Lukic, Bojan
Student ID: 204443IV

1 Problem and feasibility study for the final project

1.1 Precise problem statement

The research of topic detection is very prominent. A well-known example of topic detection is the detection of the field of research for scientific papers. Yearly, thousands of papers are published covering a wide range of topics such as Data Mining, Machine Learning, and many more. These topics can be further divided into sub-topics, which can in turn be divided into specific keywords. Usually, the trivial problem of classifying research papers is performed manually, however, most researchers are more interested in subcategories of classified papers than the broader topics the papers are discussing. For instance, a reader of a scientific paper might be interested in Neural Network Activation Functions or Overfitting which are subtopics within the field of Machine Learning.
Ideally, a person analyzing texts will find important texts from their topics of interest.

1.2 Necessity of techniques chosen to solve the problem

One of the main techniques used in this project is keyword clustering. Keyword clustering is a necessary technique which clusters bags of words taken from an article into clusters of words which are the most distinct compared to other articles in a corpus. Another technique is to observe the words with the highest occurences taken from an article, to understand which words are the most important ones.

1.3 Describe the data set and explain why this dataset possesses necessary information to solve the problem. Present example of the data point

The dataset used for this project was taken from https://archive.ics.uci.edu/ml/index.php. The dataset consists of more than 300 unlabeled science articles in *.txt fromat. They cover various scientific topics such as „Spectral Clustering Methods“ or „Reinforcement Learning“. This vast difference in topics makes these text suitable for this project. In total, 20 random texts are taken from the corpus of 300 texts. The following is a sample showing the structure of one of the texts:

ABSTRACT
THIS PAPER ADDRESSES THE PROBLEM OF DISTRIBUTED LEARNING UNDER COMMUNICATION CONSTRAINTS, MOTIVATED BY DISTRIBUTED SIGNAL PROCESSING IN WIRELESS SENSOR NETWORKS AND DATA MINING WITH DISTRIBUTED DATABASES AFTER FORMALIZING A GENERAL MODEL FOR DISTRIBUTED LEARNING, AN ALGORITHM FOR COLLABORATIVELY TRAINING REGULARIZED KERNEL LEAST-SQUARES REGRESSION ESTIMATORS IS DERIVED. [...]
INTRODUCTION
IN THIS PAPER, WE ADDRESS THE PROBLEM OF DISTRIBUTED LEARNING UNDER COMMUNICATION CONSTRAINTS , MOTIVATED PRIMARILY BY DISTRIBUTED SIGNAL PROCESSING IN WIRELESS SENSOR NETWORKS (WSNS) AND DATA MINING WITH DISTRIBUTED DATABASES. [...]

Lukic, Bojan
Student ID: 204443IV

2 Preprocessing

For preprocessing of the text data, several function have been implemented.
The first function takes a text and clears it from stopwords, punctuation marks, and other unneeded characters. Additionally, words are lowercased, split up into single elements, and stemmed. For stem completion, an online dictionary containing a vast amount of English words is used. Adverbs (such as the word „though“) are also removed to reduce the text data to the most essential and most informative words.
Lastly, because in the preprocessing of texts words are split into single elements, it is possible to combine the words back into one string with an additionally implemented function. The third and last function can split a text containing multiple words back into a vector with single words.
To compare the similarities between texts before and after preprocessing, the Jaccard measure is used. The Jaccard measure calculates the intersection of elements divided by the union of elements of two datasets. The results for two arbitrary texts taken from the corpus shows a slightly higher Jaccard measure for the processed text compared to the unprocessed one. This suggests that the texts might contain stubs (such as single characters, symbols, etc.) which differ for both texts and which are removed by preprocessing the data. Therefore, the two texts will contain more similar content after processing, explaining the slightly higher Jaccard measure for the processed text. The Jaccard measure for the chosen texts is 0.11 before preprocessing and 0.15 after.

3 Scoring the text data

For scoring the texts from the corpus the „tf-idf“ model is used. The statistic tf-idf identifies words that are important to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. After scoring all 20 texts, the top-n tf-idf score for every word from each text can be displayed. Figure 1 shows the tf-idf scores for words from a selection of texts.

