Experimental study on Topic Modeling

Elliot Farmer Garcia (1396222)   Hari Krishna Pabbati (2159295)   Phuc Vo (1410830)

Abstract

This project will use topic modeling to analyze media coverage of terrorist organizations. The goal is to identify the key topics that are discussed in the media, and to understand how these topics are framed. The project will use a dataset of articles from the Wall Street Journal and the New York Times that were published in 2017. The data will be preprocessed and cleaned, and then topic models will be created using a variety of parameters. The results of the topic modeling will be analyzed and discussed in the context of the overview statement. The project will conclude with a discussion of the implications of the findings for journalists and other stakeholders.

1 Introduction

Topic modeling is a statistical method that identifies the hidden thematic structure in a collection of documents. It is a powerful tool for extracting insights from large amounts of text data. Topic modeling can be used for a variety of tasks, such as:

- Understanding the content of large corpora of text: Topic modeling can be used to identify the key topics that are discussed in a large corpus of text. This can be helpful for understanding the overall themes of a document collection, or for identifying the most important topics that are being discussed in a particular domain.
- Clustering documents: Topic modeling can be used to cluster documents that are similar in terms of their content. This can be helpful for organizing large document collections, or for identifying groups of documents that are related to a particular topic.
- Extracting keywords: Topic modeling can be used to extract keywords that are associated with each topic. This can be helpful for summarizing the content of a document collection, or for identifying the most important keywords that are being used in a particular domain.
- Improving the performance of other text mining tasks: Topic modeling can be used to improve the performance of other text mining tasks, such as document classification and information retrieval.

Overall, topic modeling is a powerful tool for extracting insights from large amounts of text data. It can be used for a variety of tasks, such as understanding the content of large corpora of text, clustering documents, extracting keywords, and improving the performance of other text mining tasks.

In this paper, our tasks is to help journalists with a general idea of how terrorist organizations are depicted in the current media outlets. It takes advantages of text mining techniques such as tokenizing, lemmetization, stemming, and topic modeling to support a journalist in grasping the idea of how articles on Wall Street Journal and New York Times are discussing about terrorism.

2 Methodology

Loading and Splitting Data:
Firstly, load the raw data into a text object, and then split text on feed file character. Build a corpus object containing all the Splitted articles.

Cleaning and Preprocessing:
Clean up the corpus and remove any meta-data within the articles (i.e.) converting the text into lower case and checking the copyright and removing them. Then, tokenize the text and remove punctuation marks. Perform stemming and lemmatization to reduce the dimensions of the data

and finally removing Stop words.

Feature Extraction:
Extract features from the preprocessed data. Use techniques like Term Frequency-Inverse Document Frequency (TF-IDF) to extract features that represent the most important and relevant words in the articles.

Exploratory Data Analysis:
Analyze the word frequency, create summaries of the extracted features, and use plots like Box plot, for exploration.

Topic Modeling:
Using the cleaned and preprocessed corpus, create a topic model. To identify the topics in the corpus, use techniques such as Latent Dirichlet Allocation (LDA). Rerun the topic modeling with different parameters several times and save the results in output files.

Interpret and Visualize:
Discuss the different results in the context described in the overview statement. Support your findings with relevant model outputs and visualizations. Visualize the topics and the distribution of words in the topics. Compare the different models and discuss the differences in the results.

3 Experimental Results

3.1 Preprocessing and exploratory data analysis

3.1.1 Building the Corpus

In order to build the corpus from the given folder of documents, we start by creating a list of strings where each string represents one article. These articles are read into the notebook from a directory named "Articles" using Path Library from pathlib. The directory contains text files with variable number of articles per file.

To read the articles, we opens each file and use 'readLines()' method to extract contents of the files as a list of lines. We then 'join()' the lines into a single string. Lastly, we separate the single string into separate article based on the feed file character `\f`. The resulting list is then appended to the 'corpus' list created at first. There are 1,622 articles in the corpus.

3.1.2 Cleaning the Corpus

Cleaning metadata from a corpus is an important step in natural language processing (NLP) tasks because it can help to improve the accuracy of text analysis by removing irrelevant information that can negatively impact the performance of algorithms.

Metadata can include information such as the author, date, and source of a document. While this information can be useful for certain tasks, such as determining the credibility of a source or analyzing the writing style of an author, it is not always relevant or necessary for other tasks, such as sentiment analysis or topic modeling.

In addition, metadata often contains special characters or formatting that can interfere with text analysis, such as line breaks or quotation marks. Removing this information can make the text easier to process and analyze, leading to more accurate results.

