Newspaper Text Analysis Coding Challenge 2024

Ananya Ananya[1]

[1]*Indian Institute of Technology, Bhilai, India*

Abstract

This report presents an in-depth analysis of newspaper text data from a coding challenge. It covers three main tasks: 1) extracting articles and headlines from annotated newspaper images using optical character recognition, 2) generating semantic embeddings of the extracted text and analyzing sentiment, and 3) applying dimensionality reduction and clustering techniques to explore the underlying topics and structure of the articles. The report provides a detailed methodology, results, and insights gained from each task, highlighting the effectiveness of techniques like UMAP, HDBSCAN, and BERTopic modeling. The findings demonstrate the potential of combining OCR, natural language processing, and unsupervised learning methods for extracting valuable information from unstructured data sources like newspaper archives.

Keywords

Pytesseract, Sentence-BERT, HDBSCAN, UMAP, t-SNE, layout Parser, K-means

1. Introduction

Newspapers have long been a rich source of information, capturing the events, opinions, and narratives of their time. However, much of this content exists in the form of printed archives, making it challenging to access and analyze at scale. This report aims to address this challenge by applying cutting-edge techniques in computer vision, natural language processing (NLP), and unsupervised learning to a dataset of newspaper images. The analysis is divided into three main tasks. The first task focuses on extracting the textual content, including articles and headlines, from annotated newspaper images using optical character recognition (OCR) and layout parsing techniques. The second task involves generating semantic embeddings of the extracted text using state-of-the-art language models like Sentence-BERT and analyzing sentiment characteristics like polarity and subjectivity. The third task explores dimensionality reduction and clustering methods, such as UMAP and HDBSCAN, to uncover the underlying topics and structure within the articles. By combining these techniques, the report demonstrates the potential of leveraging OCR, NLP, and unsupervised learning to unlock valuable insights from unstructured data sources like newspaper archives. The findings have implications for various applications, including information retrieval, content analysis, and digital archiving of historical documents.

2. Challenge 1

Given : Images of newspaper, `layout.json` - A JSON file consisting of multiple attributes.
Task: To extract the `articles` and their corresponding `headlines`, `id`, `image_id` and `anno_id`.

2.1. Solution

2.1.1. Exploratory Data Ananlysis

Firstly, I checked for the shape and the validity of polygon. All the polygons were checked .

1. **Counting Polygons by Vertices**:

ananyah@iitbhilai.ac.in (A. Ananya)
0009-0002-2431-2511 (A. Ananya)

- The code initializes counters for polygons with 1 to 8 vertices.
- It processes each annotation in the dataset to determine the number of vertices in each polygon.
- If the number of vertices falls within the range of 1 to 8, it increments the corresponding counter.
- This step is important because understanding the distribution of polygons by their vertices helps in analyzing the complexity and structure of the annotated shapes, which is essential for accurate text extraction.

2. **Checking for Invalid Polygons**:
   - The code verifies that all coordinates of the polygons are non-negative, ensuring that the polygons are within the valid coordinate space.
   - It checks if the coordinates form a valid rectangle by comparing the x and y coordinates of the vertices.
   - If a polygon is invalid, it records the annotation ID, reason for invalidity, segmentation coordinates, image ID, and category name.
   - This validation is crucial because invalid polygons can lead to errors in text extraction. Ensuring that all polygons are valid rectangles helps maintain the integrity and accuracy of the extracted text data.

These steps are important for our task of text extraction because they ensure that the input data is correctly structured and valid, which is a prerequisite for reliable and accurate text extraction from annotated images. Moreover, they also give a high-level overview of the shapes of the regions where we are applying OCR, and check if these shapes are within the range of the whole image provided and the bounding box of those polygons doesn't go outside the image.

In total we had **436** rectangle-shaped polygons from the 10 images and 2 polygons came invalid with the id **68** and **166** and both belonged to the category of Newspaper header, hence they were not of much significance to us as we are currently focusing on articles and their corresponding headlines.

We also used functions from the `Layout Parser` library, which help in organizing and visualizing blocks of various categories within images of newspaper. Additionally, we utilized the `CustomCOCO` class to map category IDs to their respective names, similar to the functionality provided by the pycocotools library. The code processes `layout.json` containing annotations of various elements within newspaper images, such as headlines, articles, and advertisements. It reads the dataset, extracts annotations, and maps them to their respective categories. For each image, it loads the image file, filters the relevant annotations, and creates a layout of text blocks. These text blocks are then visualized on the image with color-coded boxes and labels indicating their type and ID. The images are displayed in a grid format using `Matplotlib`, allowing us to visually inspect the annotated regions.

This analysis is crucial for our task of text extraction for articles and their corresponding headlines because it provides a clear visual representation of the annotated regions within each image. By analyzing the output, we can verify the accuracy and completeness of the annotations, ensuring that all relevant text blocks are correctly identified and categorized. This step is essential for developing reliable OCR models, as it helps us understand the structure and layout of the text within the images, ultimately improving the quality and accuracy of the extracted text data. Figure 1 is an example for reference.

2.1.2. Solution's Methodology

The code is designed to process a dataset of newspaper images, extracting and organizing text from these images into structured articles. It uses OCR via the `pytesseract` library, along with image preprocessing techniques and layout information to achieve this. Here's a breakdown of its functionality and the cases it handles:

Figure 1: Annotation using Layout Parser library

1. **Loading Layout Information**: The code starts by loading layout information from `layout.json`. This layout information includes annotations that describe various elements in the images, such as articles and headlines, along with their bounding boxes (bbox) and reading orders.

2. **Preprocessing Images**: Before performing OCR, images are preprocessed to enhance text readability. This involves converting images to grayscale, applying binary thresholding to create a clear distinction between text and background, and then converting back to a format suitable for OCR.

3. **Performing OCR**: For each annotated element (e.g., article text or headline) within an image, the code crops the image to the specified bounding box, preprocesses the cropped image, and then uses `pytesseract` to extract text. It handles cases where bounding box coordinates might be invalid for cropping, skipping these annotations and logging an error message.

4. **Parsing Articles**: The code identifies starting points for articles, which could be headlines or the first text block without a predecessor in the reading order. It then follows the reading order to concatenate text blocks into complete articles. This process involves handling both article text and headlines, ensuring that all parts of an article are correctly assembled. The reading order is crucial for maintaining the logical flow of text as it appears in the newspaper.

5. **Processing Newspapers**: The main function processes all newspaper images in a specified directory, applying the parsing logic to each image based on its annotations and reading orders. It compiles all extracted articles into a list.

6. **Organizing Extracted Articles**: Finally, the extracted articles are organized into a pandas DataFrame for easy analysis and manipulation. Each article is assigned a unique ID, and the DataFrame is structured with columns for these IDs, headlines, article texts, image IDs, and annotation IDs.

The code is designed to handle several specific cases:

- **Invalid Bounding Boxes**: It checks for bounding boxes with invalid coordinates and skips these annotations, preventing errors during image cropping.
- **Image Preprocessing**: It applies image preprocessing techniques to improve OCR accuracy, addressing issues like variable lighting and text-background contrast.
- **Reading Order**: It respects the reading order specified in the layout information, ensuring that text blocks are concatenated in the correct sequence to maintain the logical flow of articles.
- **Multiple Text Blocks**: It handles articles and headlines composed of multiple text blocks, assembling them into coherent articles and headlines respectively.
- **Handling Images and Other Categories**: We made sure to deal with situations where an image or a different type of content appears in the middle of the text for headlines or articles. This helps keep the article, headline and headline-article flow intact.
- **Headlines Without Reading orders**:We noticed that some articles were missing their headlines. This happened because the reading order for them was either missing or incorrect in a few cases.

This approach is essential for extracting structured text data from unstructured newspaper images, facilitating further analysis, archiving, or digitalization efforts. We extracted in total **119** articles. Table 1 shows the articles with no headlines.

id	Head-line	Text	Image ID	Anno IDs
28	-	WEST LAKE — ‘Taxes totaling 22% mills were ...	2	[91]
29	-	traffic light at the intersection of Sulphur ...	2	[92]
31	-	ASSVUCIaGICd ress ocience writer | SAN FRAN...	3	[135]
33	-	A crusade to turn out a victory report for th...	3	[139]
35	-	SPRINGHILL, N.S. (AP) — ¢ Violent shift of ro...	3	[148]
36	-	MMS RU oat Che OUMp the miners’ name for unde...	3	[152]
98	-	WASHINGTON — The -natior today has its first...	9	[384]

Table 1
Extracted Articles from Newspaper Images with missing headlines

In our analysis, we verified the hypothesis that headlines for articles were not extracted due to missing reading orders. To do this, we focused on a specific image (with image ID 2, 3, 9) from our dataset. We first filtered the annotations to include only those related to this image. These annotations were then sorted based on their position within the image to maintain a logical reading order.

