Text Mining and Search Project
January 2026 Exam session

Topic Modeling on Summarized News Articles

Giovanni Noè, Francesco Volpi Ghirardini

Abstract
In Natural Language Processing (NLP), topic modeling and automatic text summarization are essential techniques for extracting insights from large volumes of unstructured text. This project investigates the interaction between these two tasks by analysing news articles from the CNN/Daily Mail dataset. Topic modeling algorithms are applied to both the original articles and their automatically summarized versions to assess how summarization impacts the quality and interpretability of the extracted topics. Several topic modeling methods are evaluated, and among various extractive summarization techniques, the most effective one is selected to generate summaries, after which topic modeling is performed again on the summarized texts. Experimental results show that applying extractive text summarization reduces computational costs and, depending on the topic modeling algorithm, can improve the coherence and interpretability of the extracted topics.

Table of Contents

Introduction

In the field of natural language processing (NLP), topic modeling is a widely used technique for uncovering latent themes in large collections of text. By identifying clusters of co-occurring words, topic models allow researchers to gain insights into the underlying structure of textual data without requiring manual annotation.

Extractive text summarization is another fundamental NLP task aimed at producing concise representations of longer documents while retaining the most important information. Unlike abstractive summarization, which generates new sentences, extractive methods aim to select the most informative sentences directly from the original text. This approach is often preferred in news analysis and other domains where preserving factual accuracy and context is critical.

The CNN and Daily Mail news articles dataset [1][2], which is widely used in text summarization research, serves as the primary dataset for this work. It contains a large-scale collection of news articles along with human-written summaries, offering a rich source for evaluating automatic summarization techniques.

The objective of this project is to investigate the relationship between topic modeling and text summarization, with the aim of evaluating whether topic extraction from summarized texts can enhance topic modeling results. The central hypothesis is that extractive text summarization preserves the sentences that most effectively represent the main topics of each article, potentially reducing noise in the input data. Consequently, topic modeling applied to summarized texts may lead to improved efficiency and clearer topic representations. It is important to note that many extractive summarization techniques are based on topic representation approaches. However, since the objective of this work is to independently perform topic modeling both before and after summarization in order to compare the results, the project focuses exclusively on non-topic-based extractive summarization methods, specifically those based on indicator representations. This choice ensures that the summarization process remains as independent as possible from the topic modeling stage, avoiding potential bias in the comparison.

This work follows a standard NLP pipeline, beginning with data cleaning and exploratory analysis, followed by data processing and text representation. Different representations are employed depending on the model and task, including term frequency (TF), TF-IDF, word embeddings, and sentence embeddings.

For topic modeling, the implemented methods include both classical latent topic models and embedding-based ones. All models are evaluated using visual topics inspection, topic diversity and topic coherence measures, and by the time they took to complete the execution. Models that need the number of topics to find as an input are passing through a fine-tuning process for that parameter.

For text summarization, two graph-based algorithms, a Machine Learning-based one and a clustering based one are implemented, and the resulting summaries are evaluated with ROUGE metrics. The summaries generated by the best-performing model are then used again as input in the topic modeling pipeline.

Reproducibility of The Results

To ensure the replicability of this study and its results some cautionary steps must be taken. First the code is executed on a Google Colab virtual environment using its T4 GPU, computational times reported in this work should be interpreted in this context as performance may vary on different hardware. Furthermore, all the functions that need to initialize pseudo-random processes are set with a fixed seed.

1. The Dataset

The CNN/Daily Mail dataset is a large-scale news corpus originally created to support research in automatic text summarization. However, due to its wide diversity of topics, it is also well suited for other unsupervised NLP tasks, such as topic modeling.

The dataset consists of news articles collected from the CNN and Daily Mail websites, each paired with human-written highlights that serve as reference summaries. These highlights capture the key points of the articles in a concise and fact-focused manner, making the dataset particularly valuable for both abstractive and extractive summarization.

Each data instance in the CNN/Daily Mail dataset typically contains two main components: the full news article and a set of bullet-point highlights, which serve as reference summaries. Many instances also include metadata such as authorship, publication date, or update date.

The news articles vary widely in length, style, and subject matter, covering domains such as politics, business, health, technology, entertainment, and sports. This diversity provides a broad topical distribution, which is beneficial for evaluating the robustness and generalization of topic modeling algorithms. However, this variety can also pose challenges, as some articles consist of short, disjointed phrases rather than coherent, human-readable news stories.

Another important characteristic of the CNN/Daily Mail dataset is that the highlights (i.e. reference summaries) are not direct extracts but are written by humans, meaning they represent an abstractive compression of the original content. This makes the dataset particularly useful for evaluating extractive summarization methods by measuring how well selected sentences approximate the information contained in the reference highlights.

Overall, the CNN/Daily Mail dataset provides a rich combination of long-form news articles, high-quality human-written summaries, and broad topical diversity. These features make it an ideal resource for investigating the impact of extractive text summarization on topic modeling in large-scale news corpora.

