Text Mining Project

Susanna Maugeri (839365), Veronica Grazia Morelli (839257)

June 2023

Abstract

This project focuses on the application of Natural Language Processing (NLP) techniques for the tasks of topic modeling and text summarization using the CNN and Daily Mail dataset. The task of topic modeling involves identifying the main subject or theme of a text document, while text summarization aims to condense a document while preserving its essential information. To accomplish task topic modeling, we employed advanced NLP techniques such as topic modeling algorithms, we extracted key themes and subjects from the dataset. Through meticulous analysis and evaluation, we gained insights into the topic distribution and discovered meaningful patterns and trends within the documents. For text summarization, we employed a separate methodology that utilized the Daily Mail dataset as a training and evaluation resource. Our approach involved extracting essential information from the dataset and generating concise summaries while maintaining the coherence and integrity of the original text. Through rigorous experimentation and evaluation, we assessed the quality and effectiveness of the generated summaries. The results of this project highlight the potential of NLP techniques for task topic modeling and text summarization.

1 Dataset

The CNN/Daily Mail dataset [1] is a widely used benchmark dataset in the field of natural language processing (NLP). It consists of news articles collected from the websites of CNN and the Daily Mail, paired with bullet-point summaries. The dataset is used for text summarization tasks because it provides a large collection of articles with associated summaries. The articles are typically longer pieces of text, while the summaries offer concise representations of the main points in the articles. Additionally, the CNN/Daily Mail dataset is also used for topic modeling tasks. By analyzing the articles in the dataset, researchers can discover underlying topics and patterns, allowing for a better understanding of the main themes and subjects covered in the news domain.

The dataset is large and diverse, it contains a substantial number of articles, covering various topics and providing a diverse range of textual content. This diversity allows for a comprehensive exploration of different subjects and themes. The articles in the dataset are sourced from reputable news organizations, ensuring that the content is relevant and reflects current events. This real-world relevance is valuable for both summarization and topic modeling tasks. The dataset provides reference summaries that can be used for evaluating and comparing the performance of different text summarization and topic modeling techniques. This enables researchers to benchmark their models against existing approaches. Overall, the CNN/Daily Mail dataset serves as a valuable resource for developing and evaluating text summarization and topic modeling algorithms, enabling researchers to advance the state-of-the-art in these NLP tasks.

The dataset used can be found at this link [1].

Before implementing topic modeling and text summarization techniques, some data cleaning steps are applied.

Since the two tasks applied to this dataset, Topic Modeling and Text Summarization, have different need in terms of preprocessing, it was not performed at the beginning of the work, but singularly at the beginning of each task.

[1] https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail

2 Topic Modeling

Topic modeling is an unsupervised learning technique in Natural Language Processing (NLP) that identifies hidden themes or topics in a collection of documents. It doesn't require prior knowledge or manual annotation.

By assigning probabilities to word occurrences for different topics, it organizes and extracts meaningful information from unstructured text. The algorithm analyzes the data, discovers patterns, and uncovers topics based on statistical patterns and probabilistic models.

Since the topics are not explicitly labeled or defined beforehand, they are considered *hidden* or *latent*.

However, it's important to note that Topic Modeling has its limitations. The interpretability of the topics depends on the quality of the data, the chosen algorithm, and the expertise of analysts. Additionally, the preprocessing steps, parameter selection, and evaluation metrics are crucial in obtaining meaningful results.

In summary, topic modeling is a valuable technique for exploring and understanding large textual datasets, contributing to knowledge discovery, decision-making, and information retrieval.

2.1 Preprocessing

In Topic Modeling, it is important to preprocess texts before applying algorithms. This involves cleaning and preparing the texts for analysis. The following steps were performed as preprocessing on the dataset:

- Punctuation removal: Punctuation marks like commas, periods, and parentheses were removed as they are not relevant for topic analysis.
- Stopwords removal: Common words like *the*, *is* and *a* were removed as they don't contribute significantly to defining topics. Removing them reduces noise and allows algorithms to focus on more meaningful words.
- Tokenization: The text was broken down into smaller units called *tokens*, which in this case are single words. This step treats each word as a separate unit for topic analysis.
- Lemmatization: Words were reduced to their base or dictionary form (*lemma*). For example, *playing*, *played* and *plays* were all transformed to *play*. This helps consolidate different word forms into a single representation, reducing data dimensionality and improving topic coherence.
- Lowercasing: All the texts were transformed to lowercase. This promotes uniformity and makes word matching easier by treating different cases of the same word as identical.
- NA removal: empty documents were removed.

After preprocessing, the dataset consists of 311,952 documents.

2.2 LSA - Latent Semantic Analysis

LSA (Latent Semantic Analysis) or LSI (Latent Semantic Indexing) is a mathematical technique used in natural language processing (NLP) to analyze the relationships between words and documents in a text corpus.