For each annotation identified as a headline (category ID 3), we extracted the text using OCR techniques. We then checked the reading orders associated with each headline to see if they were correctly linked to the corresponding articles. This involved examining the relationships between annotation IDs to determine if the headline was properly connected to the subsequent text blocks.

By printing out the annotation details, bounding boxes, extracted headline text, and their reading orders, we were able to confirm that some headlines were indeed missing or incorrectly linked due to

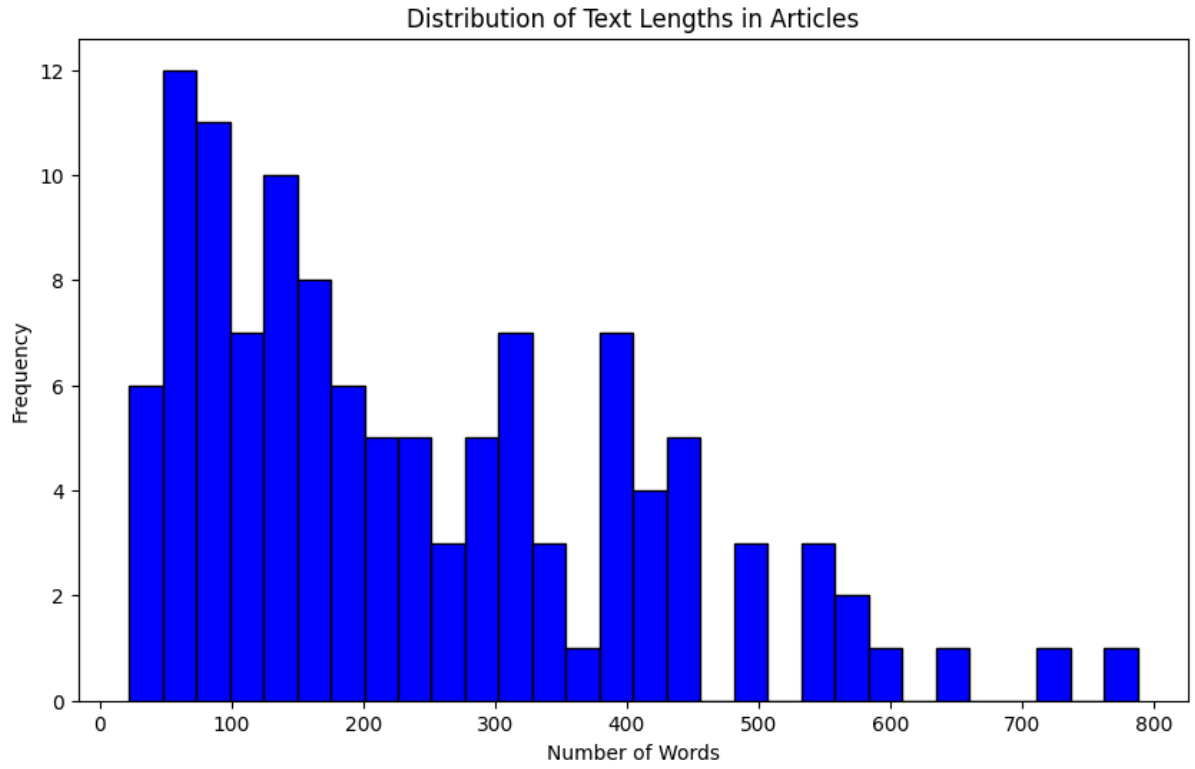


Figure 2: Length of text articles

absent or erroneous reading orders. This detailed examination helped us understand the gaps in the data and the reasons behind the missing headlines in the extracted articles.

In the final step of our process, we manually added the missing headlines for certain articles and corrected the text by concatenating parts of articles that were split across different rows in the DataFrame. This involved updating specific rows with new headline information and merging text from related rows where necessary. After making these updates, we removed the redundant rows that had been concatenated, reset the index of the DataFrame, and reassigned unique IDs to each article. This meticulous adjustment ensured that all articles were complete and correctly labeled, resulting in a total of **117** well-structured articles in our dataset.

3. Challenge 2

We first do Exploratory Data Analysis for the dataset extracted from Challenge 2. Table 2 shows that average length of article of 239 words. Figure 2 shows the distribution of length of articles.

Statistic	**Length**
Maximum length of text articles	788
Minimum length of text articles	22
Average length of text articles	239.77

Table 2
Statistics of Text Article Lengths

This challenge involves creating embeddings for articles using Sentence-BERT (SBERT). We generate embeddings for four types of text: i) summary, ii) cleaned text, iii) cleaned text without stopwords, and iv) headlines. The text extracted via OCR contains some noise.

For the cleaned text, we perform the following preprocessing steps: convert to lowercase, remove punctuation, newlines, tabs, white spaces, numbers, and apply lemmatization and stemming.

For the cleaned text without stopwords, we additionally remove stopwords.

To generate summaries, we use the following summarization pipeline: `pipeline("summarization", model="sshleifer/distilbart-cnn-12-6")`

We use SBERT with the model `SentenceTransformer("all-MiniLM-L6-v2")` to generate embeddings. The shape of the embeddings for each category is shown in Table 3

Column	Embedding Shape
summary	(117, 384)
headline	(117, 384)
cleaned_text	(117, 384)
cleaned_text_nostopwords	(117, 384)
text	(117, 384)

Table 3
Embedding Shapes for Different Text Types

We calculate the semantic similarity between the original text and the various processed texts to evaluate how much semantic information is preserved or lost during cleaning. This also helps in understanding the impact of removing non-textual elements, which makes the text cleaner and easier to process. Table 4 shows the average similarities between the original text and the different categories:

Comparison	Average Similarity
text and summary	0.7277275
text and headline	0.46785903
text and cleaned_text	0.8646999
text and cleaned_text_nostopwords	0.7410391

Table 4
Average Similarity Between Text and Different Categories

We can see that most semantic information is preserved by the cleaned text. There is more semantic alignment between text and cleaned text.
However, we generated embeddings for each type and stored them in the DataFrame.

How do you deal with the noise (e.g., spelling mistakes, punctuation errors) in the OCRed text? Is this noise a problem for SBERT? SBERT is specifically designed to generate fixed-size sentence embeddings, which makes it more suitable for tasks like semantic textual similarity, clustering, and information retrieval. Traditional BERT embeddings are token-level and require additional pooling strategies to obtain sentence-level representations. SBERT uses a Siamese network structure to directly optimize for sentence similarity tasks. This structure allows SBERT to compute sentence embeddings once and then compare them using cosine similarity, which is computationally efficient.SBERT has been shown to outperform BERT on various sentence-level tasks due to its fine-tuning on sentence pairs. This fine-tuning helps SBERT capture the semantic relationships between sentences more effectively than BERT, which is pre-trained on token-level tasks. Sentence-BERT embeddings can be particularly useful for handling noisy text, such as text with spelling mistakes and grammatical errors.

1. **Robustness to Noise:** SBERT embeddings are designed to capture the semantic meaning of sentences, which makes them inherently more robust to noise compared to traditional word