The following sections provide a detailed overview of the data cleaning process, and some insights from the data explorative analysis, to understand structure and content distribution of the dataset.

1.1 Data Cleaning

Before applying topic modeling or summarization techniques, it is essential to preprocess the raw dataset to remove noise and ensure consistency. The CNN/Daily Mail articles, as collected from online sources, often contain formatting artifacts, irrelevant text, and special tokens that can interfere with textual analysis. Data cleaning transforms the raw text into a structured and standardized format suitable for NLP tasks, improving the reliability of topic extraction and reducing the impact of spurious patterns on model performance.

The first step involves reducing the number of documents to keep the computational requirements, in terms of RAM usage, GPU resources and execution time, manageable. For this purpose, a subset of 50,000 articles is selected, with an equal number drawn from each source.

The CNN and Daily Mail articles differ in formatting, structure, and the types of noise present, which necessitates slightly different cleaning pipelines for each source.

Cleaning on CNN

The data cleaning pipeline for CNN articles involves the following steps:

- Removing articles without summaries: articles lacking reference summaries are excluded to ensure that each instance can be used for evaluating summarization models.
- Removing headers: headers, which include authorship and places, are removed to prevent irrelevant content from affecting the analysis.
- Removing footers: many articles conclude with the phrase "E-mail to a friend", these footers are removed to maintain focus on the article's main content, which is particularly important for accurate topic modeling.

Cleaning on Daily Mail

The Daily Mail dataset also contains certain header patterns that need to be removed to ensure consistency. These headers often include metadata such as the publication date, update date, and creation date. Although these patterns are not standardized across articles, any occurrences that are identified are deleted during the cleaning process.

Further Cleaning Operations

After completing the organization-specific cleaning, the CNN and Daily Mail datasets are merged, and additional preprocessing steps are applied:

- Removing non-breaking space characters: all instances of \xa0 are removed.
- Correcting missing sentence boundaries: some articles contain \n\n sequences not preceded by punctuation, full stops are inserted in these cases to improve sentence tokenization.
- Removing web links: all hyperlinks, in any format, are removed from the text.
- Shuffling articles: the dataset is shuffled with a fixed random seed to ensure reproducibility.
- Extracting final data: the cleaned news articles and their corresponding summaries are extracted into separate variables, docs and summary_labels, respectively.

Additional cleaning steps will be performed after the exploratory data analysis to further polish the dataset.

1.2 Exploratory Analysis

Once the dataset has been cleaned and standardized, exploratory analysis is conducted to gain a deeper understanding of its structure. This procedure also helps identify patterns, and potential biases in the data, informing decisions about model selection, and preprocessing strategies for subsequent topic modeling and summarization tasks.

The distribution of the number of sentences per document is first examined using a histogram, showed in figure 1.1. This visualization highlights the variability in the number of sentences across the documents and helps identify typical article sizes as well as outliers that may require further attention.

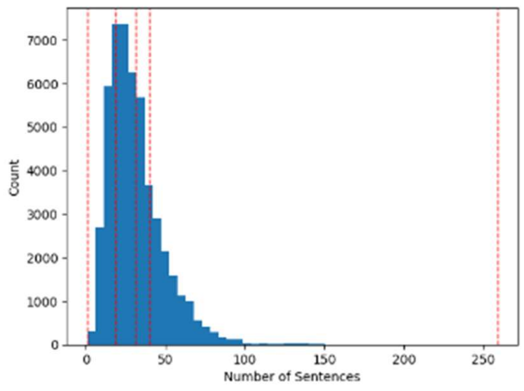

***Figure 1.1** – Distribution of the numbers of sentences in the documents of the sample*
Dotted lines represent: min, 1st qt, mean 3rd qt, max

The distribution is highly right-skewed, with more than 75% of the documents containing between 1 and 50 sentences, while at least one article extends to over 250 sentences. This indicates that most news articles are relatively short, but the presence of a few very long documents contributes to the heavy tail of the distribution. Since these extremely long documents are few and often contain content of limited relevance, all articles with more than 145 sentences, along with their corresponding summaries, are removed from the dataset.

A similar analysis is performed on the number of sentences in the summaries, beginning with the histogram in figure 1.2, which helps to visualize their distribution. This allowed for the identification of unusually long or short summaries that could affect text summarization.

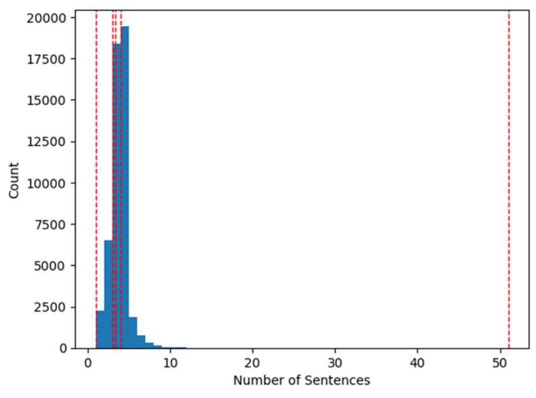

Figure 1.2 *– Distribution of the numbers of sentences in the summaries of the sample*
Dotted lines represent: min, 1st qt, mean 3rd qt, max

The distribution of summary lengths is also right-skewed, with most of the summaries containing between 1 and 5 sentences. However, there is at least one extreme case where the summary exceeds 50 sentences. As with the articles, these extreme summary cases are few and generally not informative. Therefore, all summaries containing more than 15 sentences, along with their corresponding documents, are removed from the dataset.

