CS-E4650 - Course project

Arttu Häkkinen (596077)
Eero Surakka (597791)

December 22, 2021

1 Methods

In this section, we report all methods we tried in our own experiments. Different parts of the methodical experiments are divided into smaller subsections.

1.1 Data preprocessing

The data was loaded as Pandas [1] data frame object using the symbol " as a delimiter. Next, the columns containing title and abstract were combined into one column by delimiting the original strings with one empty space. The string values of this combined column are henceforth referred to as documents.

The next part of the preprocessing was cleaning the documents. This was done in a loop where each document string was processed as follows: First, the documents were lowercased. After this, they were trimmed by replacing hyphens with empty spaces and tokenized [2] into lists of words. As next step the tokenized documents were filtered by removing the stop words [3], punctuation [4], numeric values and the expressions containing any special characters (regular expressions).

Finally, the otherwise preprocessed documents were stemmed using three different stemming algorithms: Snowball stemmer [5], Lancaster stemmer [6] and Porter stemmer [7]. We also experimented with lemmatization [8].

1.2 Data representation

We represent the data in bag-of-words form. The bag-of-words data frame is formed by looping through the tokenized documents and counting the frequencies for each unique token while adding new columns for the tokens that haven't been introduced to the data frame yet. After this loop, the bag-of-words data frame consists of rows indexed with document index and columns named by each preprocessed word in the database. Each element in the data frame is an integer expressing the term frequency.

In order to use Euclidean distance later, we needed to modify the term frequency bag-of-words data frame to normalized term frequency - inverse document frequency (tf-idf) form. We tried two different tf-idf variants.

The first one we refer as normalized term frequency. This was computed for each element in the frequency bag-of-words data without any scaling as

$$tf_{normalized}(\boldsymbol{d}_i, w_{ij}) = \frac{fr(w_{ij}|\boldsymbol{d}_i)}{N_{\boldsymbol{d}_i}}, \tag{1}$$

where $fr(w_{ij}|\boldsymbol{d}_i)$ is the frequency of how many times word $w_{ij}(i = 1, ..., N_{\boldsymbol{d}}; j = 1, ..., N_{\boldsymbol{d}_i})$ appears given the document $\boldsymbol{d}_i$, $N_{\boldsymbol{d}}$ is the total number of documents in the data base, and $N_{\boldsymbol{d}_i}$ is the length of the preprocessed and tokenized document $\boldsymbol{d}_i$.

The other variant we experimented with was more sophisticated version of the first one. In addition to just normalizing with the document size, the normalized value is scaled with the logarithm of the inverse document frequency. This was computed for each element in the original frequency bag-of-words data as

$$tfidf(\boldsymbol{d}_i, w_{ij}) = \frac{fr(w_{ij}|\boldsymbol{d}_i)}{N_{\boldsymbol{d}_i}} \cdot log \frac{N_{\boldsymbol{d}}}{N_{\boldsymbol{w}_{w_j} \neq \boldsymbol{0}}}, \tag{2}$$

where $N_{\boldsymbol{w}_{w_j} \neq \boldsymbol{0}}$ is the total number of non-zero elements in the column vector corresponding to word w_{ij}.

1.3 Feature extraction, selection and dimension reduction

The method we tried to extract features with, was forming N-grams. We created the bi- and trigrams using [9] by first computing the occurrences of word pairs in the documents, and then replacing N individual words as their N-gram if they occurred sequentially. We experimented by forming 10 to 300 most frequent N-grams found.

As feature selection, we experimented filtering out infrequent words. Experiments were conducted by filtering individual words that occurred less or equal than 1, 5 and 10 times.

As dimension reduction, we tried Latent Semantic Analysis (LSA) [10]. To make reproducing the results more consistent, we used 'arpack' as the parameter value for algorithm to remove unnecessary stochasticity. The discussion about the methods used to optimize the number of preserved dimensions is provided in subsection 1.5. The rest of the parameters were set as default.

1.4 Clustering methods, parameter settings and similarity measures

The K-means clustering was applied using [11]. The parameter settings were set as default, except the number of clusters parameter was set to $K = 5$ which will be the same for the next two clustering methods too, since we had 5 classes in the original data.

The Hierarchical clustering was applied using [12]. The parameters we played with were all the four possible linkage criterion (Ward's, single, complete and average) and affinity (cosine and euclidean). Rest of the parameters were default.

We applied Spectral clustering using [13]. The label assigning after the Laplacian embedding done in the software package computation was set to 'discretize', as we wanted our clustering to be as sensitive as possible to variation in results produced by the random initialization. Additionally, we experimented with all different affinity options possible. Our experiments optimizing the kernel coefficient γ are reported in subsection 1.5.

1.5 Determining optimal parameters

The parameters we optimized were γ for clustering, and the number of dimensions preserved in LSA. For both optimizations the strategy was similar. First we used greater step size to find the interval for the parameter value that yielded the best evaluation metric. After that, the smaller interval was combed with more dense search and the parameter value that yielded the best evaluation metric from that smaller interval was selected.