The process of LSA involves the following steps after preprocessing:

- Building the term-document matrix: A matrix is constructed where each row represents a unique word, each column represents a document, and the cell values represent the frequency or importance of the word in the document. This matrix captures word occurrence patterns across the documents.
- Dimensionality reduction: Techniques like Principal Components Analysis (PCA) or Singular Value Decomposition (SVD) are used to reduce the dimensionality of the term-document matrix. PCA finds new dimensions that capture the maximum amount of variation in the data, while SVD decomposes the matrix into three separate matrices: U, Σ, and V^T. These matrices represent the relationships between words, the importance of different latent topics, and the relationships between documents, respectively.
- Topic extraction: LSA examines the decomposed matrices to extract underlying topics in the data. Topics are represented as a combination of words, with words carrying higher weights contributing more to a specific topic.

However, LSA has limitations. It doesn't consider word order or context, struggles with polysemous words (words with multiple meanings), and may not capture fine-grained or domain-specific distinctions. Alternative techniques like Word Embeddings and Neural Network-based models have emerged to address these limitations.

2.3 LDA - Latent Dirichlet Allocation

LDA (Latent Dirichlet Allocation) is a statistical model that assumes documents are generated from a mixture of latent topics, with each topic characterized by a distribution of words.

LDA identifies topics by iteratively estimating the probabilities of topics and words in documents to maximize the likelihood of the observed text data.

The process of LSA typically involves the following steps, after the phase of preprocessing:

- Building the term-document matrix: A matrix is created where each row represents a document and each column represents a unique word in the corpus. The cells of the matrix can represent word frequencies, TF-IDF scores, or other suitable representations of word importance in the document.

- Model training: the LDA model is initialized with the desired number of topics. It randomly assigns topics to words in each document and iterates over the documents and words, updating topic assignments based on probabilities calculated from the model's current state. This process continues until convergence, where topics stabilize and significant changes cease.

- Topic inference and interpretation: once the model is trained, it can be used to infer the topic distribution for a new, unseen document, estimating topic proportions based on the learned distributions. Topics can also be interpreted by examining the most probable words associated with each topic.

However, LDA assumes a bag-of-words representation, disregarding word order and context within a document. Additionally, the user needs to specify the number of topics in advance, which can be challenging if the optimal number of topics is unknown.

2.3.1 LDA with Scikit-learn and Gensim

Gensim and Scikit-learn (sklearn) are popular Python libraries that implement Topic Modeling algorithms. Both libraries provide the capability to perform LDA (Latent Dirichlet Allocation).

2.4 BERTopic

BERT (Bidirectional Encoder Representations from Transformers) [2] is a pre-trained language model developed by Google that uses Transformer Neural Networks to understand contextual relationships between words. It captures rich language representations and has been trained on a large corpus of text data.

BERTopic [3] is an open-source library for topic analysis based on the BERT algorithm. It automatically identifies and classifies topics in large text collections. BERTopic uses BERT to extract vector representations of documents and applies the clustering technique *HDBSCAN* (Hierarchical Density-Based Spatial Clustering of Applications with Noise) to group similar documents into topics.

The goal of BERTopic is to simplify topic analysis and enable efficient extraction of relevant information from texts. The library also offers features for visualizing and interpreting the identified topics.

2.5 Top2Vec

Top2Vec [4] is a machine learning algorithm that combines the benefits of topic modeling techniques with the efficiency of neural network-based approaches.

Its key feature are:

- Hierarchical Document Clustering: Top2Vec can discover broad topics and sub-topics within a document collection, automatically determining the optimal number of topics.

- Topic Embeddings: It represents topics as dense vectors called topic embeddings, capturing semantic relationships for more nuanced analysis.

- Document and Topic Similarity: It calculates document and topic similarities based on these embeddings, allowing for finding similar documents and identifying related topics.

- Topic Labels: It generates labels for discovered topics, providing a brief summary of the main theme or concept represented by each topic, aiding topic interpretation.

- Efficient Processing: Top2Vec employs an approximate nearest neighbor algorithm for fast computation of document and topic similarities, making it suitable for large-scale document collections.

2.6 Experimental Setup

All experiments are run on Google Colaboratory. For performance reasons, it has been divided into three parts: one script containing the preprocessing of the texts, one containing LSA, LDA with both Sklearn and Gensim and BERTopic, and the last one containing Top2Vec.

For models requiring the number of topics to be specified, 50 topics were chosen after a few attempts.

LSA was performed using Gensim's *LsiModel* function, which implements fast truncated SVD. The topics and associated probabilities were extracted using the *get_topics()* method.

LDA with Sklearn was implemented using the *LatentDirichletAllocation* function, which also requires the number of topics to be specified. The resulting topics can be visualized using the *prepare()* function of the library *pyLDAvis*.

Instead, Gensim's LDA was implemented with the function *Ldamodel*. As the two models above, it needs the user to indicate the number of topics before running.

Top2Vec is the easiest method to implement as it only requires as input the list of documents. It autonomously detects the appropriate number of topics.

Each method was tested with different sample sizes: 100, 1,000, 10,000. The initial idea was to test the methods also on a sample of 100,000 and on the complete dataset of 311,952 documents, but for computational reasons this was not possible. The free plan of Google Colaboratory permits the use of a 12.5Gb or RAM, which were not enough for the biggest models.