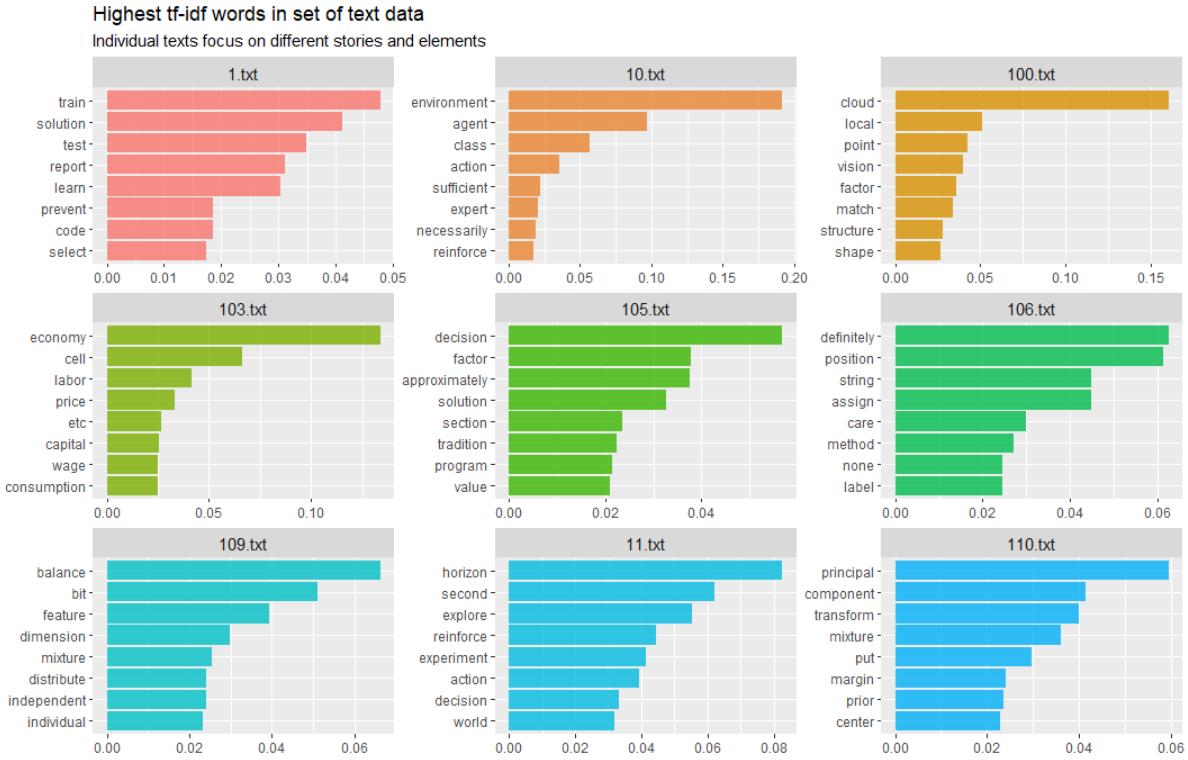

Figure 1 - tf-idf scores for words contained in 9 different texts

Lukic, Bojan
Student ID: 204443IV

After scoring all 20 texts with the tf-idf model, one can already get an idea of the topics each text is covering. For instance, the words from the file „103.txt“ with the highest tf-idf scores are „economy“, „price“, „capital“, and „wage“. This suggests that the text is about economy and finance.

Additionally, the number of topics can be condensed to a smaller number. By reducing the number of topics to 6, the probabilities of the top-n words contained in one of the 6 topics can be visualized, such as in the following figure:

