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

Text, Web and Social Media Analytics Lab

Prof. Dr. Diana Hristova

Exercise 6. Text Clustering

In this exercise, you will again analyse the Newsgroups dataset using clustering and topic modelling. We will do following steps:

A. Document preprocessing and representation

B. K-means clustering

C. LDA topic modelling

Part A:

1. Import the following packages:
   - import pandas as pd
   - from seaborn import heatmap
   - from sklearn.feature_extraction.text import TfidfVectorizer
   - from sklearn.cluster import KMeans
   - from sklearn.metrics import classification_report
   - import matplotlib.pyplot as plt
   - from gensim import corpora, models
   - from pyLDAvis.gensim_models import prepare
   - import pickle
   - import pyLDAvis

2. Load the Newsgroup data from https://raw.githubusercontent.com/selva86/datasets/master/newsgroups.json. Add a column ‘preprocessed’ to it consisting of the stemmed data from Exercise 2. Sort the data frame to contain only the following topics: 'soc.religion.christian', 'rec.sport.hockey', 'talk.politics.mideast', 'rec.motorcycles'. Apply sklearn’s tf-idf transformer to the whole dataset with max_df=0.7, min_df=0.1. Store the frequency matrix in *data_tfidf*. Store the feature names in a *words* list.

Part B:

3. Run the following code. What is it doing? What are we trying to achieve with this method?
   - *kmeans = KMeans(n_clusters = 4, max_iter=1000, random_state=42)*
   - *kmeans.fit(data_tfidf)*
4. Run the following code. What is it doing? Which cluster corresponds to which target name?
   - *common_words = kmeans.cluster_centers_.argsort()[:,-1:-11:-1]*

- *for num, centroid in enumerate(common_words):*
  print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))

5. Add a column to the data frame called *'cluster'* using kmeans.labels_. What is it doing?
6. Run the following code:
   - clusters = df.groupby(['cluster', 'target_names']).size()
   - fig, ax1 = plt.subplots(figsize = (26, 15))
   - heatmap(clusters.unstack(level = 'target_names'), ax = ax1, cmap = 'Reds')
   - ax1.set_xlabel('target_names').set_size(18)
   - ax1.set_ylabel('cluster').set_size(18)

   Does it confirm the results from 4.?

7. Add a column *pred* to the data frame setting its values to the target you would expect in 4. i.e. if you think that cluster 0 should correspond to topic rec.motorcycles, then all datapoints with df['cluster']==0 would get *pred* set to 8.
8. Print the classification report on target and pred. Are the results good?

Part C:

9. Create a gensim dictionary from the df['preprocessed'] (Hint: Remember to split the texts, see Exercise 2.). Call it *dictionary*. Remove rare and common words from it (no_below=118, no_above=0.95). Derive the absolute frequency matrix using gensim's doc2bow and store them in the list *corpus*.
10. Run the following code. What is it doing?
    - *lda = models.LdaModel(corpus, num_topics=4, id2word=dictionary, chunksize=10, iterations=100, passes=10, random_state=42)*

11. Run *lda.show_topics().* What is it dong? Assign to each topic one target from the dataset similar to Question 4.
12. Run the following code. What is it doing? Use it to redo Questions 6. to 8. for this model.
    - *topics=lda[corpus]*
    - *df['topics_lda']= [max(topics[i],key=lambda item:item[1])[0] for i in range(len(topics))]*

13. Run the following code and play around with the result:
    - *pyLDAvis.enable_notebook()*
    - *LDAvis_prepared = prepare(lda, corpus, dictionary)*
    - *LDAvis_prepared*