[1]Implemented with Sklearn

[2]Implemented with Gensim

[3]Top2Vec requires a higher length of the corpus as a minimum

2.7 Evaluation

Topic coherence is a metric used to evaluate the interpretability and semantic similarity of topics generated by topic modeling algorithms. It measures the pairwise similarity between words within a topic based on their co-occurrence patterns in the corpus. Words that frequently appear together in the same context are considered to be more coherent.

Higher topic coherence scores indicate that the words within a topic are closely related and provide a clear interpretation, while lower scores suggest that the words are more diverse and less coherent.

2.7.1 Comparability of evaluation metrics between models

The evaluation of the presented models faced some challenges due to their diverse nature. BERTopic and Top2Vec currently do not provide built-in methods for evaluating the detected topics, although the authors have mentioned the intention to include evaluation capabilities in the future.

Therefore, the evaluation in this project mainly relied on the *CoherenceModel* function of *Gensim*. Adaptations were made to the outputs of BERTopic and Top2Vec to fit this evaluation method, but not all models could be evaluated in the same way. Sklearn's implementation of LDA, for example, encountered the error *ValueError: This topic model is not currently supported. Supported topic models should implement the 'get_topics' method.*. Accordingly, it was evaluated by the *metric_coherence_gensim* function (with option *measure='c_v'*) of the *tmtoolkit* library.

Methods	**100**	**1000**	**10k**	**100k**
LSA	0.28	0.31	0.30	0.45
LDA [1]	0.75	0.75		
LDA [2]	0.23	0.28	0.42	0.59
BERTopic	0.44	0.76	0.83	
Top2Vec	-[3]	0.65	0.85	

Topic Modeling Evaluation: global coherence of the topics by method and sample dimension.

Unfortunately, as mentioned before, it was not feasible to evaluate all the methods on the sample of size 100,000 and the complete dataset.[4]. Sklearn's LDA failed on the samples of size 10k and 100k, while BERTopic and Top2Vec encountered difficulties only with the sample of size 100,000.

Anyway, we expect the goodness of the model to keep increasing when fitted on more and more documents.

Furthermore, it should be noted that Top2Vec cannot be trained on a very small document sample and raises an error in such cases. This limitation was encountered when using a sample size of 100 documents.

2.7.2 Top2Vec score

Cosine similarity is a commonly used measure to calculate the similarity between two vectors in a vector space. It is widely used in various fields, including information retrieval, natural language processing, and document search.

When calculating cosine similarity between two vectors, the resulting value ranges from -1 to 1. A value of 1 indicates perfect similarity, while a value of -1 indicates complete dissimilarity. A value of 0 indicates that the two vectors are orthogonal.

Calculating the cosine similarity for each pair of objects and then averaging them can provide a general measure of the average similarity or dissimilarity within a set of vectors. The function *get_topics* of Top2Vec returns, together with the words in each topic and the number of the topic, the cosine similarity scores of the top 50 words to the topic.

Out of curiosity, and also to complement the coherence measure, we computed the mean of the cosine similarity between the first 20 words in each topic and the topics themself.

100	**1000**	**10k**	**100k**	**300k**[1]
-	0.46	0.56	-	-

Similarity score between words and their topic by sample dimension.

[4]Complete dataset: 311952 documents

[1]Complete dataset

2.8 Results

Some examples of topics for each type of model developed, as well as examples of visualization possibilities of BERTopic and Top2Vec are shown in the Appendix A.

3 Text Summarization

Text Summarization is the ability to write a shorter, condensed version of a paragraph, an article, or a book while retaining most of the original text's meaning. Automatic Text Summarization means automating that task without human intervention using algorithms, linguistic theorems, or artificial intelligence. There are two types of Automatic Text Summarization: Extractive Summarization and Abstractive Summarization. Extractive summarization selects a subset of sentences from the text to form the summary. Sentences are scored by their relevance to the text's overall meaning. On the contrary, abstractive summarization generate summaries as a human would and it writes the summary from scratch. The latter method makes for richer and more concise summaries, while the former tends to create more heterogeneous and usually longer summaries. Abstractive approach is definitely more appealing, but much more difficult than extractive one. Following what was presented in class, we decided to implement extractive methods, such as: Luhn summarizer, Latent Semantic Analysis summarizer, Text Rank summarizer and Kullback-Lieber summarizer. These unsupervised methods are quietly easy to implement and do not require machine learning. Considering that CNN, daily mail dataset is built for abstractive summarization we decided to apply deep learning techniques, the current state-of-the-art for abstractive text summarization.

The results of Extractive and Abstractive summarization can be found in the Appendix B, at the end of the report.

3.1 Exploratory Analysis

The dataset is divided into training, validation and test. Training dataset is composed by 287113 samples, validation by 13368 samples and test set by 11490 samples. In each dataset there are no missing data.

Before proceeding with the application of the summarisation methods, a quick exploratory analysis is performed on the training dataset.

3.2 Extractive Text Summarization

Extractive summarization creates an intermediate representation of the text to find the important content. There are two types of representation: topic representation and indicator representation:

- Topic Representation

  The topic-words technique is used to identify words that represent the main topic of a document. H.P. Luhn's approach used frequency thresholds to find these representative words. Latent Semantic Analysis (LSA) is an unsupervised method that extracts a text representation based on observed words. It considers the weights of words to determine their correlation to the topic. Bayesian topic models, such as HIERSUM using Latent Dirichlet Allocation (LDA), treat documents as mixtures of topics, with each topic represented by a probability distribution over words.