Before proceeding with the analysis on the subset, the same exploratory data analysis plots are generated for the original dataset to verify that the distribution of the number of sentences of the subset is consistent.

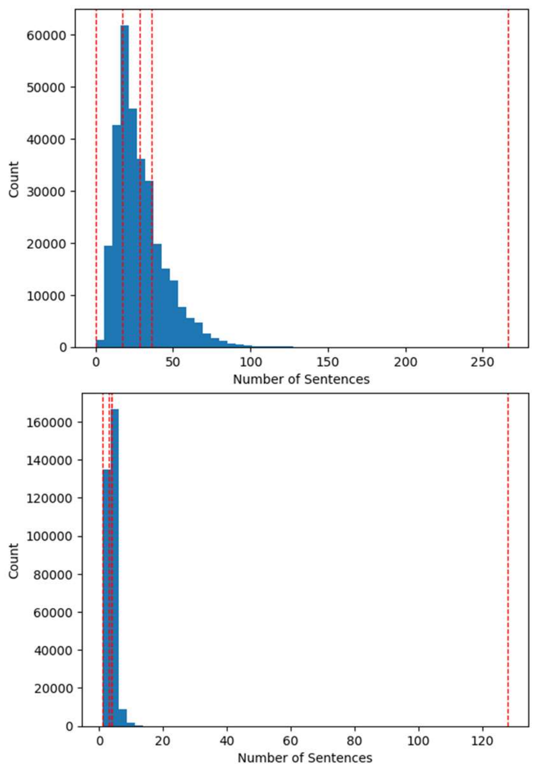

Figure 1.3 *– (a) Distribution of the numbers of sentences in the original documents (b) Distribution of the numbers of sentences in the original summaries*
Dotted lines represent: min, 1st qt, mean 3rd qt, max

Since the plots in figure 1.3 (a)/1.3 (b) and those in figure 1.1/1.2 are very similar, the subset is considered representative of the full dataset.

After cleaning and filtering, the dataset consists of 49,887 document-summary pairs, with the documents containing a total of 1,565,885 sentences (an average of approximately 31.39 sentences per document) and the summaries containing 166,940 sentences (an average of approximately 3.35 sentences per summary).

2. Topic Modeling on Full Articles

Topic modeling is a key technique in natural language processing for uncovering latent thematic structures in large text corpora. By analysing words' distribution across documents, topic modeling algorithms aim to identify clusters of semantically related terms that correspond to underlying topics. These methods provide a way to summarize, organize, and explore large collections of unstructured text without requiring manual annotation. Topic modeling has applications in areas such as information retrieval, document classification, trend analysis, and content recommendation.

In this project, four representative topic modeling methods are evaluated: LSA [3], LDA [4], pLSA [5], and BERTopic [6]. Each of these models differs in its approach to represent textual content, offering a comprehensive comparison of both classical and modern topic modeling techniques. Each model is evaluated through a combination of qualitative and quantitative criteria: manual inspection of the extracted topics, topic diversity and coherence scores, and the computational time required for execution.

2.1 Text Processing and Representation

Traditional models, such as LSA, LDA, and pLSA, rely on bag-of-words representations, with LDA and pLSA typically using term frequency (TF) and LSA often using TF-IDF. In contrast, more recent embedding-based approaches, such as BERTopic, leverage contextual word embeddings and clustering techniques to generate topics that capture richer semantic relationships and contextual information. The diversity of these modeling approaches necessitates a comprehensive text processing pipeline and the introduction of multiple text representation techniques.

Text Processing

Before applying topic modeling algorithms, the text data undergoes a series of preprocessing steps to clean the documents while preserving meaningful content. First, a custom list of stop words is created and combined with the standard stop word list provided by Natural Language Toolkit (NLTK) [7]. This ensures that common, non-informative words, both general and domain-specific, are removed from the analysis. The custom stop words and the reason of their deletion are the following:

- Common verbs and reporting words (e.g. say, be, said, says, saying, tell, told, report): these words are highly frequent in news articles and often do not contribute to distinguishing topics. They mostly serve grammatical or reporting functions (e.g., "he said", "she told"), so removing them reduces noise and improves topic coherence.
- General adverbs and connectors (e.g. also, according): these words typically appear in multiple contexts without carrying specific semantic content. Removing them prevents them from dominating term frequency statistics and diluting meaningful topics.
- Year references (e.g. year, years): while very common in news articles, these words rarely help differentiate topics. Removing them allows the model to focus on substantive content rather than temporal references.
- People references (e.g. people, man, men, woman, women, mr, mrs, ms): these terms are highly generic references to humans or titles and appear across almost all news articles. They are removed because they do not indicate any specific topic and can dominate the vocabulary if left in.
- Numerical words (one, two, three, …, ten): numbers written as words frequently appear in statistics, rankings, or lists but do not provide thematic information. Excluding them helps prevent topics from being dominated by numeric terms.