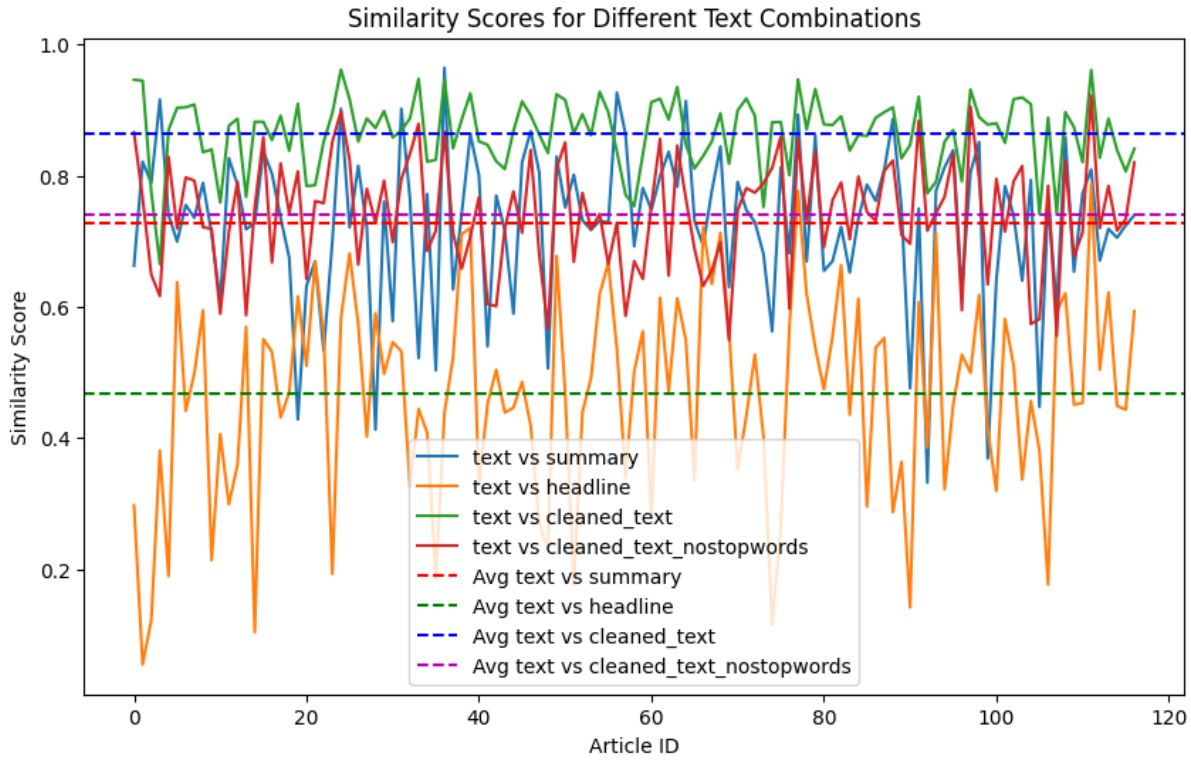


Figure 3: Similarity Scores article-wise for different combinations

embeddings. By focusing on the overall sentence context, SBERT can mitigate the impact of individual word errors, such as spelling mistakes and grammatical errors, on the final sentence representation.

2. **Fine-Tuning on Noisy Data:** Fine-tuning SBERT on noisy datasets can improve its performance in handling text with errors. For example, fine-tuning on OCR'd texts has been shown to improve the performance of BERT-based classifiers by approximately 2%, suggesting that SBERT can similarly benefit from exposure to noisy data during training. This process helps SBERT learn to manage and correct for common errors during the embedding process.

3. **Contextual Understanding:** SBERT leverages the contextual understanding capabilities of BERT, which allows it to generate embeddings that consider the context of each word within a sentence. This contextualization helps SBERT to better handle real-word errors and homophones, as it can use surrounding words to infer the correct meaning even when individual words are misspelled or grammatically incorrect.

4. **Manifold Learning:** Refined SBERT, which utilizes manifold learning, can further enhance the robustness of sentence embeddings by mapping sentences to a refined semantic space. This approach helps in discovering the local geometric structure of sentences, making the embeddings more resilient to noise and better at capturing the essential semantic structure of the text.

5. **Denoising Autoencoders** SBERT can be combined with denoising autoencoders, such as the Transformer-based Sequential Denoising Auto-Encoder (TSDAE), to improve its ability to handle noisy text. TSDAE helps in learning robust sentence embeddings by training the model to reconstruct clean sentences from noisy inputs, thereby enhancing the model's resilience to various types of noise, including spelling mistakes and grammatical errors.

SBERT embeddings are well-suited for handling noisy text due to their focus on sentence-level context, ability to be fine-tuned on noisy data, and advanced techniques like manifold learning and

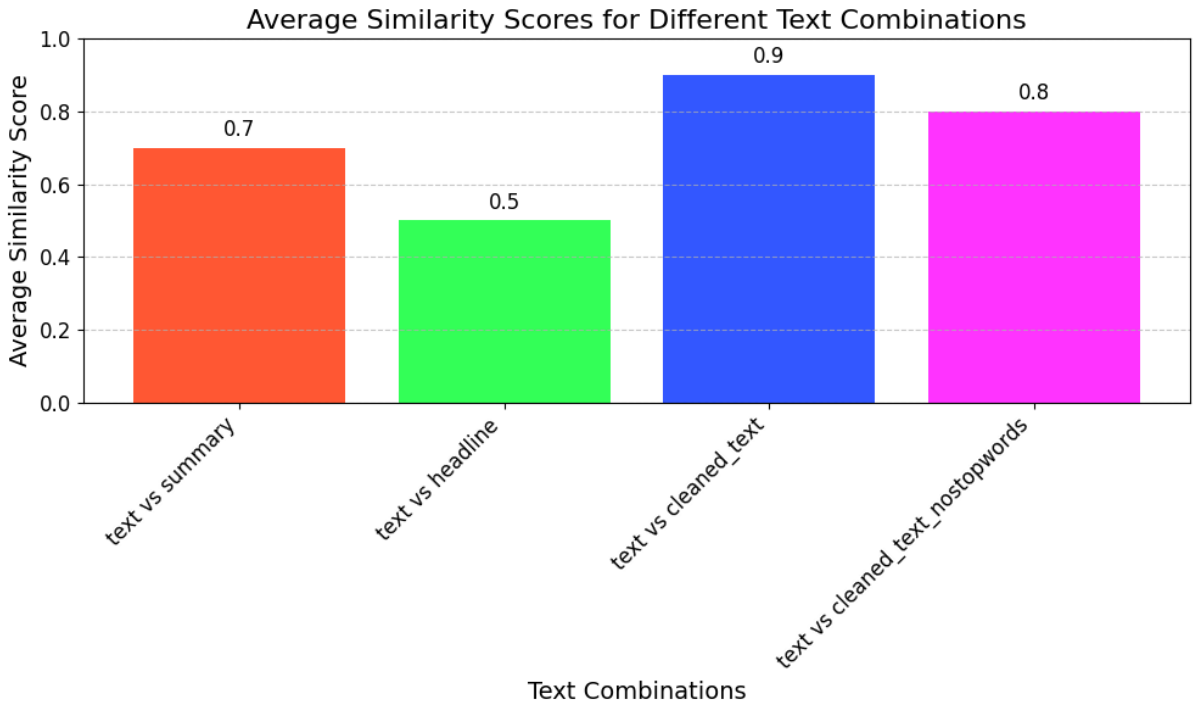


Figure 4: Comparison of average similarity scores

denoising autoencoders. These features make SBERT a powerful tool for generating robust sentence embeddings even in the presence of spelling mistakes and grammatical errors.

How do you handle embedding very long articles? The embeddings generated by SBERT (Sentence-BERT) have a fixed length, regardless of the length of the input text. This is a key feature of the model, which allows it to produce embeddings of a consistent size for any input text, whether it's a short headline or a long article. **Fixed-Length Embeddings**

1. **Model Architecture**:
    - SBERT uses a transformer-based architecture, specifically a variant of BERT (Bidirectional Encoder Representations from Transformers). The model is designed to produce fixed-size embeddings for any input text.
    - The model's architecture includes a final dense layer that maps the output to a fixed-size vector. For the model `SentenceTransformer("all-MiniLM-L6-v2")`, this fixed size is 384 dimensions.
2. **Embedding Process**:
    - When we input a text (regardless of its length), the text is tokenized into smaller units (tokens).
    - These tokens are then passed through the transformer layers of the model.
    - The output of the transformer layers is a sequence of vectors, one for each token.
    - The model then applies a pooling operation (e.g., mean pooling, max pooling, or using the `[CLS]` token) to combine these token vectors into a single fixed-size vector. This vector is the embedding of the text.
3. **Consistency Across Text Lengths**:

- Because of the pooling operation, the length of the input text does not affect the size of the output embedding. Whether the input is a short headline or a long article, the resulting embedding will always have the same fixed size (384 dimensions in this case).

Why 384 Dimensions?

- The specific model `SentenceTransformer("all-MiniLM-L6-v2")` is designed to produce 384-dimensional embeddings. This is a design choice of the model is to balance between embedding size and performance.
- The fixed size of 384 dimensions is chosen to provide a good trade-off between capturing enough semantic information and keeping the embeddings compact for efficient storage and computation.

Summary

- The embeddings have a fixed length of 384 dimensions, regardless of the input text length.
- This fixed length is a feature of the SBERT model architecture, which uses a pooling operation to produce a consistent-size vector.
- The fixed-size embeddings allow for efficient comparison and storage of text representations, making them suitable for various natural language processing tasks.

Provide a brief discussion of the tradeoffs involved in generating embeddings for headlines
From Figure 3 and Figure 4, we observe that there is the least semantic alignment/similarity between headlines and their corresponding text compared to other types. Headlines are often very short and lack sufficient context. Additionally, headlines can sometimes be misleading due to their phrasing. For example, "Police chased the thief with a gun" could imply either that the police had a gun or the thief had a gun. Due to the lack of complete context, relying solely on headlines can lead to incorrect interpretations.

Headlines are shorter which can limit the amount of information captured in the embeddings. This brevity can lead to embeddings that are less representative of the full content. They can be ambiguous. This can introduce noise and reduce the accuracy of the embeddings. Headlines are suitable for quick summarization tasks or when computational resources are limited. However, they may not be ideal for tasks requiring deep semantic understanding.

In case we used classical embeddings like **tf-idf**, we might have to take into account the length of articles, grammatical and spelling errors, etc.

3.1. Sentiment Analysis

Subjectivity Analysis Subjectivity measures the degree of personal opinion or emotion in the text, with values ranging from 0 (very objective) to 1 (very subjective).