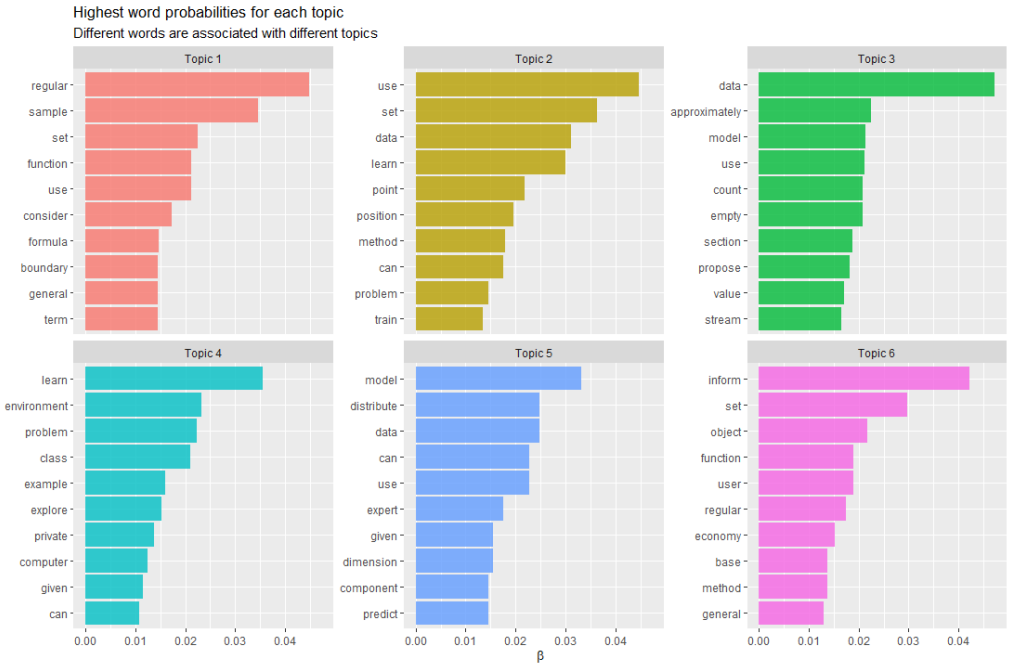

Figure 2 - Highest word probabilities for each of the 6 topics

Lastly, the distribution of document probabilities for each topic can be computed. „td_scores“ computes the probability that each document in a corpus is generated from each topic and shows which topics are coming from which documents. In this case, each text is strongly associated with a single topic, as seen here:

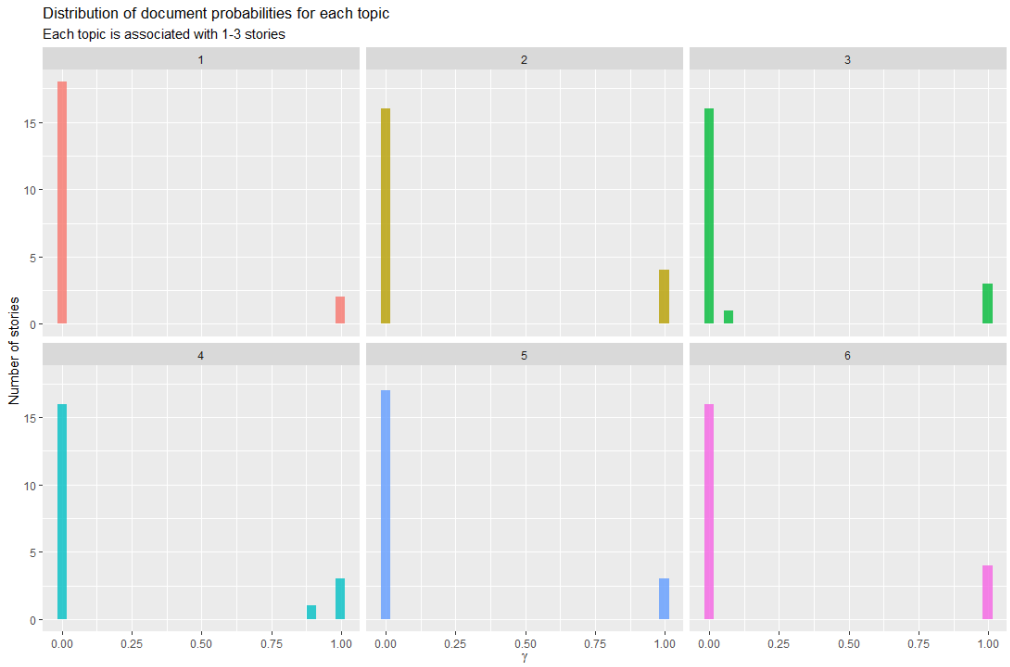

Figure 3 - Distribution of document probabilities for each topic

Even though the first results already seem promising, further steps are taken to analyze the texts.

4 Vectorizing, Clustering, and Decision Boundary

In this section, the texts will be further analyzed by first vectorizing and then clustering the words. In the end, an example of a set of clustered words with corresponding decision boundaries (using k-nearest-neighbors) will be shown.