Next, each document is cleaned by stripping all non-alphabetic characters, followed by lowercasing, tokenization and lemmatization. During this process, words that appear in the stop word list are discarded, leaving only meaningful, normalized terms that better capture the semantic content of the text. Lemmatization reduces inflected forms of words to their base forms, which helps improve the consistency of the vocabulary across the corpus.

To support topic modeling and coherence evaluation, a Gensim [8] dictionary is created from the processed documents. Terms included in the dictionary must appear in at least 0.1% of the documents but no more than 80%, and the total number of terms is capped at 20,000. This dictionary is subsequently used for constructing the document-term matrix required by count-based models, as well as for computing topic coherence scores to assess the quality and interpretability of the extracted topics.

Text Representation

The choice of text representation plays a critical role in topic modeling, as it determines how documents are encoded for analysis. Depending on the model, different representations capture varying aspects of semantic content, ranging from simple frequency counts to contextual embeddings. In this work, three primary types of representations are used: Term Frequency (TF), Term Frequency-Inverse Document Frequency (TF-IDF), and BERT [9] word embeddings:

- The TF representation captures how often each word occurs in a document, allowing models such as LDA and pLSA to identify patterns of co-occurrence across the corpus.
- The TF-IDF representation improves upon raw term frequency by weighting terms according to their importance. TF-IDF is particularly useful for LSA, where the underlying linear algebra techniques benefit from emphasizing distinctive terms.
- Word embeddings provide dense, continuous vector representations of words that capture semantic relationships. In this project, embeddings are used with BERTopic, specifically employing the all-MiniLM-L6-v2 BERT model [10], a compact and efficient encoder that produces high-quality sentence embeddings for semantic analysis.

2.2 Models

This study employs a variety of topic modeling algorithms to extract meaningful themes from news articles, ranging from traditional probabilistic models to modern embedding-based approaches.

LSA, pLSA, and LDA require the number of topics to be specified as an input parameter. Since this value is not known a priori, in this project, a systematic fine-tuning procedure is adopted. Each of the three traditional models is run on a subset of 10,000 documents using different numbers of components in different ad hoc sets for each model. Also, for each value, the model is executed three times with different random seeds to account for variability in the results. For each model, the number of components that achieves the best performance according to the evaluation metrics is selected, and the corresponding results are discussed in the following analysis. The optimal number of topics determined by this means are 10 for LSA, 20 for LDA, and 30 for pLSA. BERTopic is excluded from this tuning process, as it does not require the number of topics to be specified in advance and instead determines the topic structure automatically through clustering.

The following subsections describe on a high level the models used in this work.

Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) is a linear algebra-based technique that uses singular value decomposition (SVD) to reduce the dimensionality of a term-document matrix constructed using term TF-IDF representations. LSA is grounded in the distributional hypothesis, which states that the meaning of a word can be inferred from the contexts in which it appears. Under this assumption, words that frequently co-occur in similar contexts are considered semantically related. LSA captures these co-occurrence patterns while ignoring syntactic and other higher-order semantic information.

In this project, LSA is implemented using the TruncatedSVD function from the scikit-learn library [11].

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a probabilistic topic modeling method that uncovers latent themes within a collection of documents. It assumes that each document is composed of a mixture of topics, and each topic is represented as a probability distribution over words. Using a normalized TF representation, LDA infers the hidden topic structure by estimating the relationships between documents and topics, as well as between topics and words.

In this project, LDA is implemented using the LatentDirichletAllocation function from the scikit-learn library, with default parameters except for the number of components, which is tuned as part of the analysis.

Probabilistic Latent Semantic Analysis (pLSA)

Probabilistic Latent Semantic Analysis (pLSA) is a generative probabilistic model that extends LSA by modeling the probability distribution of topics within documents. Using a TF-based term-document matrix, pLSA estimates the likelihood that a topic appears in a document and that a given word belongs to a topic. This probabilistic framework allows for more interpretable topic assignments compared to standard LSA.

In this work, pLSA is implemented using scikit-learn's NMF function with beta_loss="kullback-leibler" and solver="mu", which enables the factorization of the term-document matrix in a manner consistent with the probabilistic interpretation of pLSA.

BERTopic

BERTopic is a modern embedding-based topic modeling approach that leverages contextual word embeddings and clustering algorithms to extract topics. In this work, BERTopic employs a pretrained Transformer-based model to produce the embeddings that are subsequently clustered to identify coherent topic groupings. Unlike traditional bag-of-words approaches, BERTopic captures semantic relationships between words, producing topics that are often more coherent and interpretable in practice.

BERTopic is of course implemented using the BERTopic function from the bertopic library.

