result stand for? What are its pros and cons?
    c. Put the frequencies for the **first document** from a. and b. in a data frame with a column *keys* containing the feature names from a. and four additional columns

with the corresponding frequencies from a. and b. (Hint: Convert the corpus to an array first.). Keep only those rows of the data frame with keys representing words that exist in the first document.

d. Sort the data frame (descending) on:
    i. TF-IDF Frequencies
    ii. Absolute Frequencies

   Do you see differences? Why?

4. Bag-of-words (gensim):

a. Create a corpus as input for gensim with *corpus_gen=[doc.split() for doc in data_stem]*. What is this command exactly doing?
b. Create a gensim Dictionary() based on the corpus. Call it *id2word*. Apply to it *id2word.filter_extremes(no_below=566, no_above=0.95)*. What is this method doing?
c. Apply the following operations. What is each of them doing?
    i. print(id2word.token2id)
    ii. print(id2word.token2id.keys())
    iii. print(id2word.dfs)

    iv. corpus1=[id2word.doc2bow(doc) for doc in corpus_gen]
    v. corpus2=[[(token[0],(token[1]/sum(n for _, n in doc))) for token in doc] for doc in corpus1]
    vi. corpus3=[[(token[0],1) for token in doc] for doc in corpus1]
    vii. tfidf=TfidfModel(dictionary=id2word, normalize=True)
    corpus4=[tfidf[id2word.doc2bow(doc)] for doc in corpus_gen]
d. Put all the frequencies from iv. to vii. in a data frame for the first document in the corpus. Add a column *keys* with the feature names. Compare the results with those from sklearn by merging the two data frames. Where do you think that the differences come from?

5. Ngrams:

a. Apply the following transformation to the stemmed data using *fit_transform()*: *CountVectorizer(ngram_range=(2, 2), max_df=0.95, min_df=0.05)*. What is it doing? Print the corresponding features. What are the differences to the features in Question 3. and Question 4? What happens if you change the values of *max_df* and *min_df*?
b. Put the result in a data frame with columns=feature_names. Print the data frame head and the maximum values for each word. What do you notice?