4.1 Word2vec

Word2vec is a function that creates a model from a corpus mapping different parameters for each word in a text. The underlying algorithm works with a neural network model to learn word associations from a corpus of text. After creating the model, 2 words from one text are selected for similarity measures. The similarity measures are between the selected words and the remaning words from the same text.
Here, for each text the two words with the highest tf-idf scores are selected as similarity references. The idea is that the words with the highest tf-idf scores are the most unique ones to their text. Running similarity measures on these words will, in theory, make it possible to extract additional words that are related to the topic the two words with highest tf-idf score are reflecting.
In this example, the similarity measure is run on the file „117.txt“. The two words with the highest tf-idf score from that document are „support“ and „position“. After running the similarity measures for those two words on the whole text, each word is given one score describing the similarity to the word with the highest tf-idf score („word_similarity_x“), and one score describing the similarity to the word with the second highest tf-idf score („word_similarity_y“). The figure below shows the words from the text plotted with the similarity to the word „support“ on the x axis and the similarity to the word „position“ on the y axis.

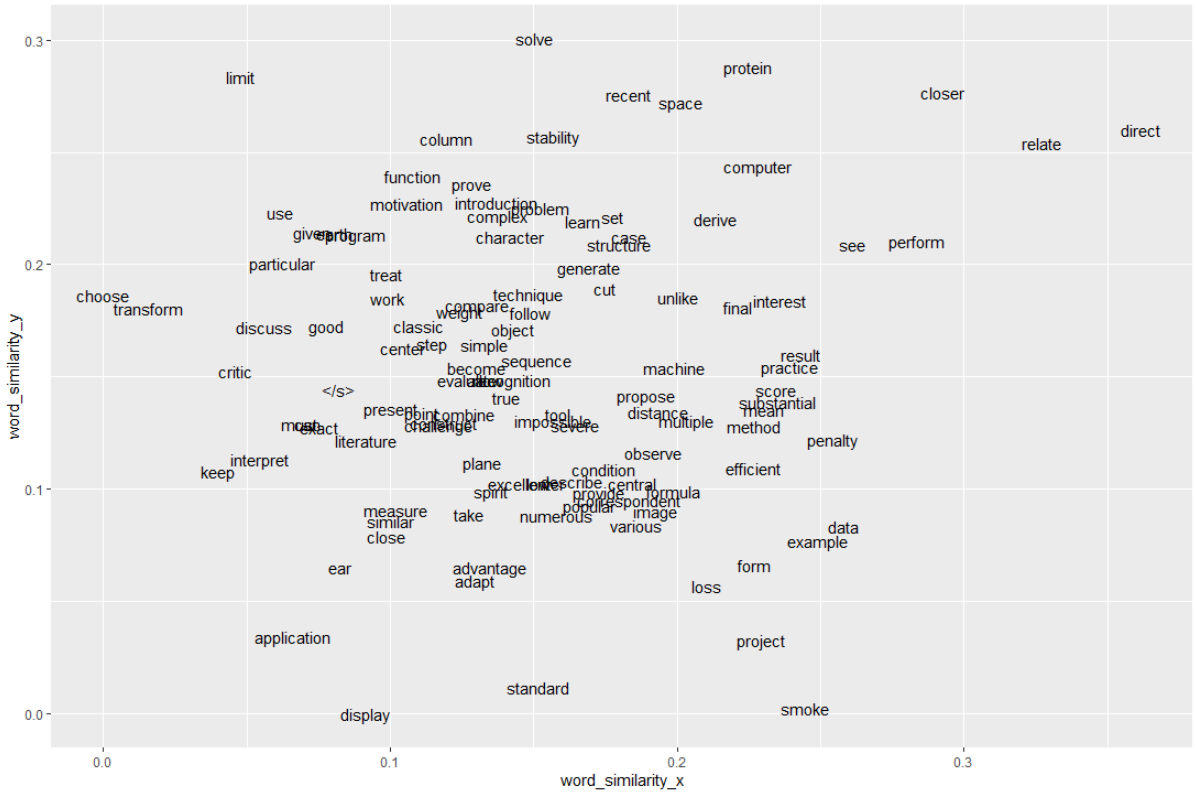

Figure 4 - Visualization of the words from "117.txt" after vectorization

There are some words in the plot, that show a high similarity for both „support“ and „position“, such as the word „direct“. „limit“ has a high correlation with the word „position“ but a weak one with „support“. Words like „application“ have a very low similarity to either of the two reference words.