2.3 Evaluation

In the context of topic modeling no single metric is sufficient to fully capture the quality of the extracted topics. For this reason, multiple complementary evaluation dimensions are considered. Specifically, each model is evaluated through manual inspection of the extracted topics to assess their interpretability and semantic coherence, quantitative measures such as topic diversity and coherence scores to provide objective comparisons, and the computational time required for execution to account for efficiency and scalability. Together, these criteria offer a comprehensive view of both the effectiveness and practicality of each topic modeling approach.

Visual Inspection

Topics extracted by each model are manually examined to assess their interpretability and semantic meaningfulness, providing a qualitative measure of topic quality.

By examining a sample of the topics extracted by each model, it is evident that LSA performs the worst, as most of its topics feature the word "police" and generally lack semantic coherence. In contrast, the other three models produce topics that are both semantically cohesive and diverse, indicating strong performance.

Metrics Evaluation

Topic coherence provides an objective assessment of the semantic consistency among the top words of each topic. In this work, coherence is computed for each model using the CoherenceModel function from the Gensim library [13], employing the c_v variant of the measure.

Topic diversity, on the other hand, is measured to evaluate how distinct the topics are from each other, ensuring that the models capture a broad range of themes rather than overlapping or redundant topics. In this project, diversity is computed using eq. 2.1.

$$1 - \frac{1}{n}\sum_{i \neq j} Sim(t_i, t_j) \tag{2.1}$$

Where n is the number of topics, t_i and t_j are the TF-IDF representation of the top ten words of topics i and j and $Sim(t_i, t_j)$ is the cosine similarity between them.

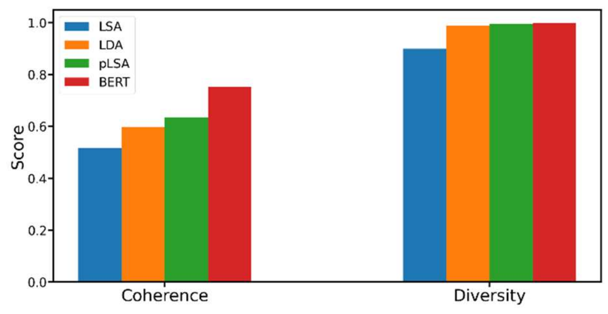

***Figure 2.1** – Scores on the metrics for the topics*

Coherence and diversity scores of the models used in this work are presented in figure 2.1. From the plot, the differences in model performance are evident. Coherence scores show LSA performing poorly, around 0.5, while LDA and pLSA achieve scores near 0.6, and BERTopic outperforms all with a score above 0.7. Diversity scores follow a similar ranking: LSA falls noticeably below the others at just under 0.9, whereas LDA, pLSA, and BERTopic all achieve values close to 1, indicating that these models capture a broader and more distinct range of topics.

Time Evaluation

The execution time for each model is recorded to assess efficiency and scalability, highlighting the trade-offs between model complexity and practical usability.

Table 2.1 presents the computational times for all models. For the bag-of-words models, the reported times include both tuning and tokenization. For BERTopic, the execution time also accounts for the generation of word embeddings.

Model	Tokenization	Tuning	Fitting	Total
LSA	3m:15s	36m:30s	0m:03s	39m:48s
LDA	3m:15s	77m:29s	7m:01s	87m:45s
pLSA	3m:15s	122m:02s	25m:09s	150m:26s
BERTopic	---	---	41m:25s*	41m:25s

***Table 2.1** – Time of the steps to obtain the topics*
**also includes embedding time*

The recorded computational times reveal clear differences in efficiency among the models. For the bag-of-words models (LSA, LDA, and pLSA), tokenization takes roughly 3 minutes. Fine-tuning dominates the computational cost, with pLSA being the most time-consuming at over 2 hours. Final execution is comparatively negligible for LSA and LDA but becomes significant for pLSA that took an additional 25 minutes.

Considering the total computational time, BERTopic appears faster than LDA and pLSA; however, if fine-tuning for the traditional models is excluded, their execution times are significantly lower. This is especially true for LDA, which, while achieving mediocre results on other evaluation metrics, is considerably faster than BERTopic, representing a good and computationally cheaper alternative to the embeddings-based model.

3. Text Summarization

In the exploration section it was discovered that the average number of sentences the articles have is thirty, all this data might contain some uninteresting or redundant information that make the topic modeling procedure slower or more inefficient. The aim of this section is to then reduce the number of sentences to five, which is a number close to that present in the summaries but that still enables the documents to have enough information so that the topic modeling algorithms can be tested again.

To get the summaries with the highest quality possible four different algorithms based on a wide range of extractive summarization techniques are going to be tested and compared. The first two are going to have a deterministic vectorization of the data and graph-based scoring, the third is going to have its sentences encoded with a Transformer-based embedding model and scores computed with a machine learning model. The fourth, and final, algorithm is going to be a combination of the two approaches just described, having its sentences embedded with a Transformer but evaluated in a deterministic way.