- Indicator Representation

  These approaches represents sentences using features like length or position within the document. Graph methods, such as TextRank, construct a graph where sentences are vertices and edges represent sentence similarity. Machine learning approaches, like Naive Bayes, classify sentences as summary or non-summary based on their features, using a training set of documents and summaries. These techniques provide different ways to extract and represent the main topic or summary of a document.

3.2.1 Luhn Summarizer

Luhn's summarizer was one of the first attempts in the field of text summarization. In the paper "The Automatic Creation of Literature Abstracts" [5]. Luhn et al. propose that important sentences for summarization contain significant words that are not considered stop-words. A list of language-specific stop-words is necessary for this algorithm. In extractive summarization methods like Luhn's, sentences with the highest scores based on significant words are included in the summary. To perform summarization with Luhn summarizer we used Sumy library [6].

3.2.2 LSA (Latent Semantic Analysis) Summarizer

Latent Semantic Analysis (LSA) is a technique used in natural language processing to analyze the semantic similarity between documents and words. It uses Singular Value Decomposition (SVD) to transform data into a lower-dimensional space, capturing the underlying semantic relationships. LSA summarization involves constructing a corpus, preprocessing the text, creating a document-term matrix, applying SVD to reduce dimensionality, and calculating similarity between documents. The most similar documents or sentences are then selected to generate a summary, condensing the information while retaining important details. LSA has limitations and may not always capture the nuances and context-specific details present in the original text. To perform summarization with LSA summarizer we used Sumy library [6].

3.2.3 KL (Kullback-Lieber) Summarizer

The KL (Kullback-Leibler) summarizer is a text summarization technique that uses KL divergence to compare a summary with the original document. KL divergence measures the difference between two probability distributions. The summarization process involves representing the document numerically, constructing candidate summaries, calculating KL divergence between each summary and the original document, and selecting the summary with the lowest divergence. However, the effectiveness of KL summarization can vary based on implementation and document characteristics. To perform summarization with KL summarizer we used Sumy library [6].

3.2.4 TextRank Summarizer

TextRank summarizer is a graph-based technique inspired by PageRank for text summarization. It involves segmenting the document into sentences and representing them as nodes in a graph. The edges between nodes capture semantic similarity. A graph is constructed, and sentences are ranked iteratively based on their importance, considering both their own quality and connections in the graph. The top-ranked sentences are extracted to form the summary. TextRank leverages the sentence relationships to identify important and relevant information. To perform summarization with text rank summarizer we implemented it from scratch.

3.2.5 Experimental Setup and Evaluation

All experiments are run on Google Colaboratory platform where you can easily use a GPU. The GPU in question is a Tesla T4 with a memory usage of 15360MiB.

The dataset is a Pandas dataframe organized in two columns: *Source text* and *Target text*.

To perform summarization with *Sumy* library firstly I need to create a *Document* for each sentences. We use the *PlaintextParser*. Later, we need to apply some preprocessing steps, it is suggested to apply Stemming, with a stemmer provided by the same library, and stopwords removal. The used summarizers are: LuhnSummarizer, LsaSummarizer, KLSummarizer. An example function for lsa summarisation can be found in the snippet below.

```
def lsa_summarizer(article: str,
    sentence_count: int, lst_stopwords:
    list) -> str:

    parser = PlaintextParser.from_string(
        article, Tokenizer('english'))
    summarizerLsa = LsaSummarizer(Stemmer
        ('english'))
    summarizerLsa.stop_words =
        lst_stopwords
    lsa_summary = summarizerLsa(parser.
        document, sentences_count=
        sentence_count)

    return ' '.join([str(sentence) for
        sentence in lsa_summary])
```

Listing 1: LSA Summarizer function example

To implement the Text Rank summarizer, the following steps were followed:

1. The text was split into individual sentences using the sent_tokenize() function from the NLTK library.

2. Preprocessing steps were applied to clean the text, including converting to lowercase, removing punctuation, numbers, special characters, and stopwords. GloVe word embeddings were used to obtain vector representations for each sentence. Specifically, pre-trained Wikipedia 2014 + Gigaword 5 GloVe vectors were utilized.

3. Similarity scores between sentences were calculated, and the scores were stored in a matrix. Cosine similarity was employed to measure the similarity between sentence pairs.

4. The matrix was converted into a graph, where sentences served as vertices and similarity scores as edges. The PageRank algorithm was applied using the NetworkX library to rank the sentences.

5. A specific number of top-ranked sentences were selected to compose the final summary.

3.2.6 Evaluation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of evaluation metrics used to assess the quality of automatic text summarization systems. It measures the overlap between system-generated summaries and reference summaries created by humans. ROUGE scores include variants such as ROUGE-N (n-gram overlap), ROUGE-L (longest common subsequence), ROUGE-S (skip-bigram co-occurrences), and ROUGE-SU (combination of skip-bigrams and unigrams). These metrics calculate precision, recall, and F1-score to quantify the similarity between system and reference summaries. Higher ROUGE scores indicate better summarization performance, and they are widely used to evaluate the effectiveness of summarization systems.