1.6 Extra: Black-box embedding

In addition to experimenting with the above-mentioned methods introduced during the course, we wanted to try one state-of-art deep learning approach applied via software package [14]. It works as a black-box method where we only feed the combined document strings as a as input to the computational model and the embeddings come out as output.

2 Results

In this short section we provide the results of our experiments. The evaluation metric used was the Strehl and Ghosh version of the Normalized Mutual Information (NMI) [15] where we set the average method as 'geometric'. In total, we tested hundreds of different combinations of methods and parameters to find our own approach. We are not going to explain all the experiments done, but we can assure that every possible combination of methods described in section 1 was tested. The achieved NMI values for the baseline method, our own white-box method and the black-box embedding method are reported below in the table 1. The baseline method is the exact method described in the project handout with Lancaster stemmer [6].

As our own white-box approach we tested the best parameter–method combination to be the following: Data was preprocessed as described in section 1.1, using the Lancaster stemmer [6]. The data was represented in tf-idf bag-of-words form computed with equation 2. We decided to not use N-grams as they yielded bad results for this data, and filtered out infrequent words that occurred only one time in the whole database. Then we applied LSA dimension reduction to present the data in 130 dimensions. Finally, clustering the data was executed with Spectral clustering [13] using 'rbf' as affinity and $\gamma = 1.03$. Optimization plots for LSA dimensions and γ can be found from Appendix B.

In the black-box method the embeddings were formed as described in 1.6. After that, K-means clustering [11] with default parameters was applied to embedded data.

	Baseline	White-box	Black-box
NMI	0.64	0.79	0.82

Table 1: NMI scores with baseline, white-box and black-box methods.

Based on the results we would choose black-box clustering based on its performance – unless if it would be important to have a model that is completely explainable, we would suggest the white-box approach since it also provided good clustering.

The main topics of the five clusters are listed below. They are inferred based on the most frequent key words in separate clusters we got with the white-box clustering model. Top 20 key words of each clusters can be found from the Appendix C.

- id 0: Articles focused on relational databases since the key words "database", "relational", "query", "analysis" and "information" appear.
- id 1: Articles about deep learning and computer vision since key words like "computer", "vision", (neural) "network", "performance" and "learn" appear.
- id 2: Articles about cryptography, security and encryptic algorithmic attacks since key words "cryptography", "encrypt", "algorithmic", "attack", "security" and "system" appear.
- id 3: Articles about robot design and development since key words "robot", "control", "design" and "develop" appear.
- id 4: Articles about computational programming and compilers since key words like "compiler", "program", "compute", "language", "code" and "optimize" appear.

After checking the key word lists for each cluster, we still found stop words for scientific research paper context like "use", "based" and "paper" which could have been filtered out before clustering phase to reach even better results with the white-box model. Within the time scope of this project we didn't have time to do that, and were happy with black-box model results since they topped the set requirements of greater than 0.81 NMI.

References

[1] https://pandas.pydata.org/

[2] https://www.nltk.org/api/nltk.tokenize.html

[3] https://pythonspot.com/nltk-stop-words/

[4] https://www.geeksforgeeks.org/string-punctuation-in-python/

[5] https://pypi.org/project/snowballstemmer/

[6] https://www.nltk.org/api/nltk.stem.lancaster.html

[7] https://www.nltk.org/api/nltk.stem.porter.html

[8] https://www.nltk.org/api/nltk.stem.wordnet.html

[9] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[10] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

[11] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[12] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

[13] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html

[14] https://pypi.org/project/sentence-transformers/

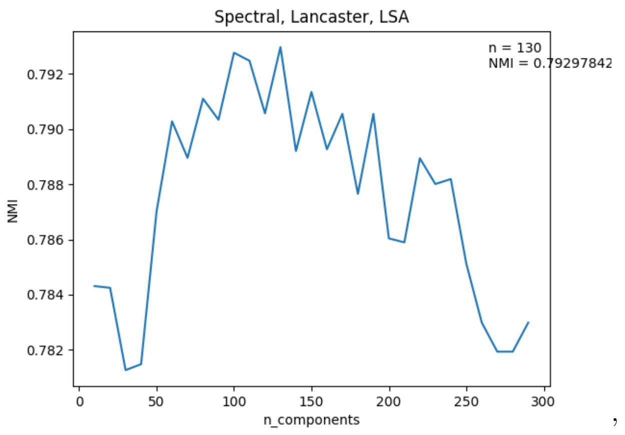
[15] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html

Appendix A: Execution instructions

Our project contains 3 files, baseline.py, advanced.py, and extra_blackbox.py. These can be run by typing the following command to terminal: ”python3 <filename>”

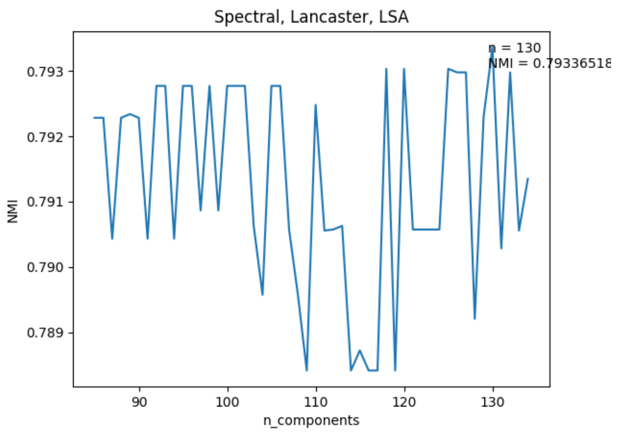
The following packages are required for running the programs: re, matplotlib, string, sklearn, pandas, numpy, snowballstemmer, nltk. Additionally the nltk packages might need some local downloading but the instruction will be presented in terminal if needed.