Given that one of the algorithms presented features a trainable model the document data is split into a training set and a test set, with the first one containing 85% of the original data. The three algorithms that score sentences in a deterministic manner are only going to summarize the test data so that a fair comparison can be made.

3.1 Models

The first three algorithms that are going to be presented are custom made and work following roughly the same pattern: the sentences are first vectorized, then a function scores them and, based on the latter, the sentences are finally going to be ranked using the Maximal Marginal Relevance function in eq. 3.1.

$$MMR(s_i) = \lambda \cdot score(s_i) - (1-\lambda) \cdot \max_{s_j \in S} Sim(s_i, s_j) \quad (3.1)$$

Where λ is the MMR parameter, in all the implementations is set to 0.3 to give a smaller weight to redundancy, $score(s_i)$ is the importance of the sentence according to the model, S the set of the already selected sentences, $Sim(s_i, s_j)$ is the similarity between sentences i and j. In all of the implementations this similarity is going to be computed on the same vectors used for the representation of the sentences.

The first three algorithms are also going to have their input sentences tokenized with NLTK first.

First Algorithm: TF-IDF + HITS

This first graph-based architecture is the simpler of the two since it does not make use of embeddings and its scoring algorithm is mathematically simpler. The sentences are first normalized in the same way it has been shown in section 2.1, minus the lemmatization which is not needed, and then vectorized through TF-IDF vectorization. These vectors are used to compute a cosine similarity matrix on which a NetworkX [12] graph is created. The nodes of this graph (which represent single sentences) are then evaluated with the Hyperlink-Induced Topic Search algorithm [13] so that their scores, together with the similarities from the matrix created earlier, can be used to rank all the sentences through the MMR function.

Second Algorithm: GloVe + PageRank

Going onto a more intricate graph-based topic extraction algorithm a modified version of TextRank [14] is implemented. The sentences are first normalized in the same way it was shown in the previous section to then embed their words using GloVe [15] embeddings. Since this step already stored the vectors into torch tensors, an approximation of cosine similarity is obtained by first normalizing the vectors and then performing a dot product between them and their transpose. This matrix is used to create a NetworkX graph whose nodes are scored with the PageRank algorithm [16] and, together with the similarities from the matrix created earlier, are ranked using the MMR function.

Third Algorithm: BERT + XGBoost

With the recent interest in machine learning solutions for Text Mining tasks, an algorithm that uses these models during most of its steps is tried and compared to the more traditional solutions. The first of these steps is to encode the training and test documents' sentences using the all-MiniLM-L6-v2 [8] pretrained Transformer.

Before the classification model can be trained labels are needed and so for each document the first five training sentences scoring the highest MMR computed on both on average F1 score on the three ROUGE metrics and cosine similarity of the embeddings are selected as the positive class. With these labels an XGBoost [17] classifier is trained using the hyperparameters shown in table 3.1, which were chosen based both on informed guesses and a few trials (note: the scale_pos_weight was chosen on the presence of five positive classes for each twenty-five negatives).

Hyperparameter	Value
n_estimators	100
max_depth	5
learning_rate	0.1
reg_alpha	0.1
reg_lambda	1
scale_pos_weight	5

Figure 3.1 *– XGBoost classifier hyperparameters*

The test sentences are finally ranked and selected based on the MMR computed on their probability score of being part of the positive class given in output from the classifier. The use of MMR is particularly important for this algorithm since it introduces sentence-level relativity within the same document, which XGBoost could not model.

BERT Extractive Summarizer

This final algorithm [18] has been fully imported from a GitHub repository [19] and first automatically preprocesses the sentences present in the raw document data, then uses a pretrained BERT model to tokenize and embed them. The pretrained Transformer model used to generate embeddings chosen for this project is distilbert-base-uncased [20], since it's what the creator of the repository uses in his examples.

The embeddings obtained in the previous step are clustered using KMeans so that the sentences closest to the centroid can be selected for the final summary.

3.2 Evaluation

Following the same reasoning made in the evaluation section for topic modelling different dimensions are going to be considered to give a more holistic evaluation of the summaries. To further help this process two bounds are going to be added for the metrics: a random summarizer, that selects a random n sentences from a document, and a theoretical upper bound, obtained by choosing sentences based on both their average F1 score for the ROUGE metrics with cosine similarity computed on BERT embeddings.

Metrics Evaluation

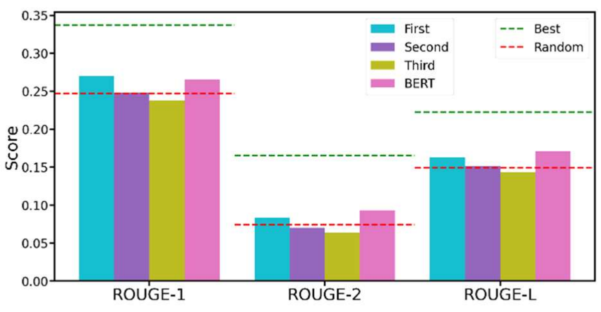

Figure 3.1 *– Scores on the metrics for the summaries*

In figure 3.1 the F1 scores for ROUGE-1, ROUGE-2 and ROUGE-L of the four models are presented, alongside the baseline and theoretical upper bound. The first thing that can be noticed is that all models scored well below the theoretical upper bound, with the second and third algorithms also performing the same as the random summarizer. However, the first summarizer and BERT extractive summarizer managed to perform better than just random chance, with the first of the two achieving a slightly higher score for ROUGE-1 while the second gave the highest score in the other two metrics.