3.3 Abstractive Text Summarization

Abstractive summarization is an NLP task that involves generating concise and coherent summaries by understanding the input text and creating new sentences. Unlike extractive summarization, which selects and combines existing sentences, abstractive summarization allows for more flexibility in expressing the main ideas. It employs techniques such as semantic analysis, language modeling, and text coherence modeling. Abstractive summarization is useful for condensing complex or lengthy texts into shorter, readable summaries. It finds applications in news, documents, and research papers where capturing key information is important.

3.3.1 Pegasus

Pegasus [7] is a cutting-edge model for abstractive text summarization developed by Google Research. It uses the transformer architecture and is trained on a large dataset of news articles and their human-written summaries. Pegasus generates concise and coherent summaries by leveraging an abstractive approach, allowing it to create new phrases and reorganize information. The model captures global context and dependencies using the attention mechanism in transformers. Pegasus undergoes pre-training and fine-tuning stages to optimize its summarization performance. It outperforms previous models in terms of summary quality and fluency, making it a valuable tool for generating accurate and coherent summaries across various domains.

3.3.2 T5

T5 (Text-to-Text Transfer Transformer) [8] is a transformer-based model developed by Google Research for text summarization and other NLP tasks. It follows an encoder-decoder architecture and can be fine-tuned on specific summarization datasets. T5 is trained in a "text-to-text" manner, allowing it to handle different tasks by providing specific input-output format pairs. T5 comes in different sizes: t5-small, t5-base, t5-large, t5-3b, t5-11b. For summarization, T5 takes a "summarize:" prompt followed by the source text and generates a summary. It has achieved state-of-the-art performance on various benchmark datasets and is highly versatile and widely used in text summarization applications.

3.3.3 Experimental Setup

All experiments are run on Google Colaboratory platform where you can easily use a GPU. The GPU in question is a Tesla T4 with a memory usage of 15360MiB. The dataset is a Pandas dataframe organized in two columns: *Source text* and *Target text*. To make training faster, the dataset was sampled: 10000 samples in the train dataset, 1000 samples in the validation dataset.

We fine-tuned the following models from HuggingFace ☺ Transformers library: Pegasus and T5 small. *AutoTokenizer* and *AutoModelForSeq2SeqLM* are used. Before fine-tuning the model is necessary to tokenize the text. This step is necessary because the text must be pre-processed in the same way as the text on which the original model was trained. The following pre-processing is applied to the original dataset:

```python
prefix = "summarize: "

def preprocess_function(examples):

    inputs = [prefix + doc for doc in
        examples["source_text"]]
    model_inputs = tokenizer(inputs,
        max_length=512, truncation=True)
    labels = tokenizer(text_target=
        examples["target_text"],
        max_length=64, truncation=True)
    model_inputs["labels"] = labels["
        input_ids"]

    return model_inputs
```

Listing 2: Preprocess function example

Subsequently, the text is tokenized. The metric computed for evaluation is the ROUGE score imported directly from HuggingFace's *evaluate* library. *Seq2SeqTrainingArguments* and *Seq2SeqTrainer* are used to define the training. The *DataCollatorForSeq2Seq* is also used. Models are trained for 3 epochs with 2e-5 learning rate, 0.01 weight decay and 4 batch size, the results are saved after each epoch.

3.3.4 Results

Metric	**Pegasus***	**T5 small**
Training Loss	1.271	2.067
Validation Loss	1.634	1.791
Rouge1	0.443	0.278
Rouge2	0.218	0.136
Rougel	0.320	0.233
Rouge avg	0.327	0.2157
Gen Len	70.748	19.000

*Trained on complete CNN/Daily mail dataset.

The results with Pegasus are considerably higher than with the T5 small. The reason for this result is that the Pegasus model was trained on the CNN/Daily mail dataset, which includes exactly the group of samples we used for training. In contrast, T5 was trained on a different dataset.

References

[1] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*, 2015.

[2] Maarten Grootendorst. Topic Modeling with BERT. https://towardsdatascience.com/topic-modeling-with-bert-779f7db187e6. Accessed: 2023-05-28.

[3] Maarten Grootendorst. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[4] Dimo Angelov. Top2vec: Distributed representations of topics. 2020.

[5] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[6] Miso Belica. Sumy, 2021.

[7] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

3.4 Appendix A

3.4.1 LSA

Examples of topics extracted:

- $' - 0.567 * "apple" + -0.236 * "watch" + 0.179 * "per" + 0.166 * "cent" + -0.132 * "software" + -0.129 * "user" + -0.127 * "ipod" + 0.123 * "people" + 0.120 * "child" + -0.118 * "music"'$
- $'-0.377 * "hospital" + -0.266 * "nurse" + -0.259 * "ebola" + -0.221 * "duncan" + -0.166 * "health" + -0.156 * "patient" + 0.145 * "woman" + -0.135 * "per" + -0.124 * "cent" + -0.117 * "texas"'$
- $'0.298 * "woman" + 0.220 * "people" + -0.200 * "obama" + -0.184 * "president" + 0.158 * "minister" + -0.157 * "child" + -0.151 * "say" + -0.150 * "state" + 0.145 * "school" + -0.138 * "per"'$
- $' - 0.249 * "family" + 0.219 * "car" + 0.218 * "time" + 0.209 * "royal" + 0.190 * "people" + 0.184 * "prince" + -0.174 * "million" + -0.160 * "show" + 0.150 * "say" + 0.116 * "found"'$

3.4.2 BERTopic

The following are the top 5 words for the first 12 topics extracted by BERTopic.