- **General Trend**: The subjectivity of the text content (all articles in one newspaper) (orange bars) is generally higher than that of the headlines (all headlines in one newspaper) (blue bars). This suggests that the full text of the articles in newspapers tends to contain more personal opinions or emotions compared to the headlines.
- **Variation in Subjectivity**: All the articles are more subjective, with subjectivity values ranging from 0.31 to 0.47. There is not much variation in subjectivity across different articles. Headlines are more objective.

Analysis and Inference: The higher subjectivity in the full text compared to headlines indicates that while headlines aim to be catchy and attention-grabbing, the articles themselves delve deeper into personal opinions and emotions. This could be due to the nature of journalistic writing, where the body of the article provides detailed analysis and viewpoints, whereas headlines are designed to be concise, impactful and attention-grabbing.

Polarity Analysis Polarity measures the overall sentiment of the text, ranging from -1 (very negative) to 1 (very positive). Here are the key observations:

- **Variation Across Image IDs**: The polarity values for both headlines and text vary significantly across different image IDs (newspapers). This indicates that the sentiment of the images (newspapers) is not consistent and can change considerably from one article to another.
- **Headlines vs. Text**: Generally, the polarity values for article text (all articles in one newspaper) are higher than those for the headlines (all headlines in one newspapers). This suggests that the text tends to be more positive. Often, headlines are more negative or controversial sounding to make them catchy and grab the readers' attention. For example, Image ID 0 shows a high positive polarity for both the headline and the text.
- **Negative Polarity**: Some image IDs (newspapers), such as 5, 8, and 9, show negative polarity values for the headline.

Analysis and Inference: The patterns in polarity and subjectivity suggest that headlines are crafted to be more engaging and emotionally charged, possibly to attract readers' attention, while the actual article content tends to be more balanced, positive, and subjective. This strategy might be employed by journalists to draw readers in with provocative headlines, while the articles themselves provide a more nuanced and detailed perspective. The significant variation in polarity across different image IDs highlights the diverse nature of news content, reflecting a wide range of sentiments and topics covered in the newspapers. Table 5 also justifies that article text are more positive.

The analysis of subjectivity and polarity in headlines and article text reveals distinct strategies in journalistic writing. Headlines are designed to capture attention with higher emotional charge and sometimes negative sentiment, whereas the full articles provide a more comprehensive and subjective view, often with a more positive sentiment. Figure 5 and Fig 6 also compares the Subjectivity and Polarity from which we also verify our hypothesis and draw inferences.

Sentiment	Mean values
Polarity_text	0.072353
Subjectivity_text	0.366556
Polarity_headline	0.014809
Subjectivity_headline	0.179834

Table 5
Mean Values for Subjectivity and Polarity across headlines and article text for all the 9 images (newspaper)

Polarity Analysis (Text) The polarity values for the text range from -0.2 to 0.5, with a peak around 0.1. This indicates that the overall sentiment of the text is slightly positive. The distribution shows a moderate spread, suggesting a variety of sentiments across different articles.

Subjectivity Analysis (Text) The subjectivity values for the text range from 0.0 to 0.8, with peaks around 0.3 and 0.4. This suggests that the text contains a moderate level of personal opinion or emotion. The distribution indicates that most articles have a balanced mix of subjective and objective content.

Polarity Analysis (Headlines) The polarity values for the headlines range from -1.0 to 0.75, with a peak around 0.0. This suggests that headlines are generally neutral in sentiment, with some variation towards both positive and negative sentiments. The concentration around zero indicates that headlines are crafted to be attention-grabbing without being overly positive or negative.

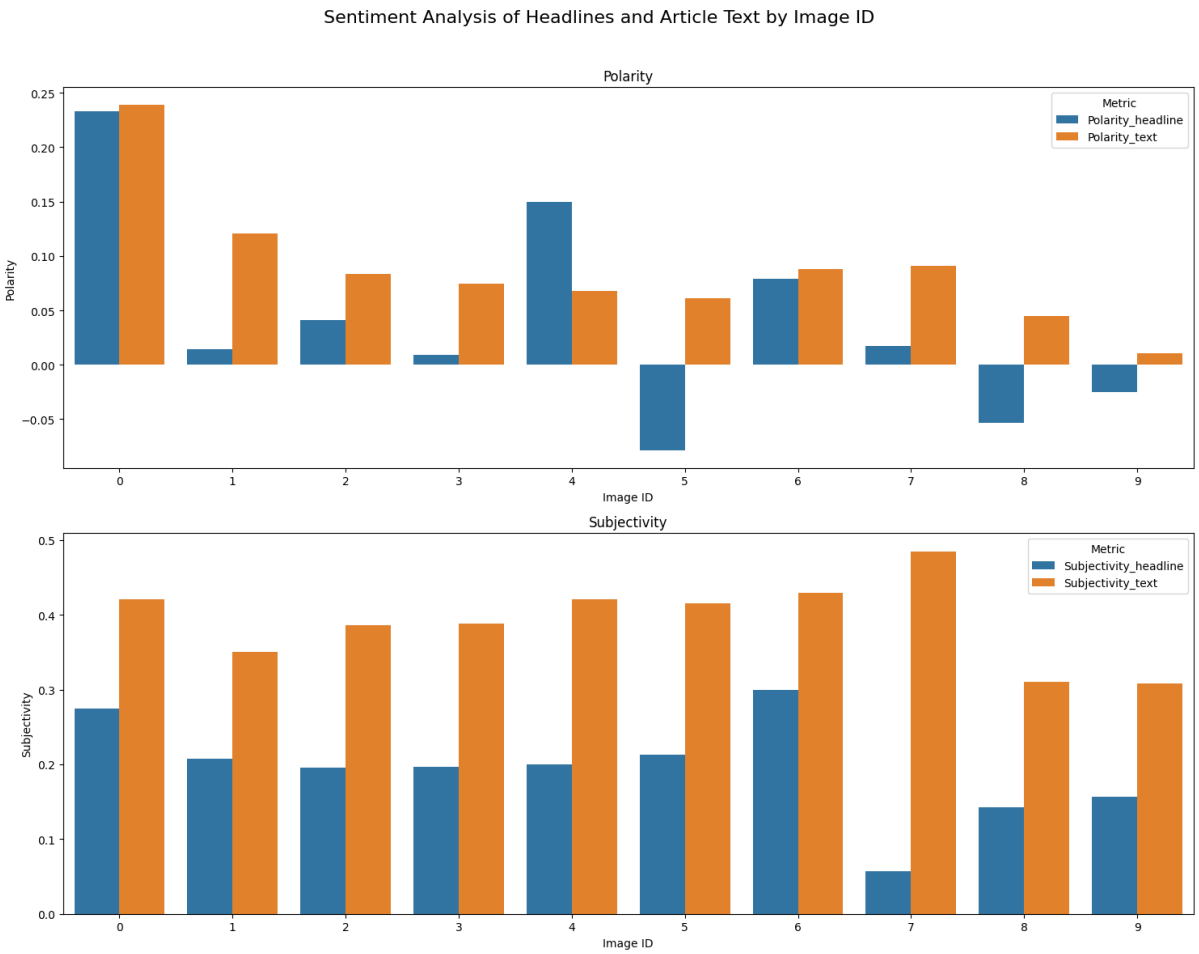


Figure 5: Polarity and Subjectivity for text and headlines, highlighting differences in sentiment and subjectivity peaks

Subjectivity Analysis (Headlines) The subjectivity values for the headlines range from 0.0 to 1.0, with a significant peak at 0.0. This indicates that most headlines are very objective. However, there is a secondary peak around 0.8, suggesting that some headlines are highly subjective, likely to attract readers' attention.

Inference From this analysis, we can infer the following:

- **Sentiment Consistency**: Both the text and headlines exhibit a generally neutral to slightly positive sentiment, with text showing a broader range of sentiments.
- **Subjectivity Variation**: The text tends to have a moderate level of subjectivity, while the headlines show a bimodal distribution with peaks at 0.0 and 0.8. This indicates that while many headlines are objective, a significant number are highly subjective.
- **Emotional Charge in Headlines**: The higher subjectivity in some headlines suggests that they are crafted to be more engaging and emotionally charged, possibly to attract readers' attention. In contrast, the text provides a more balanced and detailed perspective.

This pattern is typical in news or media content, where headlines are designed to capture attention, and the full articles provide more comprehensive information.