In the next section, the vectorized words are clustered using the k-means clustering technique.

4.2 K-means Clustering

Using k-means, the vectorized words can now be clustered. The default cluster amount of 4 will be selected, resulting in the following plot:

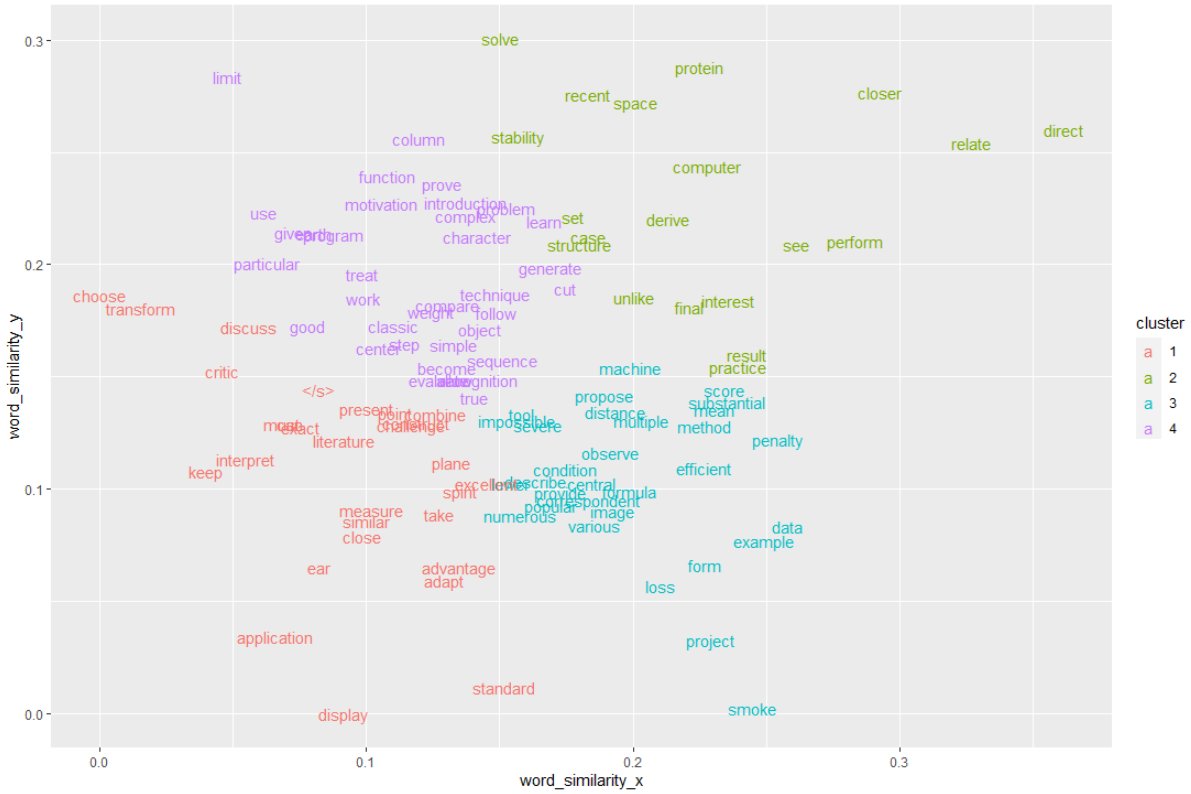

Figure 5 – Resulting word clusters after running the k-means algorithm

The words of interest are located in cluster 2. These words show a relatively high similarity to both reference words. Words in clusters 3 and 4 are showing similarities to one of the reference words, but a relatively low one to the other. Cluster 1 contains words with low similarities to either of the two reference words. These are most likely words commonly used in many texts and are therefore of lesser interest.

In the following section, the decision boundaries for the clustered words will be determined.

4.3 K-nearest-neighbor Boundaries

By creating the decision boundaries for the previously clustered words, it is possible to observe at which coordinate points words will fall under which cluster. This can be beneficial when creating the clusters and decision boundaries for the words from each text to observe under which cluster new words will fall.
Figure 6 shows the previously clustered words with the decision boundary for each cluster.