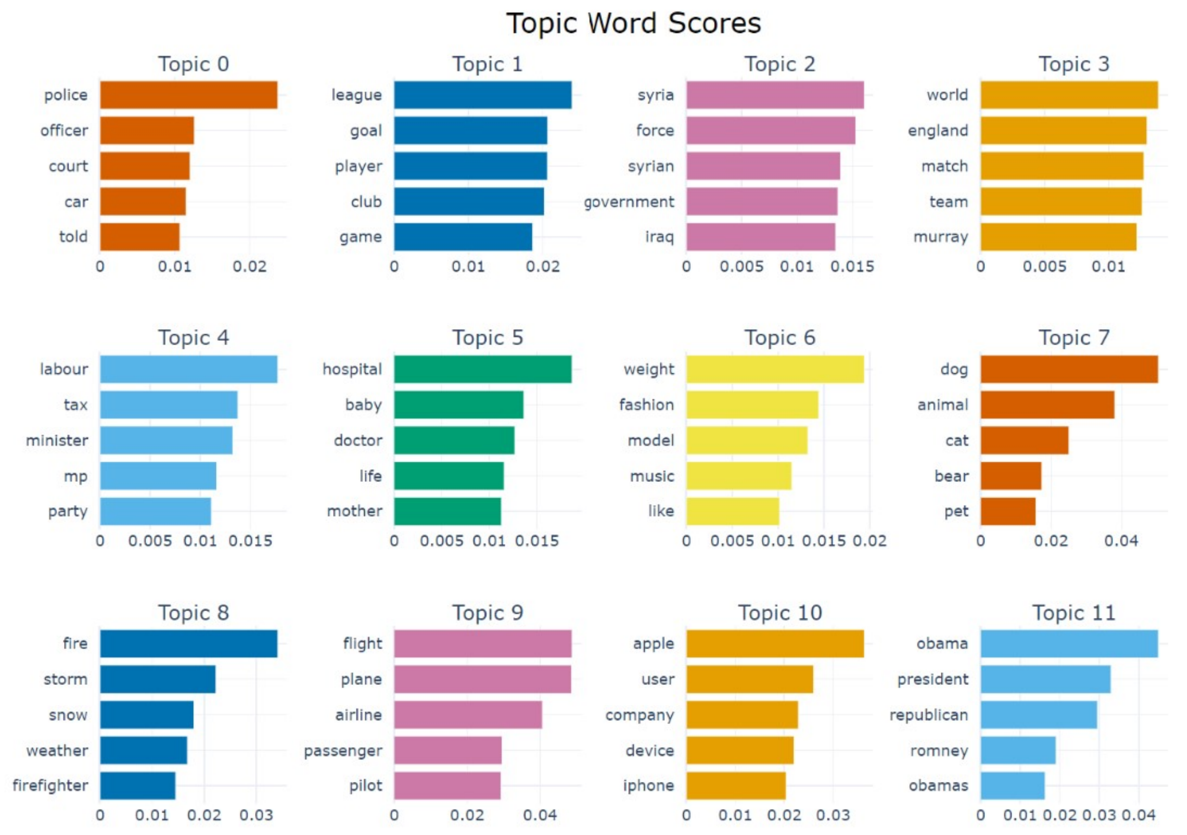

Figure 1: Topics extracted by BERTopic, sample size 10.000

3.4.3 Top2Vec

Top2Vec gives the possibility to easily build the wordclouds of the top words of any topic.

For example, considering the sample of 10.000 documents,

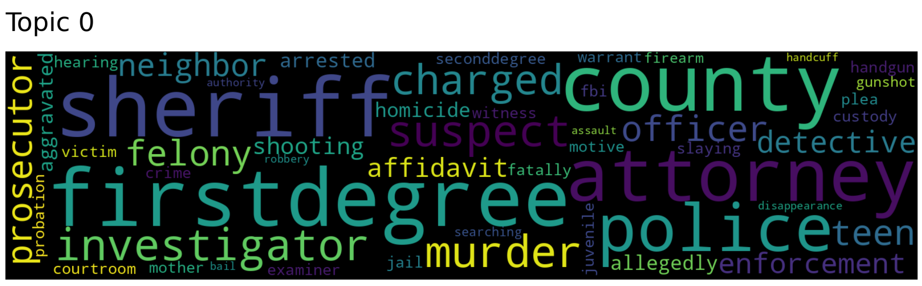

Figure 2: Topic words related to the cluster of the word *police*.

