Methods of data mining

Project

Miro-Markus Nikula

712686

miro-markus.nikula@aalto.fi

Task1

a) I used Panda's dataframe-methods to combine the title and abstract for every document. I then removed stopwords, digits and individual symbols. The stopword-list I used can be found here https://gist.github.com/sebleier/554280 and it contains 127 English stopwords.

b) For calculating the TF-IDF-variants for each word in every document I used Sklearn's TfidfVectorizer. It takes as a parameter a list of strings and returns a matrix of normalized TF-IDF-features. The result was a very sparse matrix with 13913 columns. The code for steps a and b can be found in the zip-file in prepro.py.
The formula used to calculate TF-IDF:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Images from an article of Mayank Tripathi (6/2018) available in:
https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/

c) I clustered the data of TF-IDF-features into 5 clusters using Sklearn's KMeans and outputted a list of every document with their respective cluster. The clustering step is in clustering.py.

d) Using the version of Strehl and Gosh (2003) of normalized mutual information, I calculated in mutual_info.py an average NMI value of **0.68** for the clustering. The result in my opinion is surprisingly good considering that there were almost 14000 dimensions in the data, but when comparing the distribution between different clusters and the five different classes, I noticed that the clustering corresponded to the class division quite well.

Task2

To get a better clustering result, I started by trying to reduce the dimensions of the data to be clustered. In the original data after removing stopwords etc. I was left with 13913 different words. I tried lemmatizing the words by using WordNetLemmatizer from Pyhton's nltk-module (natural language tool kit). The lemmatizer can for example transform a plural of a word to the single form (mice -> mouse) and it can transform verbs to their basic form (moving -> move). Adding this step made the preprocessing stage a lot heavier and it now took about 20 seconds to run, but I was able to reduce the number of different words to 11019. Besides adding word lemmatizing to preprocessing the methods in the whole pipeline remained the same. As a result, I was able to increase average NMI to **0.71**. Not a very significant improvement in my opinion considering that nearly 3000 words were dropped.

I tried spectral clustering on the data, but could not improve the result, in fact, the result was much worse this time as I got an NMI of **0.58**. I could however get a better result using spectral clustering after reducing the number of dimensions using PCA. With using 75 dimensions instead of the previous 11019 I was able to get an NMI of **0.69.** This is quite curious since 75 dimensions seemed to preserve only 22.58% of the variance in the data and when trying with for example 1000 dimensions where over 90% of the variance was preserved, the clustering wasn't nearly as good.

In addition to KMeans and Spectral clustering I also tried Agglomerative clustering with “Ward”-linkage but could not get any results worth mentioning.

	KMeans (as instructed)	KMeans + word lemmatizing	Spectral + word lemmatizing	Spectral for 75 PCA + word lemmatizing
NMI	0.68	0.71	0.58	0.69

All of the NMI values are averages of multiple clustering results.

Task3

I tried to determine the different cluster topics of the best clustering result I had obtained. My best result is with using KMeans-clustering on lemmatized word data. For this I used Counter-object from Python’s Collections module to count the most frequent words of each cluster. Below is a list of the most common words (with the word frequency) that also in my opinion best described the cluster in question.

Cluster				
0	Robot (488)	System (430)	Control (289)	
1	Image (635)	Detection (370)	Learn (329)	Network (317)
2	Database (665)	Relational (313)	Query (274)	SQL (92)
3	Compiler (442)	Program (356)	Language (232)	Code (213)
4	Security (429)	Quantum (325)	Key (278)	Cryptography (237)

When looking at the most common words by cluster there are many words related to the same category. For example, cluster 0 seems to include mainly documents covering robotic systems and their control, cluster 1 could be about computer vision and applying deep learning networks in image detection, cluster 2 includes documents about relational databases and (SQL)-queries, cluster 3’s documents cover different programming languages and their compiling and in cluster 4 there can be seen words related to information security and cryptography and apparently calculating encryption keys using quantum computers. Of course, as the NMI-value also suggests there must be many documents that were misclassified, but the clustering was good enough to clearly distinguish these five different topics.

Appendix

To reproduce my results these steps should be followed:

Make sure to have the following dependencies installed:

-pandas

-sklearn

-ntlk

The following are not required for reproducing my best result, but are used for pca and counting the most common words:

-numpy

-collections

I have separated the processing into three different python files: prepro.py, clustering.py and mutual_info.py. First prepro.py should be ran. It uses the stopwords.txt-file (included in the zip), but

```
import nltk
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

could be used instead to produce a different list of stopwords.

Prepro.py produces an output csv-file (line 60) that will be used in clustering.py (line 11). After running prepro.py, run clustering.py. KMeans is the default clustering algorithm, but spectral or agglomerative clustering can also be used by commenting out the part with KMeans. clustering.py outputs another csv-file (line 31) that will once more be used by mutual_info.py (line 8). Once ran, mutual_info.py should print the NMI value of the clustering in contrast to the class-values in the original absdata.csv-file.