By removing metadata, the text is reduced to its essential components, making it easier to analyze and understand. This can ultimately lead to more accurate and meaningful insights from the data.

To remove the meta data, we skip the metadata at the beginning and the end of every document then slice the article after copyright and before the document ID (last line).

Next, we define 3 helpers as follows:

1. The "TOKENIZER" variable is defined using the "RegexpTokenizer" class from the NLTK library, which splits the text on any non-word character.

2. The "LEMMATIZER" variable is defined using the "WordNetLemmatizer" class, which is used to lemmatize verbs in the text.

3. The "STOPWORDS" variable is a set of stopwords defined using the "stopwords" function from the NLTK library. This set is augmented with the lemmas "say" and "mr", which are removed from the text because they appear frequently in articles from the New York Times.

By combing these helpers functions, we are able to preprocess the text and created a list of strings named "processed_corpus" with each string is an article or document. This ensure that our analysis is conducted on a clean, standardized dataset. This allows us to extract more accurately identified patterns and trends within the corpus.

3.1.3 Extracting features

For the purpose of obtaining the features from the list of "cleaned" articles, we uses the TfidfVectorizer from scikit-learn to extract features from the preprocessed corpus of documents. TfidfVectorizer is a text feature extraction method that computes the Term Frequency-Inverse Document Frequency (TF-IDF) metric for each term in the corpus. TF-IDF is a numerical statistic that reflects how important a word is to a document in a corpus. The TF-IDF value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Once the TF-IDF values are computed for each term in the corpus, the resulting matrix is transformed into a Pandas dataframe for easier handling and analysis. The resulting dataframe, df, has one row for each document in the corpus and one column for each unique term in the corpus. Each cell in the dataframe represents the TF-IDF value for the corresponding term in the corresponding document.

By using TF-IDF to extract features from the corpus, we can create a numerical representation of the documents that can be used for various machine learning tasks, such as clustering, classification, and information retrieval.

3.1.4 Creating summaries and plots on features as exploration

As a result, we were able to extract from 1622 documents 32752 features with summaries as follows.

The top 30 most frequent words in the corpus were identified using TF-IDF, and the results are shown in the above list. The most frequent words include "state", "trump", "islamic", "syria", "force", "attack", "american", "iraq", "military", and "president". Interestingly, the list also includes several location names, such as "syrian", "iraqi", "mosul", and "city", as well as names of various groups, such as "unite", "group", "people", "officials", "government", and "administration". Overall, these frequent words provide a good indication of the major themes and topics covered in the corpus, including politics, military actions, and conflict in various regions.

The top 50 most frequent words are visualized using a word cloud. It is a type of visualization that displays a collection of words in which the size of each word indicates its frequency or importance. In a typical word cloud, the most frequently occurring words are displayed with larger font sizes and are placed in the center, while less frequent words are shown with smaller font sizes and are positioned around the edges of the cloud.

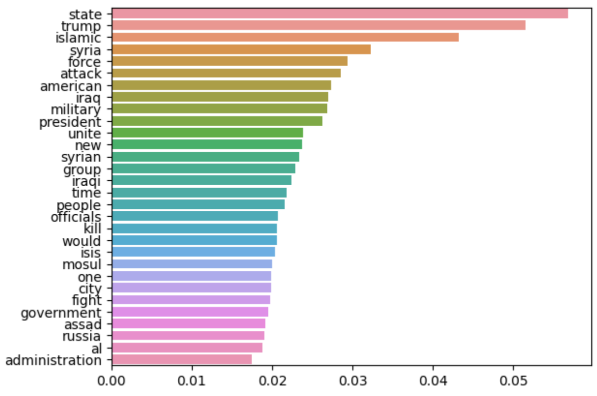

Figure 1: Bar plot of top 30 words. X-axis is the TF-IDF score, Y-axis is the word associated with the score.

"*ISIS*" is the only organization mentioned in the top 50 words. This indicates the significance of the group in the context of the document, which is likely related to the conflict in Syria and Iraq.

Assad" and *Syrian*" both appear in the top 20 words, which suggests a focus on the Syrian civil war and Bashar al-Assad's regime.

"*Russia*" appears in the top 30 words, indicating the country's involvement in the conflict in Syria and its relationship with the United States.