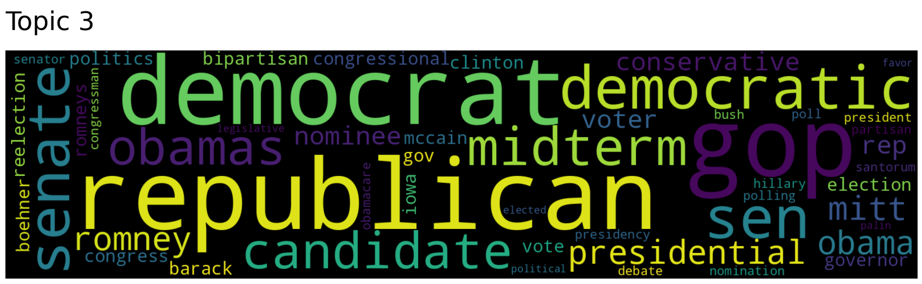

Figure 3: Topic words related to the cluster of the word *Obama*.

Topic modeling can provide important and interesting highlights on how a certain topic is being addressed. CNN is an American news network, and here are the results of searching for words related to two countries bordering the USA, Canada and Mexico.

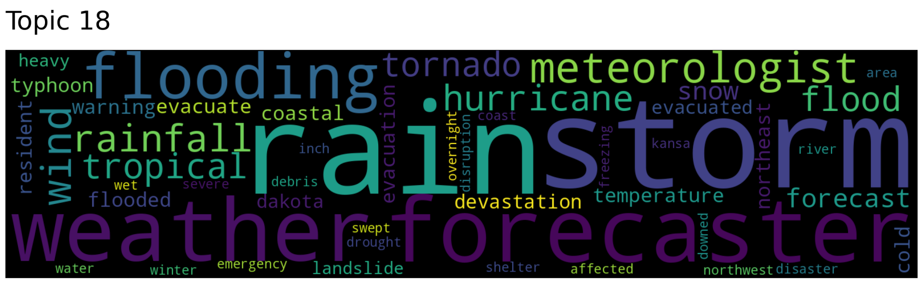

Figure 4: Topic words related to the cluster of the word *Canada*.

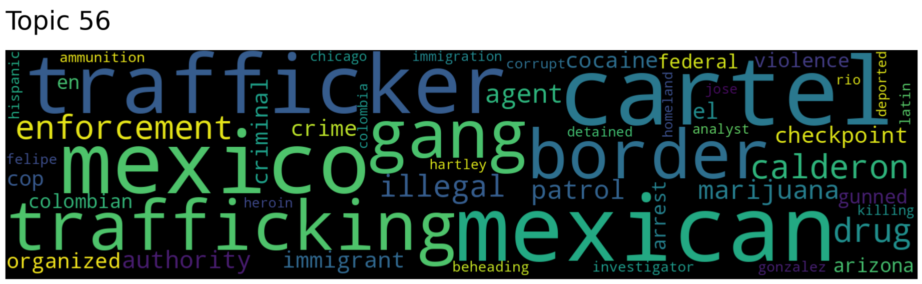

Figure 5: Topic words related to the cluster of the word *Mexico*.

While in the former terms related to the weather appear, in the latter there are terms related to the traffick of drugs and illegal practices, as well as immigration.

3.5 Appendix B

Source Text without preprocessing

A Russian father who fed parrots, guinea pigs, cats and puppies to his pet snake and filmed the gruesome footage is being hunted by police. Father-of-two Andrei Generalov, 32, from St. Petersburg in north-western Russia, started uploading videos of his Boa Constrictor 'King' devouring rats and hamsters. But as the videos rose in popularity, he began taking suggestions from viewers who demanded more 'blood and gore'. With this in mind, he started feeding 'King' larger animals in the sickening videos, which included terrified puppies. In the videos, he tells his audience: 'Why these senseless sacrifices? It's for the sake of food (ha ha). 'I like to film the way the animals fight for their lives, suffer and eventually die.' Scroll down for video . Andrei Generalov, 32, from St. Petersburg, feeds animals to his pet snake 'King' and uploads the video online . In this video, the Boa Constrictor curls round a petrified guinea pig with its muscular body before suffocating it . An excited Mr Generalov says: 'I like to film the way the animals fight for their lives, suffer and eventually die' The guinea pig tries to wriggle free but the snake has jaws lined with small hooked teeth, for holding their prey . A petition has started in St Petersburg, Russia, by those appalled by the suffering of the animals . But furious campaigners - outraged at the appalling suffering - began a petition to take Mr Generalov to court. After collecting more than 4,000 signatures, they handed it to the prosecutors office who ordered police to arrest him. Chief Prosecutor Sergey Litvinenko said: 'An arrest warrant has been put out for this man but he has now disappeared. 'Anyone with information of his whereabouts should contact us or the police directly.' Local petition signer Tigran Evdokimov, 28, said: 'What this man did was sickening beyond belief. 'To film such cruelty and to enjoy it is the work of a very sick and dangerous man.' But his friend, Leopold Polyakov, 35, defended Mr Generalov describing him as a 'good, family man'. He said: 'Andrei is a very good man, a family man. He has three cats at home and has never shown any sign of cruelty towards people or animals. 'I find it just incomprehensible that he would do this.' Boa Constrictors will eat almost anything they can catch, including birds, monkeys and wild pigs . In the videos, the Russian father tells his audience: 'Why these senseless sacrifices? It's for the sake of food' He started feeding his snake, 'King', hamsters and rats but moved on to bigger animals, including puppies . The boa constrictor is a species of large, non-poisonous snake. It is a member of the Boidae family and is found in the wild in North, Central and South America, as well as some islands in the Caribbean. They live for between 20 and 30 years, reach up to 13ft in length and can weigh up to 60lbs or 27kgs. Like their cousins, Anacondas, Boas are excellent swimmers but prefer to stay on dry land. In the wild they tend to live in hollow logs and mammals burrows. Their jaws are lined with small hooked teeth, for grabbing and holding their prey, while wrapping their muscular bodies around their victim, squeezing it until it suffocates. Boas will eat almost anything they can catch, including birds, monkeys and wild pigs. Their jaws can stretch to allow them to swallow their prey whole. Female boas incubate their eggs outside their bodies and can give birth to up to 60 babies during their lives. Some boa constrictors are hunted for their fine, ornate skin and as such, many of the species are endangered.