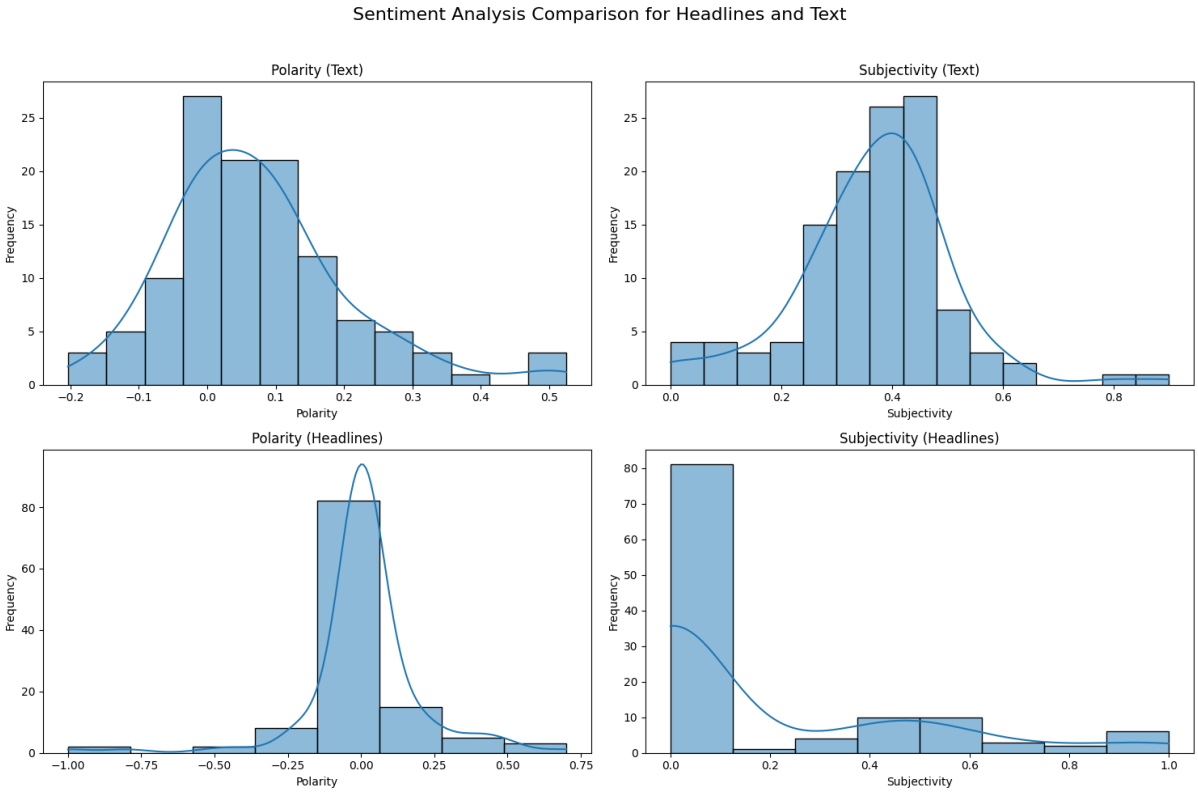


Figure 6: Comparison of polarity and subjectivity metrics for headlines and article text across different image IDs (newspaper)

4. Challenge 3

4.1. Dimensionality Reduction

Dimensionality reduction is the process of transforming data from a high-dimensional space into a lower-dimensional space while retaining meaningful properties of the original data. This technique is crucial in fields dealing with large datasets, such as machine learning, signal processing, and bioinformatics, as it helps mitigate the "curse of dimensionality." The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces, often leading to increased computational complexity and data sparsity.

4.1.1. Importance of Dimensionality Reduction

Dimensionality reduction is important for several reasons:

- **Noise Reduction**: It helps in removing redundant and noisy features, improving the quality of data.
- **Data Visualization**: It enables the visualization of high-dimensional data in 2D or 3D, making it easier to understand and interpret.
- **Computational Efficiency**: Reducing the number of dimensions decreases the computational cost and time required for data processing and model training.
- **Overfitting Prevention**: By reducing the complexity of the data, it helps in preventing overfitting in machine learning models.

4.1.2. Techniques

Principal Component Analysis (PCA) PCA is a linear dimensionality reduction technique that transforms the data into a new coordinate system where the greatest variances by any projection of the data come to lie on the first coordinates (called principal components). It is widely used for data compression and visualization.

t-Distributed Stochastic Neighbor Embedding (t-SNE) t-SNE is a nonlinear dimensionality reduction technique particularly well-suited for embedding high-dimensional data into a space of two or three dimensions, which can then be visualized. It is commonly used for visualizing clusters in high-dimensional data.

Matrix Decomposition Matrix decomposition techniques, such as Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF), decompose a matrix into product matrices to reduce dimensionality. These techniques are used in applications like image compression and topic modeling.

Uniform Manifold Approximation and Projection (UMAP) UMAP is a nonlinear dimensionality reduction technique that is used for visualizing high-dimensional data. It is known for preserving more of the global structure of the data compared to t-SNE and is faster to compute.

4.1.3. Summary Table

Technique	Definition	Best Use Case	Advantages	Disadvantages
PCA	Linear transformation to new coordinate system with maximum variance	Data compression, visualization	Reduces dimensionality while preserving variance	Assumes linear relationships, may lose information
t-SNE	Nonlinear embedding for high-dimensional data visualization	Visualizing clusters in high-dimensional data	Captures complex relationships, good for visualization	Computationally intensive, not suitable for large datasets
Matrix Decomposition	Decomposes a matrix into product matrices	Image compression, topic modeling	Reduces dimensionality, interpretable components	May require large computational resources
UMAP	Nonlinear technique for high-dimensional data visualization	Visualizing high-dimensional data	Preserves global structure, faster than t-SNE	May require parameter tuning, less interpretable

Table 6
Summary of Dimensionality Reduction Techniques

Figure 7 analyzes the variance explained by different numbers of principal components for text embeddings stored in a DataFrame. It standardizes the embeddings, applies Truncated Singular Value Decomposition (SVD) to reduce dimensionality, and then plots the fraction of variance explained by the principal components for up to six different embedding columns. This helps in understanding how much information is retained as the number of components increases. 120 components preserve 100% information, 67 components preserve 90% information.

Stress In the context of dimensionality reduction, **"stress"** is a goodness-of-fit metric used to measure the difference between the observed similarity matrix and the estimated pairwise distance matrix in a

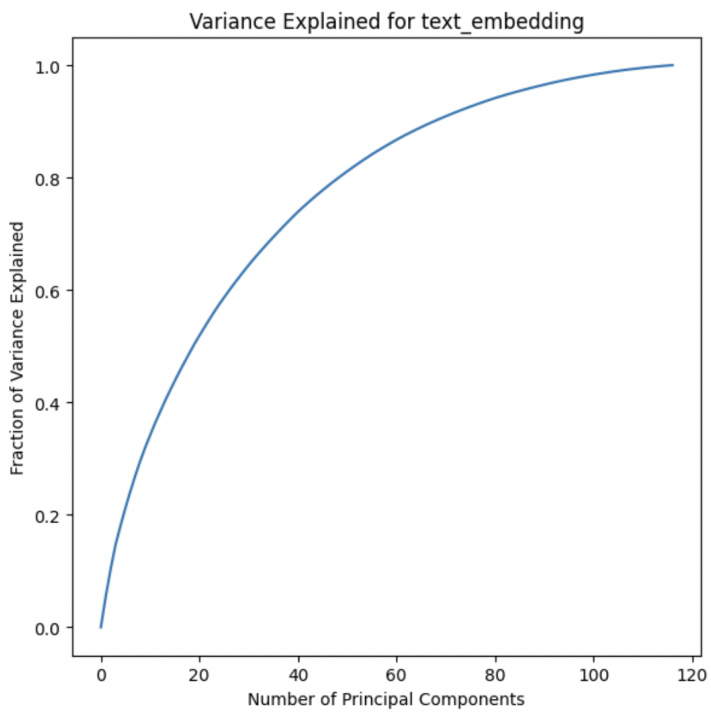


Figure 7: Variance explained v/s Principal Components

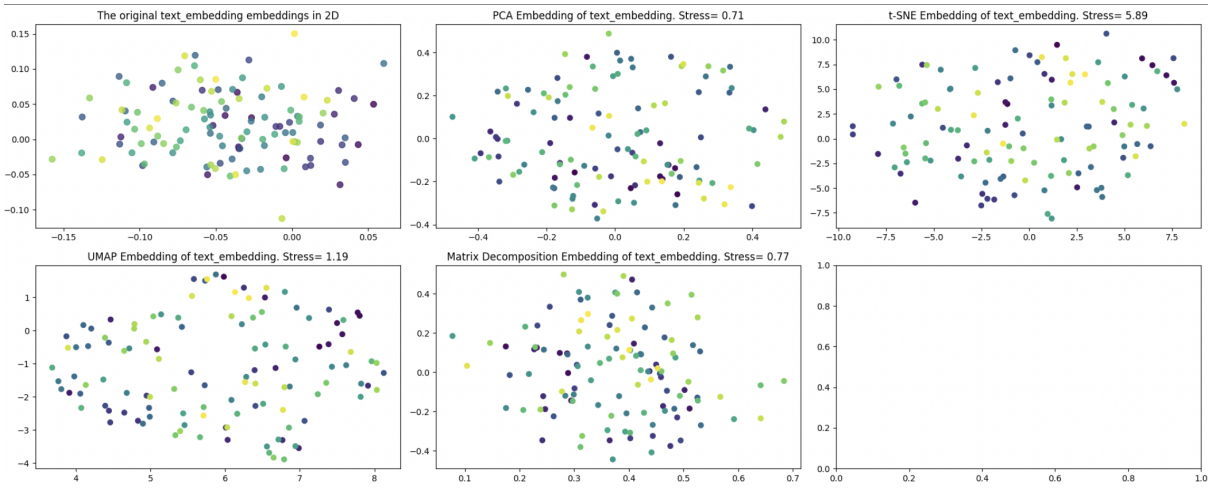


Figure 8: PCA, t-SNE, Matrix Decomposition (SVD), UMAP dimensionality reduction algorithms with respective stress values and 2-D visualisations

lower-dimensional space. It is commonly used in techniques like Multidimensional Scaling to evaluate how well the reduced-dimensional representation preserves the original distances between data points.