Trump" and *Obama*" both appear in the top 50 words, which suggests that the document covers a relatively recent time period (i.e., within the last decade). The presence of both names may also suggest a political angle to the document. Other

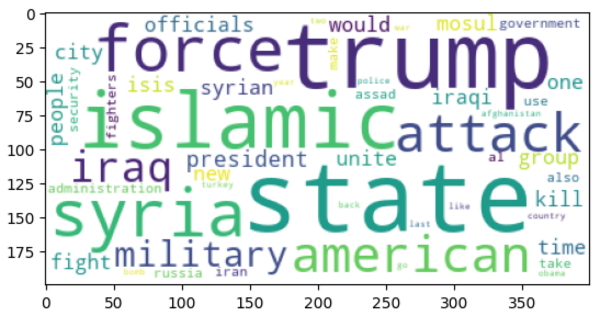

Figure 2: Word cloud of the top 50 words in the corpus

words related to conflict and security, such as *fight*", *attack*", *military*", and *security*", also appear in the top 50 words. This further supports the idea that the document is related to a conflict or security situation.

3.2 Topic Modeling

3.2.1 Creating a topic model

Prior to the creation of the topic model by Genism library, we transform the preprocessed documents into a format that can be fed into Genism.

First we split each preprocessed document into a list of individual words (or tokens), which are then stored as a separate list for each document in the variable texts. This is done using a list comprehension, which iterates over each document in the preprocessed corpus and applies the split() method to tokenize the text.

Next, a CorporaDictionary object is created from the list of tokenized documents. This object is used to create a dictionary of all the unique words (or tokens) in the corpus, along with a unique integer ID for each word. This dictionary is stored in the variable id2word.

Finally, a bag-of-words representation is created for each document in the corpus using the doc2bow() method of the id2word object. This method takes a list of tokens (i.e., the tokenized text of a single document) as input and returns a list of (word ID, word count) pairs, where each pair represents a word in the document and the number of times it appears. These pairs are stored as a separate list for each document in the variable bows.

We then create a formatter to takes the output of LDA show_topics() and formats it into a dictionary with the following structure:

```
{
    topic_num_1: {
        word_1: probability_1,
        word_2: probability_2,
        ...
    },
    topic_num_2: {
        word_1: probability_1,
        word_2: probability_2,
        ...
    },
    ...
}
```

The function does this by iterating over each topic in the output and converting the string representation of the topic, which is a series of words and their probabilities, into a dictionary. Specifically, the function splits each word-probability pair using the split() method and constructs a dictionary that maps each word to its corresponding probability. The function then maps each topic number to its corresponding dictionary of words and probabilities using a dictionary comprehension.

3.2.2 Running Latent Dirichlet Allocation Model (LDA model

We first perform a grid search for the best hyperparameters for an LDA topic modeling model. We then initialize an empty list model_results to store the results of each model, and create a directory called lda_scores to save the results of each model run.

Then we loop through a range of n_topics values from 2 to 10 and iterates through all possible combinations of alpha and beta values (0.01, 0.3, and 1). For each combination of hyperparameters, we create an LDA model using the LdaModel function from the gensim package. It then calculates the model's perplexity and coherence scores using the log_perplexity and get_coherence functions from the LdaModel and CoherenceModel classes, respectively.

Finally, the results of each LDA model run are saved into a JSON file in the lda_scores directory, named according to the hyperparameters used for that model run. It also saves the full grid search results to a single JSON file in the same directory. The format_topics function is used to format the topics for each LDA model run in a JSON-friendly way.

3.2.3 Minimizing perplexity

The "n_topics" parameter specifies the number of topics to discover, which in this case is set to 10. The "alpha" parameter controls the sparsity of the document-topic distribution, while the "eta" parameter controls the sparsity of the topic-word distribution.

The "perplexity" metric is a measure of how well the model is able to predict unseen data, with lower values indicating better performance. In this case, the perplexity value is negative, which is unusual, but indicates that the model is doing well at predicting the data.

The "coherence" metric is a measure of how semantically coherent the topics are, with higher values indicating better coherence. In this case, the coherence value is 0.415, which is relatively low but not unexpected given the nature of the corpus.

Looking at the individual topics, it appears that topic 0 is related to Islamic militant groups, while topic 1 is related to the conflict in Syria. Topic 2 appears to be more generic, possibly related to social media or news, while topic 3 is related to terrorist attacks. Topic 4 is about Trump and politics, while topic 5 is more general and seems to be about people's opinions. Topic 6 is again related to politics, with a focus on the Trump administration. Topic 7 is related to national security, with a focus on refugees and Muslim bans. Topic 8 is about crime, with a focus on police and shootings, and topic 9 is again more general, possibly related to events happening over the course of a year.