Time Evaluation

Given the vastly different algorithm structure and the equally dissimilar processes the data go through in each of them, time evaluation can be of particular interest.

Model	Embedding	Training	Test	Total
First	---	---	1m:04s**	1m:04s
Second	---	---	1m:49s**	1m:49s
Third	15m:33s	31m:03s*	0m:12s	46m:48s
BERT	---	---	21m:56s**	21m:56s

Table 3.2 *– Time of the steps to obtain the summaries*
**also includes the time to create the labels*
***also includes vectorizatio/embedding time*

As it can be gathered from table 4.1 the algorithm featuring the ML model took the longest, not so much for the training of the model itself but rather for the creation of training labels. On the other hand, the first two architectures were fast, summarizing more than seven thousand articles in just over a minute.

Final Considerations

A single summarizer needs to be chosen as the best so that topic modeling can be performed on its summary and BERT extractive summarizer is selected for the task. Even though it took longer than the algorithm combining TF-IDF and HITS, its results on the metrics are ever so slightly better. Given this, the training data is also summarized with the same model and merged with the test summary, so that topic modeling can be executed on the full data and fairly compared to the first run. The total number of sentences in the documents have thus been brought from 1,565,885 sentences (with an average of 31.39 sentences per document) to 249,869 (with an average of 5 sentences per document).

4. Topic Modeling on Summaries

After generating the summaries, topic modeling is applied to the summarized corpus in order to analyse how text compression affects the discovery and structure of latent topics. The same modeling pipeline used for the original documents in section 2 is adopted here, allowing a direct and fair comparison between topics extracted from full-length articles and those obtained from their summarized versions.

The summarized documents, being significantly shorter than the original articles, provide a more compact representation of the content. On one hand, summarization may remove redundant or noisy information, potentially leading to clearer and more focused topics. On the other hand, compression can result in the loss of important contextual cues, which may reduce topic diversity or cause semantic drift.

For the technical details regarding preprocessing steps, text representations, model architectures, parameter settings, and evaluation metrics, the reader is referred to the topic modeling section on the original documents. All implementation choices remain unchanged; this ensures that any observed differences in topic diversity, coherence, or interpretability can be attributed primarily to the effect of summarization rather than to variations in modeling techniques.

4.1 Evaluation

The evaluation of topic modeling on the summarized documents follows the same procedure as for the original articles, using the same qualitative and quantitative criteria. Since the methodology remains unchanged, this section focuses solely on presenting the results and on the comparison of topic quality and model performance before and after summarization.

Visual Inspection

By examining a sample of the topics extracted from the summarized documents, it appears that LSA produces slightly better topics in terms of semantic cohesion. In contrast, both pLSA and LDA generate noticeably poorer topics, with a loss of semantic consistency within certain topics-particularly in the case of LDA. BERTopic, however, shows similar performance on summarized documents compared to its results on the full texts.

Metrics Evaluation

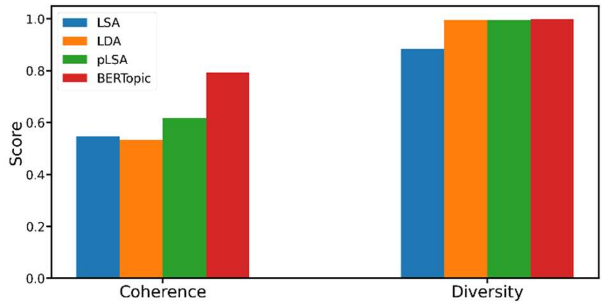

Figure 4.1 *– Scores on the metrics for the topics*

The coherence and diversity scores of the models applied to the summarized documents are presented in figure 4.1. From the plot, it is evident that LSA and BERTopic show an improvement in coherence compared to the non-summarized documents. In contrast, as anticipated from the visual evaluation, both pLSA and LDA experience a loss of coherence, with LDA showing a particularly pronounced drop. Regarding diversity, LSA exhibits a slight decrease, LDA increases, and the other models remain essentially unchanged.

It is noteworthy that the model exhibiting the most significant improvement on the summarized documents is also the best-performing model overall. This suggests that applying extractive summarization not only reduces computational costs but, depending on the model, can also enhance the quality and interpretability of the topics generated. In other words, the summarization step successfully preserves the most informative content, allowing the model to focus on the core themes of the articles. This result highlights the potential of combining summarization with topic modeling to achieve both efficiency and improved semantic coherence.