Higher stress values indicate a poorer fit, meaning that the low-dimensional representation does not accurately reflect the original high-dimensional distances.

Figure 8 shows `t-SNE` has the highest stress-value. Here number of components for each technique was 2.

4.2. Clustering

Clustering is a common method used to group similar items together without being told how to group them, which is especially useful when dealing with text. However, when we have a lot of features (over 500 for each piece of text), it becomes hard to measure how similar or different these items are

Figure 9: PCA, t-SNE, Matrix Decomposition (SVD), UMAP dimensionality reduction algorithms- 3-D visualisations

Technique	Components	Stress
PCA	2	0.7056
	3	0.6351
t-SNE	2	89.7238
	3	109.6772
UMAP	2	0.6312
	3	0.8479
MD	2	0.7691
	3	0.6846

Table 7
Stress Values for Different Dimensionality Reduction Techniques

because the usual ways of measuring distance don't work well with so much information. This problem is known as the "Curse of Dimensionality."

To deal with this, reducing the amount of information before clustering can help a lot. One popular method for doing this is called UMAP, which stands for Uniform Manifold Approximation and Projection. UMAP is faster and can handle more data than older methods like t-SNE, and it's good at keeping the overall shape of the data. This makes UMAP great for both making the data easier to see and understand, and for preparing the data before grouping it into clusters. We'll be using UMAP for these purposes.

Moreover, UMAP also has lower stress values than t-SNE as shown in Table 7

When using raw sentence embeddings with HDBSCAN clustering:

- When `min_cluster_size=2`, there are only 3 unique labels (clusters), and 23.08% (0.230769) of the data points are labeled as noise (-1). This suggests that most data points are assigned to a small number of clusters, potentially oversimplifying the structure of the data.
- As the `min_cluster_size` value increases, the number of unique labels (clusters) decreases further, and the percentage of data points labeled as noise increases significantly.

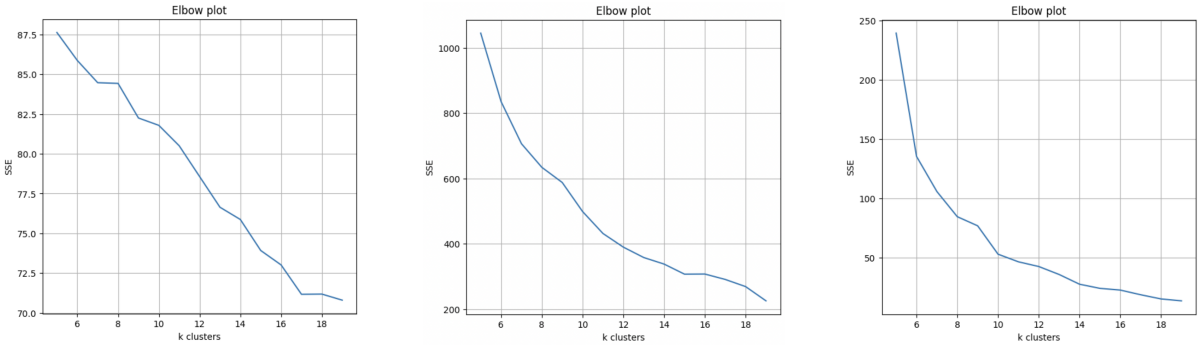
- From `min_cluster_size=5` to `min_cluster_size=10`, there is only one unique label (cluster), and 100% (1.0) of the data points are labeled as noise.

These observations indicate that raw sentence embeddings pose challenges when using HDBSCAN clustering:

- If the `min_cluster_size` is set too low, the algorithm tends to create a small number of large clusters, potentially oversimplifying the data structure.
- If the `min_cluster_size` is set too high, almost all data points are labeled as noise and not assigned to any cluster, resulting in a loss of valuable information.

In this case, when min_cluster_size=2, all data points are lumped into 3 clusters, oversimplifying the data structure. As the min_cluster_size increases, a large percentage of data points are labeled as noise and not clustered, with 100% of the data points labeled as noise from min_cluster_size=5 onwards.

Figure 10 shows that raw and t-SNE embeddings fail to find elbow point for K-Means cluster, however UMAP embedding is able to find (number of components = 2). UMAP embedding also have least SSE. Elbow method is used to find number of clusters to form in Kmeans clustering



(a) raw (b) t-SNE (c) UMAP

Figure 10: Elbow point for raw, t-SNE, UMAP embeddings (number of components = 2)

The ability to find the elbow point in the SSE plot suggests that UMAP embeddings provide a more suitable representation of the data for K-Means clustering compared to raw and t-SNE embeddings. UMAP embeddings likely preserve the cluster structure of the data better, making it easier for K-Means to identify meaningful clusters. This highlights the effectiveness of UMAP as a preprocessing step for clustering, as it can improve the performance of clustering algorithms like K-Means by providing a more informative low-dimensional representation of the data.

HDBSCAN For our current application, we need an algorithm that doesn't require us to specify the number of clusters in advance and can handle noisy data well. Density-based algorithms are a good choice because they don't need a predefined number of clusters and can work with clusters of any shape. Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) is a better option because it has fewer and more straightforward parameter settings compared to DBSCAN and can effectively manage clusters with varying densities.

K-means clustering is highly sensitive to noise and outliers. This is because K-means aims to minimize the variance within each cluster, and noisy data points or outliers can significantly distort the mean of a cluster, leading to suboptimal clustering results.

We perform hyperparameter tuning for the UMAP and HDBSCAN algorithms, which are used for dimensionality reduction and clustering, respectively. The objective is to find the optimal combination of hyperparameters that minimizes a custom loss function while ensuring that the number of clusters falls within a desired range.

The objective function is designed to balance two factors: the cost of assigning data points as noise (outliers) and a penalty for having the number of clusters outside the desired range. The cost is calculated as the ratio of data points with low probabilities (below a threshold) of belonging to any cluster, as determined by HDBSCAN. If the number of clusters is outside the specified range, a penalty of 0.15 is added to the cost.

The loss function incorporates domain knowledge by constraining the number of clusters to be between a lower and upper bound, which are set based on the expected number of labels for the given problem. This approach aims to find a clustering solution that assigns as many data points as possible to meaningful clusters while avoiding excessive noise or an unrealistic number of clusters.

The Bayesian search function uses the Hyperopt library to perform Bayesian optimization, which is an efficient method for exploring the hyperparameter space and finding the optimal configuration. The search process involves iteratively evaluating the objective function with different hyperparameter combinations and updating a probabilistic model to guide the search towards promising regions of the hyperparameter space.

Following are the Hyperparameters we tune

1. **n_neighbors**: This is a hyperparameter for UMAP, which controls the size of the local neighborhood used to approximate the manifold structure. A higher value of n_neighbors will capture more global structure, while a lower value will focus on local structure.

2. **n_components**: This is another hyperparameter for UMAP, which determines the dimensionality of the embedded space after dimensionality reduction. It specifies the number of components (dimensions) to project the data onto.

3. **min_cluster_size**: This is a hyperparameter for HDBSCAN, which sets the minimum size (number of data points) required to form a cluster. Smaller groups of points will be considered noise or outliers.

4. **random_state**: This is a seed value for the random number generator used by UMAP. Setting a fixed value ensures reproducibility of the results.

The best hyperparameters we got were: *'min_cluster_size': 4, 'n_components': 2, 'n_neighbors': 5, 'random_state': 42*

We already set `n_components` as 2 and `random_state` as 42

BertTopic Modelling BERTopic is a state-of-the-art topic modeling technique that leverages the power of transformers and c-TF-IDF to create dense clusters of topics, resulting in easily interpretable topics with important words retained in the topic descriptions. It is a versatile tool that supports guided, semi-supervised, hierarchical, and dynamic topic modeling.