Appendix B: Parameter optimization plots

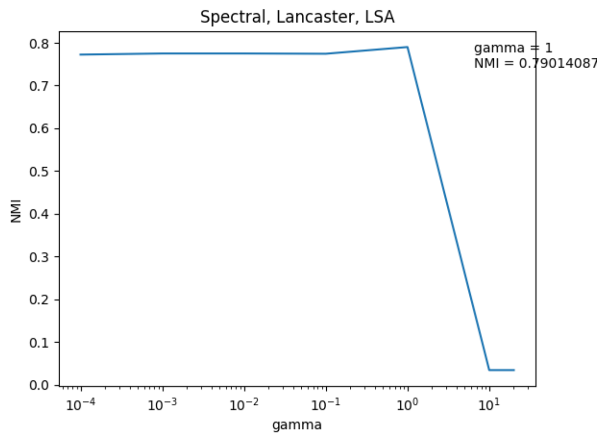


,

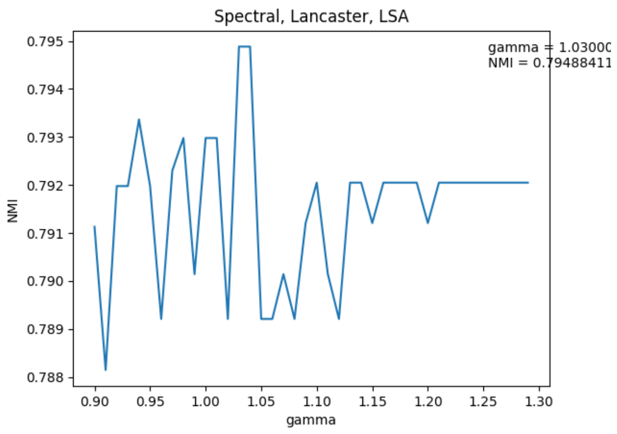
We searched optimal dimensions to be preserved in LSA (n), first by running n from 10 to 300 with step size of 10.



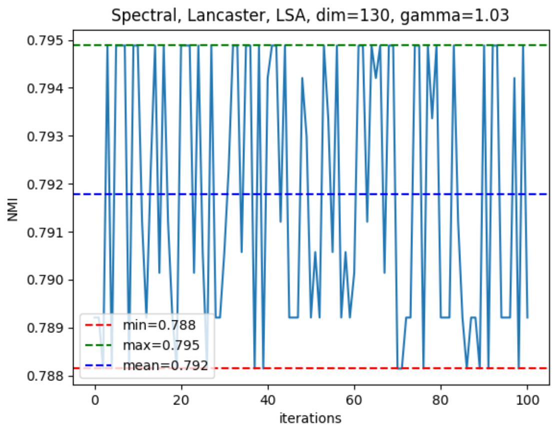
Then we searched more densely the n range from 80 to 135 with step size 1. We selected the best n found (n = 130).



We searched for optimal γ, first by running gamma from 0.001 to 20 with step size of logarithmic step sizes.



Then we searched more densely the γ range from 0.9 to 1.3 with step size 0.01. We selected the best γ found ($\gamma = 1.03$).



Finally, because some is present, we ran the clustering with the selected parameters ($\gamma = 1.03$, n = 130) for 100 times and calculated the mean NMI as the final result for our experiments.

Appendix C: Top 20 keywords in the clusters and their occurance

Cluster id: word and its count

- 0: us 296, bas 273, vis 239, propos 223, method 217, comput 208, result 207, im 196, perform 190, model 183, learn 176, detect 176, acc 173, apply 169, system 166, pap 164, network 155, dat 147, expery 138
- 1: compil 191, us 170, program 153, pap 136, bas 133, comput 119, pres 111, optim 105, apply 103, gen 96, impl 95, process 95, system 94, langu 92, cod 91, high 89, perform 86, tim 86, show 84
- 2: us 196, sec 187, propos 167, bas 164, cryptograph 156, pap 128, key 121, comput 121, perform 101, encrypt 101, algorithm 100, system 99, dat 99, provid 93, efficy 93, apply 92, attack 91, impl 91, analys 87
- 3: robot 191, us 171, bas 124, system 121, control 110, perform 108, result 105, expery 102, pres 100, propos 91, develop 88, design 88, diff 77, approach 76, demonst 75, model 75, method 74, apply 72, work 72
- 4: label 195, databas 183, rel 158, us 152, dat 150, system 113, bas 104, pap 91, propos 86, query 84, inform 81, approach 78, apply 76, man 74, model 74, result 73, process 72, pres 68, analys 67, provid 65