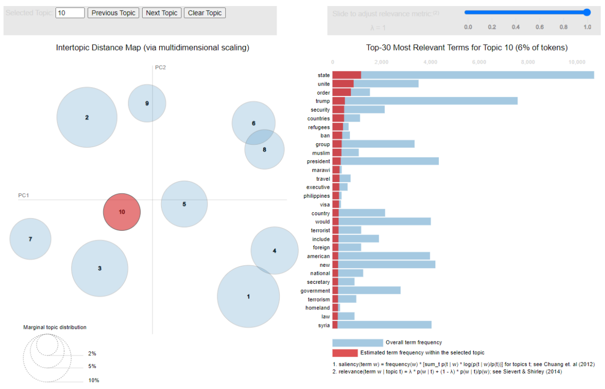

Figure 3: Topic number 10 was chosen with list of relevant words displayed in a bar graph on the right

Overall, these topics seem to capture some of the major themes and events that were happening in the corpus of text data. However, it's important to keep in mind that topic modeling is an unsupervised technique, meaning that the topics are discovered based solely on the distribution of words in the text data, and not on any external knowledge or labels. Therefore, the quality of the topics is highly dependent on the quality of the input data and the parameters used in the model.

3.2.4 Maximizing Coherence

This is a result from an LDA model that has been optimized for coherence. Coherence measures how semantically coherent the topics are, based on how often words co-occur within a sliding window of a fixed size across the documents in the corpus. The higher the coherence score, the better the model's ability to capture meaningful topics.

The model has 5 topics, with different probabilities for each word within each topic. The top words for each topic are as follows:

Topic 0: state, islamic, syria, force, u, military, group, iraq, american, syrian Topic 1: shall, state, order, unite, entry, section, secretary, nationals, effective, visa Topic 2: time, new, one, like, people, make, would, go, york, take Topic 3: go, think, attack, people, gigot, get, trump, one, state, police Topic 4: trump, state, president, new, unite, would, american, time, administration, one

Based on the top words, it seems that the topics are related to the following themes:

Topic 0: conflict and military intervention in the Middle East Topic 1: legal and administrative procedures Topic 2: general topics related to time and people Topic 3: terrorism and law enforcement Topic 4: American politics, specifically the Trump administration.

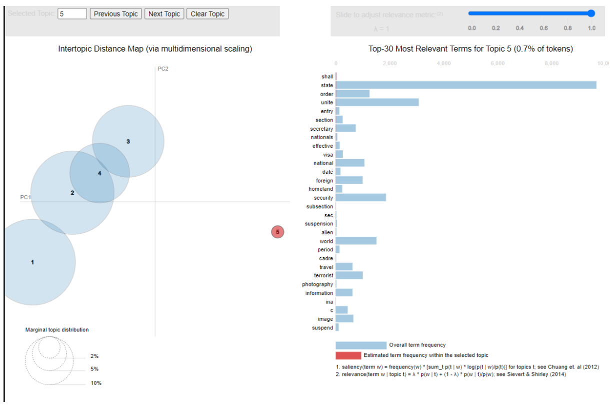

Figure 4: Topic number 5 was chosen with list of relevant words displayed in a bar graph on the right

However, it is important to note that topic modeling is an unsupervised technique, and it is up to the analyst to interpret the topics and assign meaning to them. It is also possible that some topics are more coherent than others, and some may be redundant or not very informative. Therefore, it is recommended to evaluate the coherence of the topics in context and to compare them with other models to make a more informed decision.

4 Link to Git Repository

Code repository can be found at this link.

5 Conclusion

In conclusion, this assignment provided hands-on experience on text preprocessing, tokenization and stop word removal. By analyzing the corpus of text files, we were able to identify the most common words. we have performed LDA topic modelling after carefully analyzing the data. This model provides valuable insights into the variety of topics that can be found in a large corpus of text and the relative importance of certain words within each topic.

6 Author Credit Statement

Elliot Farmer Garcia:
`Software, Python code, Supervision`
Hari Krishna Pabbati:
`Formal Analysis, Methodology, Writing-review  editing`
Phuc Vo:
`Conceptualization, experimentation and results, Writing - original draft`

7 References

- https://medium.com/analytics-vidhya/topic-modelling-using-lda-aa11ec9bec13

- https://github.com/kavgan/nlp practice/blob/master/tf-idf/Keyword/Extraction/with/20TF-IDF/and/SKlearn

- https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/