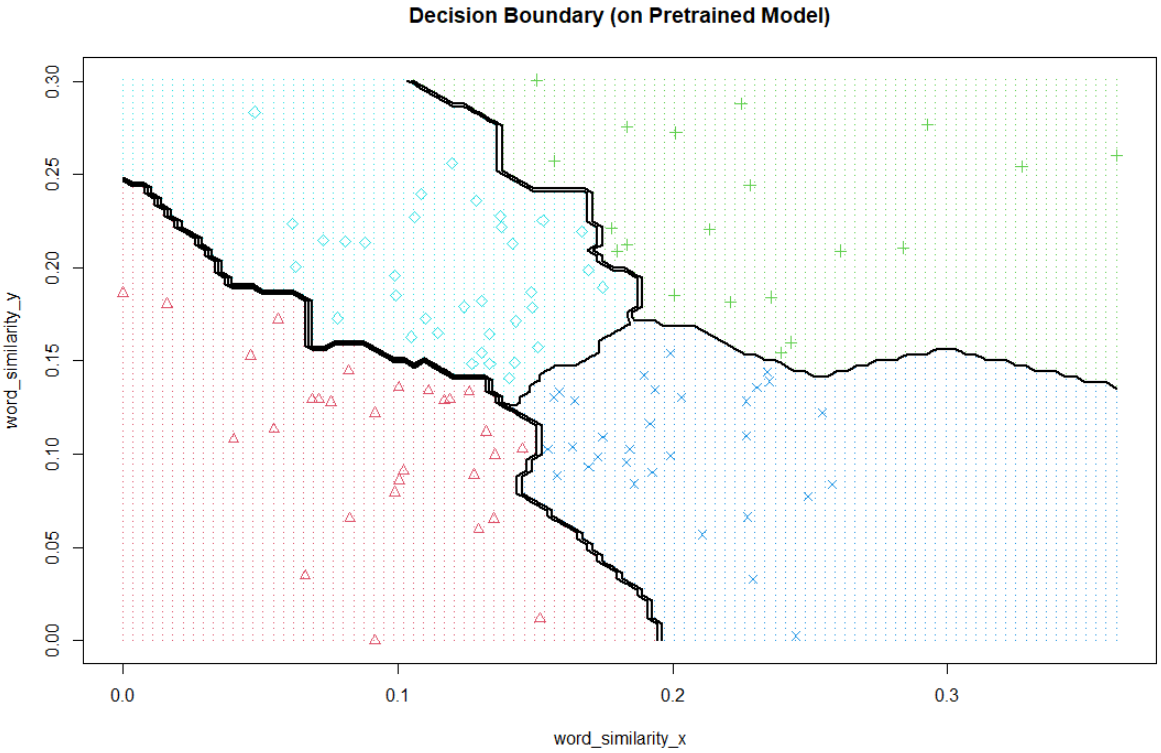

Figure 6 - Decision boundaries for previously clustered words

Finally, the „word_similarity_x“ and „word_similarity_y“ can be summed for each word to obtain a combined similarity score per word. The words with the highest combined similarity are of interest here. By ranking all words for each text, the 6 words with the highest combined similarity score can be extracted and presented with the 2 reference words in a table:

Word	Text 1	Text 10	Text 100	Text 101	Text 102	Text 103	Text 104	Text 105
1	train	environment	cloud	user	empty	economy	decision	definitely
2	solution	agent	local	person	online	cell	factor	position
3	size	computer	application	growth	view	application	classic	good
4	generate	example	require	run	usual	whose	error	focus
5	exact	ear	program	current	appeal	equal	space	language
6	course	provide	paper	effect	complete	make	see	application
7	code	probably	develop	target	approximately	condition	proceed	statistics
8	piece	class	specific	potential	practice	central	differ	active

Table 1 - Top 8 words with highest combined similarities for 8 of the 20 analyzed texts. Note that the first two words for each column are the reference words corresponding to the text.

This method provides an additional way of extracting the topics for each text. It is debatable if the tf-idf or the addition of the vectorization & similarity method performs better, since the prefered keywords for describing a certain text will differ from observer to observer.
It can be benificial to run both methods depending on the underlying corpus and personal preferences.

5 Conclusion and Next Steps

This project showed multiple ways of categorizing texts from a corpus into topics by extracting the most significant keywords from each text.
Further steps could involve creating a universal topic model with clusters and decision boundaries for a combination of words correlating to each topic. In theory, an observer could then determine the topic of a text by extracting its keywords and comparing the combination of keywords to the topic clusters.