You can infer from Table 8 that there are many articles having label as -1 i.e. noise/outliers.

BERTopic provides 4 different ways to reduce outliers:

1. The default method is by calculating the cTF-IDF of outliers and assigning them to the best matching cTF-IDF representations of non-outlier topics.

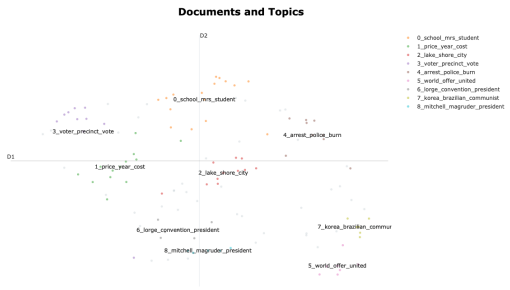
2. Using probabilities to assign topics (similar to the second method above).

3. Using topic distribution to find the most frequent topic to assign.

4. Calculating the cosine similarity between outlier embeddings and topic embeddings.

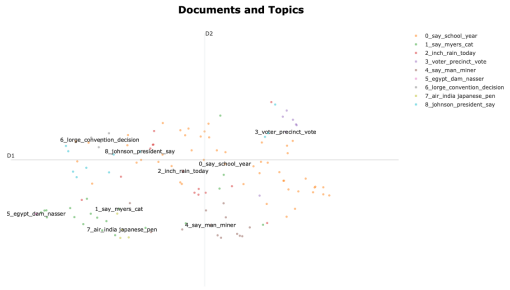
I used the 4th one. Now we have topics with 0 outliers as seen in Table 9

Meaningfulness of Clusters The clusters group together related topics. Below are some observations:

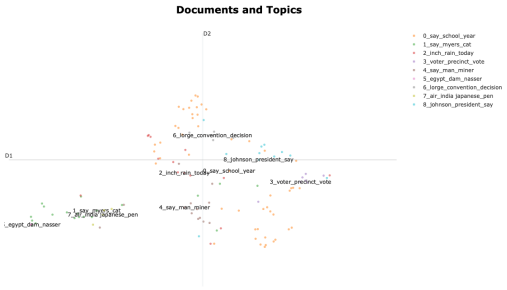


(a) HDBSCAN with outliers

(b) HDBSCAN+UMAP with outliers



(c) HDBSCAN with outliers removed



(d) HDBSCAN+UMAP with outliers removed

Figure 11: Clusters using BERTopic modeling (HDBSCAN) and (UMAP+HDBSCAN) with and without outliers

Cluster Identification

- **Cluster 0 (Orange)**: This cluster is labeled "0_say_school_year" and is spread across the plot but has a concentration in the lower right quadrant. This suggests that documents related to school years are somewhat diverse but still form a recognizable group.
- **Cluster 1 (Green)**: Labeled "1_say_myers_cat," this cluster is more concentrated in the lower left quadrant, indicating a tighter grouping of documents related to this topic.
- **Cluster 2 (Red)**: Labeled "2_inch_rain_today," this cluster is dispersed but has a noticeable presence in the upper right quadrant.
- **Cluster 3 (Purple)**: Labeled "3_voter_precinct_vote," this cluster is more towards upper right quadrant, indicating a strong grouping of documents related to voting and precincts.
- **Cluster 4 (Brown)**: Labeled "4_say_man_miner," this cluster is spread out but has a concentration in the lower right quadrant.
- **Cluster 5 (Pink)**: Labeled "5_egypt_dam_nasser," this cluster is tightly grouped in the lower left quadrant.
- **Cluster 6 (Gray)**: Labeled "6_lorge_convention_decision," this cluster is towards upper left quadrant, indicating a strong grouping of documents related to conventions and decisions.
- **Cluster 7 (moss)**: Labeled "7_air_india_japanese_pen," this cluster is more dispersed but has a presence in the lower left quadrant.
- **Cluster 8 (Cyan)**: Labeled "8_johnson_president_say," this cluster is spread out but has a concentration in the upper left quadrant.

Cluster Coherence

- The clusters seem to capture meaningful relationships between documents. For example, documents related to "voter precinct vote" (Cluster 3) are tightly grouped, indicating a strong thematic similarity.
- Similarly, documents related to "egypt dam nasser" (Cluster 5) are tightly grouped, suggesting a coherent topic.

Overlap and Dispersion

- Some clusters, like "0_say_school_year," are more dispersed (as handling outliers), indicating a broader topic that encompasses a variety of subtopics.
- Other clusters, like "5_egypt_dam_nasser," are tightly grouped, indicating a more specific and focused topic.

Clusters of Important Topics Based on the clustering, we can identify several important topics:

- **Education and School Year (Cluster 0)**

  This cluster includes documents related to school years, educational events, and related activities. The dispersion suggests a broad topic with various subtopics.

- **Voting and Precincts (Cluster 3)**

  This cluster is tightly grouped, indicating a strong thematic focus on voting, precincts, and related electoral activities.

- **Weather and Rain (Cluster 2)**

  This cluster includes documents related to weather events, specifically rain. The dispersion suggests a variety of weather-related subtopics.

- **Political Events and Figures (Cluster 8)**

  This cluster includes documents related to political events and figures, specifically President Johnson. The spread indicates a broad topic with various subtopics.

- **International Affairs (Cluster 5)**

  This cluster is tightly grouped around documents related to the Egypt dam and Nasser, indicating a focused topic on international affairs and infrastructure projects.

The above hypothesis and justifications can also be verified manually, from Table 9 and 8.

The clusters of vectors appear meaningful, capturing relationships between documents based on their topics. The clustering algorithm has successfully identified several important topics, such as education, voting, weather, political events, and international affairs. The coherence and dispersion of these clusters provide insights into the thematic structure of the documents, indicating both broad and specific topics. This can also be verified by checking for the article and its corresponding

Hierarchial Plot Figure 13 (with outlier one) displays a hierarchical clustering visualization, which groups similar data points or features together based on their proximity or dissimilarity. From the image, we can interpret the following:

1. The data points or features being clustered are related to various topics such as arrests, schools, costs, voting, cities, conventions, management, international affairs, and communism.

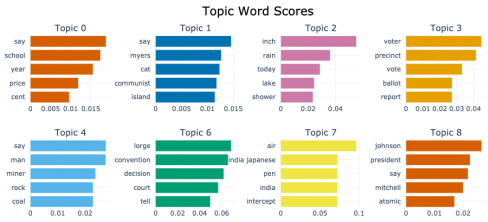
Topic	Count	Name	Representation	Representative Docs
-1	43	-1_president_inch_today_new	[president, inch, today, new, report, day, air...]	[washington phint ma jor american development ...]
0	15	0_school_mrs_student_bomb	[school, mrs, student, bomb, high, child, high...]	[successful muscallne united way cam palgn hol...]
1	12	1_price_year_cost_myers	[price, year, cost, myers, cat, bill, cent, wa...]	[cost living council say conduct thorough audi...]
2	10	2_lake_shore_city_tonight	[lake, shore, city, tonight, traffic, low, lak...]	[grant ta city del riu approve week texas traf...]
3	9	3_voter_precinct_vote_state	[voter, precinct, vote, state, report, electio...]	[generally mild weather greet texas voter tues...]
4	8	4_arrest_police_burn_cause	[arrest, police, burn, cause, smith, power, la...]	[rock island til ap rock island county states ...]
5	6	5_world_offer_united_suez	[world, offer, united, suez, egypt, acheson, d...]	[washington ap secre tary state dulle charge t...]
6	5	6_lorge_convention_president_decision	[lorge, convention, president, decision, presi...]	[washington ins labor secretary tobin urge con...]
7	5	7_korea_brazilian_communist_crash	[korea, brazilian, communist, crash, united, s...]	[belem brazil pf captive reportedly turn eap...]
8	4	8_mitchell_magruder_president_plan	[mitchell, magruder, president, plan, testify,...]	[washington ap lead py presidnet nixon nation ...]

Table 8
Extracted Topics from Newspaper Articles with outliers

Topic	Count	Name	Representation	Representative Docs
0	49	0_say_school_year_price	[say, school, year, price, cent, state, mrs, s...]	[successful muscallne united way cam palgn hol...]
1	19	1_say_myers_cat_communist	[say, myers, cat, communist, island, today, un...]	[cost living council say conduct thorough audi...]
2	13	2_inch_rain_today_lake	[inch, rain, today, lake, shower, texas, state...]	[grant ta city del riu approve week texas traf...]
3	6	3_voter_precinct_vote_ballot	[voter, precinct, vote, ballot, report, cast, ...]	[generally mild weather greet texas voter tues...]
4	12	4_say_man_miner_rock	[say, man, miner, rock, coal, kill, rescue, ex...]	[rock island til ap rock island county states ...]
5	1	5_egypt_dam_nasser_soviet	[egypt, dam, nasser, soviet, sudan, offer, can...]	[washington ap secre tary state dulle charge t...]
6	4	6_lorge_convention_decision_court	[lorge, convention, decision, court, tell, abo...]	[washington ins labor secretary tobin urge con...]
7	3	7_air_india_japanese_pen_india	[air, india, japanese, pen, india, intercept, j...]	[belem brazil pf captive reportedly turn eap...]
8	10	8_johnson_president_say_mitchell	[johnson, president, say, mitchell, atomic, na...]	[washington ap lead py presidnet nixon nation ...]