Target Text without preprocessing

Andrei Generalov, 32, uploads videos of his Boa Constrictor eating animals. `\n`Feeds his snake called 'King' parrots, guinea pigs, cats and even puppies. `\n`Boas have jaws lined with hooked teeth for grabbing and holding prey. `\n`They wrap their bodies around their victim, squeezing it until it suffocates. `\n`Petition started to take him to court and he is now being hunted by police.

In the next page you can find the **Predicted Summaries**.

Summarizer	**Predicted Summary**	**ROUGE**
Text Rank	Excited Mr. Generalov says like a film, where animals fight, live, suffer, and eventually die. A guinea pig tries to wriggle free from a snake's jaws lined with small hooked teeth holding its prey. The three cats at home have never shown any signs of cruelty towards people or animals. It's like a film, where animals fight, live, suffer, and eventually die. In his mind, he started feeding a king, which included larger animals and sickening videos, including terrified puppies. For the sake of food, he started feeding the snake like a king, with hamsters, rats, and even larger animals, including puppies.	rouge1: 0.33 rouge2: 0.06 rougeL: 0.06 avg rouge: 0.22
LSA	A Russian father who fed parrots, guinea pigs, cats and puppies to his pet snake and filmed the gruesome footage is being hunted by police. Father-of-two Andrei Generalov, 32, from St. Petersburg in north-western Russia, started uploading videos of his Boa Constrictor 'King' devouring rats and hamsters. An excited Mr Generalov says: 'I like to film the way the animals fight for their lives, suffer and eventually die' The guinea pig tries to wriggle free but the snake has jaws lined with small hooked teeth, for holding their prey . Their jaws are lined with small hooked teeth, for grabbing and holding their prey, while wrapping their muscular bodies around their victim, squeezing it until it suffocates.	rouge1: 0.55 rouge2: 0.31 rougeL: 0.31 avg rouge: 0.47
Lunh	Father-of-two Andrei Generalov, 32, from St. Petersburg in north-western Russia, started uploading videos of his Boa Constrictor 'King' devouring rats and hamsters. An excited Mr Generalov says: 'I like to film the way the animals fight for their lives, suffer and eventually die' The guinea pig tries to wriggle free but the snake has jaws lined with small hooked teeth, for holding their prey. Boa Constrictors will eat almost anything they can catch, including birds, monkeys and wild pigs. Boas will eat almost anything they can catch, including birds, monkeys and wild pigs.	rouge1: 0.37 rouge2: 0.11 rougeL: 0.11 avg rouge: 0.28
KL	A Russian father who fed parrots, guinea pigs, cats and puppies to his pet snake and filmed the gruesome footage is being hunted by police. 'I like to film the way the animals fight for their lives, suffer and eventually die.' An excited Mr Generalov says: 'I like to film the way the animals fight for their lives, suffer and eventually die' The guinea pig tries to wriggle free but the snake has jaws lined with small hooked teeth, for holding their prey . It's for the sake of food' He started feeding his snake, 'King', hamsters and rats but moved on to bigger animals, including puppies.	rouge1: 0.38 rouge2: 0.12 rougeL: 0.12 avg rouge: 0.29
Pegasus	Andrei Generalov, 32, from St. Petersburg uploads videos of his Boa Constrictor 'King' devouring rats and hamsters. As the videos rose in popularity, he began taking suggestions from viewers who demanded more 'blood and gore' He started feeding 'King' larger animals in the sickening videos, which included terrified puppies. A petition has started in St Petersburg, Russia, by those appalled by the suffering of the animals.	rouge1: 0.30 rouge2: 0.11 rougeL: 0.11 avg rouge: 0.23
T5 Small	Andrei Generalov, 32, from St. Petersburg, took suggestions from viewers . He started feeding his snake, 'King', hamsters and rats but moved on to bigger animals, including puppies . But furious campaigners - outraged at the appalling suffering - began a petition to take Mr	rouge1: 0.11 rouge2: 0.01 rougeL: 0.01 avg rouge: 0.07