Time Evaluation

The following table presents the computational times of all models when applied to the summarized articles:

Model	Tokenization	Tuning	Fitting	Total
LSA	0m:44s	0m:47s	0m:01s	1m:32s
LDA	0m:44s	7m:59s	3m:04s	11m:47s
pLSA	0m:44s	7m:18s	2m:56s	10m:58s
BERTopic	---	---	38m:54s*	38m:54s

Table 4.1 *– Time of the steps to obtain the topics*
**also includes embedding time*

The computational times for the models on summarized articles are substantially lower than those recorded on the full dataset. Tokenization for the bag-of-words models (LSA, LDA, pLSA) takes less than one third of the time. Fine-tuning times also drop dramatically: LSA now takes less than a minute, while LDA and pLSA require under 8 minutes, compared to 77 and 122 minutes, respectively, on the full dataset. Final execution times are similarly reduced.

BERTopic shows a slight decrease in total execution time, from 41 minutes 25 seconds on the full articles to 38 minutes 54 seconds on summarized documents.

Overall, the reduction in document length through summarization leads to a significant improvement in computational efficiency, particularly for the traditional bag-of-words models, while BERTopic benefits only modestly of the documents' lengths reduction.

Conclusions

In conclusion, the interaction between extractive text summarization and topic modelling, with the performance of the latter being compared both before and after summarization, was extensively explored. With reference to everything discussed in the three evaluation sections several conclusions can be drawn. First, both the first topic modelling and text summarization experiments on the sample of the CNN/Daily Mail worked, with BERT based algorithms performing better than traditional algorithms on both tasks. The topic modelling test on the summarized data proved that this way of working can be of use, since the traditional models greatly sped up their fitting time and BERTopic increased its coherence score; furthermore LSA increased its coherence and LDA increased its diversity. There were still a few downsides, the summarization time was not worth the sped up of BERTopic and LDA and pLSA decreased their respective coherence scores while LSA had a slight drop in diversity.

With all of this being said it can be concluded that there are clearly some advantages to performing text summarization before topic modelling, even though this depends on the algorithm used and on the steps of the procedure.

References

[1] Hermann K. M., Kocisky T. Grefenstette E., Espeholt L., Kay W. Suleyman, M. Blunsom, P. (2015) *Teaching machines to read and comprehend*, Advances in Neural Information Processing Systems pp. 1684-1692

[2] Direct source of the data: *https://cs.nyu.edu/~kcho/DMQA/*

[3] Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., Harshman R. (1990) *Indexing by Latent Semantic Analysis*, Journal of the American Society for Information Science 41(6) pp. 391-407.

[4] Blei D. M., Ng A. Y., Jordan M. I. (2003) *Latent Dirichlet Allocation*, Journal of Machine Learning Research 3 pp. 993-1022.

[5] Hofmann T. (1999) *Probabilistic Latent Semantic Analysis*, Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999), pp 289-296.

[6] Grootendorst M. (2022) *BERTopic: Neural Topic Modeling with a Class-Based TF-IDF Procedure*, Proceedings of the 16th

Conference of the European Chapter of the Association for Computational Linguistics, System Demonstrations, 1-7.

[7] Bird, S., Klein, E., Loper, E. (2009) *Natural Language Processing with Python*, O'Reilly Media.

[8] Řehůřek R., Sojka P. (2010) *Software Framework for Topic Modelling with Large Corpora*, Proceedings of LREC 2010 Workshop on New Challenges for NLP Frameworks pp. 45-50.

[9] Devlin J., Chang M. W., Lee K., Toutanova K. (2019) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, Proceedings of NAACL-HLT 2019 pp. 4171-4186.

[10] HuggingFace repository: *https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2*.

[11] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay É. (2011) *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research 12 pp. 2825-2830.

[12] Hagberg A., Schult D., Swart P. (2008) *Exploring Network Structure, Dynamics, and Function using NetworkX*, Proceedings of the 7th Python in Science Conference (SciPy 2008) pp. 11-15.

[13] Kleinberg J. M. (1999) *Authoritative Sources in a Hyperlinked Environment*, Journal of the ACM (JACM) 46(5) pp. 604-632.

[14] Mihalcea R., Tarau P. (2004) *TextRank: Bringing Order into Texts*, Proceedings of EMNLP 2004 pp. 404-411.

[15] Pennington J., Socher R., Manning C. D. (2014) *GloVe: Global Vectors for Word Representation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) pp. 1532-1543.

[16] Page L., Brin S., Motwani R., Winograd T. (1999) *The PageRank Citation Ranking: Bringing Order to the Web*, Stanford InfoLab Technical Report.

[17] Chen T., Guestrin C. (2016) *XGBoost: A Scalable Tree Boosting System*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, pp. 785-794.

[18] Miller D. (2019) *Leveraging BERT for Extractive Text Summarization on Lectures*.

[19] GitHub repository: *https://github.com/dmmiller612/bert-extractive-summarizer*.

[20] HuggingFace repository: *https://huggingface.co/distilbert/distilbert-base-uncased*.