Table 9
Extracted Topics from Newspaper Articles with no outliers

(a) Topic Keywords with outlier



(b) Topic Keywords with outlier removed

Figure 12: Topic Keywords

2. The clustering algorithm has grouped these data points into several distinct clusters, with the distance or dissimilarity between the clusters shown on the x-axis. The closer the clusters are on the x-axis, the more similar they are to each other.[1]

3. Some of the major clusters formed include:

   - `1_price_year_cost` and `0_school_mrs_student` are closely clustered together, suggesting a potential relationship between these features.
   - `3_voter_precinct_vote` and `2_lake_shore_city` form another distinct cluster.
   - `6_large_convention_president`,`8_mitchell_manager_president`, `5_world_offer_united`, and `7_korea_brazilian_communist` are grouped together, indicating similarities related to leadership, international affairs, and communism.[2]

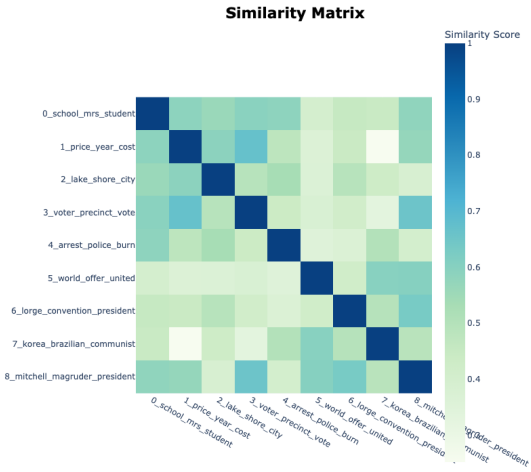
The hierarchical structure of the clustering allows for the identification of broader, higher-level clusters as well as more granular, lower-level clusters, providing insights into the relationships and similarities between the different data points or features.

[1]The x-axis represents the dissimilarity measure used in the clustering process.
[2]These clusters reflect thematic groupings based on political, economic, and social factors.

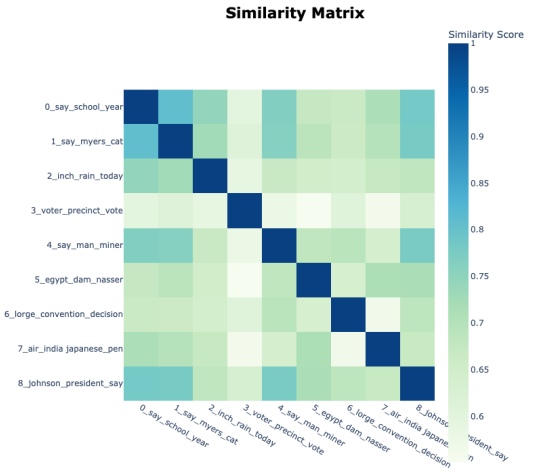
(a) Hierarchical Plot with outlier

(b) Hierarchical Plot with outlier removed

Figure 13: Hierarchical Plot



(a) Similarity Matrix with outlier



(b) Similarity Matrix with outlier removed

Figure 14: Similarity Matrix

LDA The Table 10 shows the output of Latent Dirichlet Allocation (LDA) topic modeling on the newspaper article text. LDA is an unsupervised technique that identifies topics as distributions over words in a corpus. Each row represents a different topic, with the top 10 most relevant words and their corresponding weights shown.

For example, Topic 0 appears to be related to mining, with top words like “mine”, “per”, “cent”, “increase”, and “air”. Topic 4 seems focused on political and military affairs, with words like “house”, “president”, “senate”, “atomic”, “war”, and “johnson”. Topic 3 covers education, with “school”, “precinct”, “president”, “year”, and “city” as prominent words.

By examining the top words for each topic, we can infer the underlying themes and subjects present in the newspaper articles. The weights indicate the importance and relevance of each word to that particular topic. This unsupervised approach can reveal latent patterns and themes in the text data that may not be immediately obvious. The LDA output provides a high-level semantic overview of the document collection.

5. Challenges

5.1. OCR and Image Preprocessing

One of the primary challenges encountered was the extraction of text from newspaper images using OCR. Some headlines were not being accurately extracted due to poor image quality and complex layouts. To address this, we applied various image preprocessing techniques to enhance text readability. This included converting images to grayscale, applying binary thresholding, and other methods to

Topic	Terms
0	0.003*"year" + 0.003*"mitchell" + 0.003*"mine" + 0.003*"per" + 0.002*"state" + 0.002*"increase" + 0.002*"cent" + 0.002*"air" + 0.002*"school" + 0.002*"president"
1	0.004*"state" + 0.004*"year" + 0.003*"lake" + 0.003*"school" + 0.003*"house" + 0.002*"time" + 0.002*"students" + 0.002*"county" + 0.002*"four" + 0.002*"president"
2	0.004*"mrs" + 0.004*"city" + 0.003*"precinct" + 0.003*"school" + 0.002*"reported" + 0.002*"year" + 0.002*"cost" + 0.002*"billion" + 0.002*"motor" + 0.002*"president"
3	0.007*"school" + 0.004*"precinct" + 0.004*"president" + 0.003*"year" + 0.003*"state" + 0.002*"reported" + 0.002*"way" + 0.002*"high" + 0.002*"per" + 0.002*"city"
4	0.005*"house" + 0.005*"president" + 0.003*"senate" + 0.003*"state" + 0.003*"atomic" + 0.003*"day" + 0.003*"war" + 0.002*"johnson" + 0.002*"hospital" + 0.002*"mrs"
5	0.004*"year" + 0.004*"president" + 0.003*"house" + 0.003*"city" + 0.003*"school" + 0.003*"bomb" + 0.002*"washington" + 0.002*"night" + 0.002*"state" + 0.002*"ford"
6	0.003*"state" + 0.003*"year" + 0.003*"reported" + 0.002*"president" + 0.002*"house" + 0.002*"voters" + 0.002*"officials" + 0.002*"time" + 0.002*"oil" + 0.002*"th"
7	0.003*"house" + 0.003*"year" + 0.003*"state" + 0.002*"war" + 0.002*"air" + 0.002*"president" + 0.002*"committee" + 0.002*"force" + 0.002*"per" + 0.002*"cent"
8	0.004*"state" + 0.004*"president" + 0.004*"reported" + 0.004*"school" + 0.003*"inches" + 0.003*"day" + 0.002*"mrs" + 0.002*"texas" + 0.002*"per" + 0.002*"cent"
9	0.003*"night" + 0.003*"year" + 0.003*"president" + 0.003*"state" + 0.002*"city" + 0.002*"monday" + 0.002*"time" + 0.002*"washington" + 0.002*"house" + 0.002*"west"

Table 10
Topic Modeling Output (LDA)

(a) Word Cloud with stop word removal

(b) Word Cloud with custom stop word removal

Figure 15: Word Clouds

improve the contrast between text and background, thereby improving OCR accuracy.

5.2. Missing Headlines

During the extraction process, we found that 7 articles were missing their headlines. This issue was traced back to missing or incorrect reading orders in the annotations. To resolve this, we manually corrected the headlines by examining the annotations and reading orders, and then updating the dataset accordingly. This involved adding the missing headlines and ensuring that the text blocks were correctly linked to their respective articles.

5.3. Embedding Generation and Information Loss

When generating embeddings for the text, we created embeddings for various types of text, including summaries, cleaned text, headlines, and cleaned text without stopwords. We then calculated the semantic similarity between the original text and these processed texts to evaluate the loss of information. This step was crucial to ensure that the embeddings retained as much semantic information as possible, despite the preprocessing steps.

5.4. Clustering with HDBSCAN and UMAP

Clustering the text data posed several challenges, particularly when using the HDBSCAN clustering model with BERTopic. We had to carefully select parameters such as `min_cluster_size` to balance the number of clusters and the amount of noise. Additionally, we applied UMAP for dimensionality reduction, which required tuning parameters like `n_neighbors` to minimize information loss and optimize

clustering performance. We developed a custom algorithm to find the best hyperparameters for our data, ensuring that the clusters were meaningful and representative of the underlying topics. The default hyperparameters for HDBSCAN+UMAP resulted in many points being identified as noise/outliers. We had to get the best hyperparameters, which result in less points being identified as noise.