News Article Clustering and Summarization

Saumya Didwania
sd4469@nyu.edu

Jonathan Lu
jxl219@nyu.edu

Saumyaa Shah
sns9906@nyu.edu

Center for Data Science, New York University
60 5th Ave, New York, NY 10011

1 Introduction

The internet enables information to flow easily across a wide audience. Theoretically, having all this information at our fingertips should be a good thing as it should aid us in making informed decisions. In reality however, the sheer volume of information may do more harm than good, especially when it comes to potential misinformation from conflicting news sources. The time it takes for a person to scour through and digest just a portion of this information is simply immeasurable. The Washington Post, for example, outputs on average 1200 articles per day [1] - which is almost an article every minute. Humans cannot keep up with just one news outlet, let alone the countless sources for news across the globe. Our approach for this issue creates clusters for articles about a certain topic and

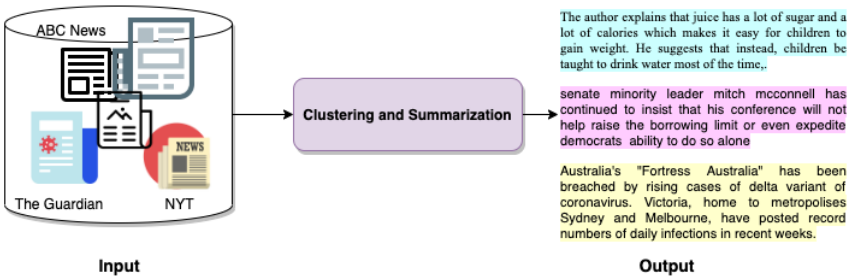


Figure 1. Process Overview

outputs coherent summaries to allow for understanding of the big news-worthy events. This solution can be used for daily newsletters, a hub aggregating articles about a certain topic, or even translating summaries into multiple languages to allow for the distribution of news globally. In this paper, we will walk through the embedding, clustering, and summarization methods we evaluated to output a comprehensible summary from a compilation of multiple news sources.

2 Related Work

Many of the papers we explored use event-based clustering around specific topics using both news articles and social media platforms. One such paper is Carta et al. [2] in which the authors use event-based detection to help predict future stock price increases. They first identify news stories related to real-world events on a daily basis and then try and see if there are certain ”hot” events for that day which is associated with a higher number of mentions in social media.

Preprint. Under review.

For summarization, we looked at an earlier paper by Facebook AI Research which builds upon a sequence-to-sequence (seq2seq) model using abstractive summarization and controlled variables (length, entities, source-style, etc.) to generate short summaries [3]. This involves usage of recurrent neural networks (RNNs) for constructing the encoder-decoder and by mapping one entity to another (hence seq2seq). While this architecture works for shorter sentences, longer sentences (and even articles) prove to be an issue as performance slumps, since RNNs struggle to memorize longer input data. Our model aims to alleviate this issue by leveraging the attention mechanism in a transformer. Transformers allows the encoder/decoder to see the entire input sequence all at once, directly modeling the dependencies between the input and output tokens using self-attention.

Facebook has plans to deploy a similar AI-based assistant tool that automatically reads news articles and summarizes them for users so that users don't have to read them [4]. However, this tool has sparked controversy in that social media platforms are flooded with both genuine and fake news. When a long article is shortened to just a short snippet, it may be difficult to understand the context, especially with misinformation flowing around.

3 Methodology

Our objective is to create an algorithm that would comb through the various sources of news articles on a regular basis and create a summary for each newsworthy event. Our input would be the body of all the articles published over a given time frame, such as a week or a day, and our output would be a legible summary that would be fit for journalists to either publish, or allow them to understand the topic enough to help ground their research on a long-form content piece.

3.1 Proposed Algorithm

The proposed algorithm consists of 4 stages: **Named Entity Recognition, Embedding, Clustering and Summarization**. Firstly, the input, a pre-processed corpus of news articles, is converted to a list of named entities per article. These named entities are then converted to word embeddings to create a vector representation of each article. The articles are then clustered into similar groups using this representation. Finally, for articles present in a cluster, we generate a news summary.

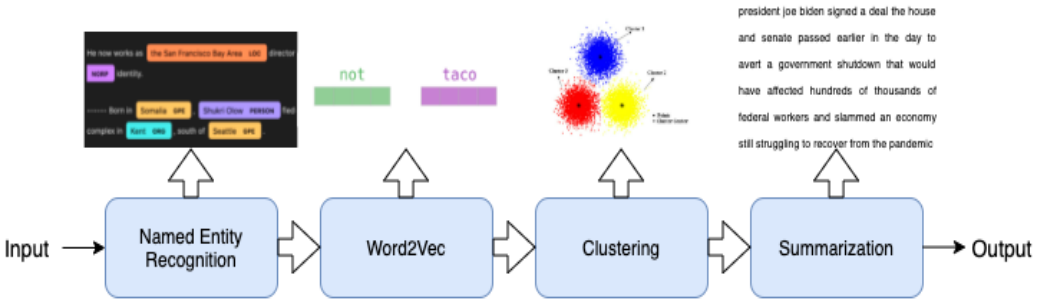


Figure 2. Proposed Approach

3.1.1 Named Entity Recognition

Named Entity Recognition(NER) is an information extraction method used to extract names, locations, date/time, etc. i.e. entities that signify an event, from unstructured text. Before generating the embeddings, we perform NER and filter out articles having less than 2 named entities. Through this, we were able to remove most of the "noisy articles" such as crossword puzzles.

3.1.2 Word2Vec

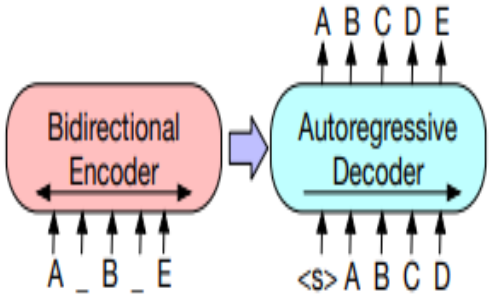
To represent the extracted named entities in vector space, we transform them into word embeddings. For this process, we train a Word2Vec [5] model on the named entities. Word2Vec is a **shallow 2-layer neural network** that learns word embeddings by predicting surrounding words. By learning the syntactic and semantic relationships between words, it embeds similar words closer in vector space.

3.1.3 k-means Clustering

We apply mini-batch k-means clustering on the embeddings to obtain clusters of similar articles. k-means clustering aims to partition the data into **k** clusters, by minimizing the Euclidean distance between points in a cluster and the cluster mean. Each point is assigned to the cluster with the closest mean/centroid. Here, the parameter k represents the number of clusters i.e. the number of summaries to be generated by the algorithm.

3.1.4 BART Summarization

Lastly, we feed all the articles in a cluster to BART (Bidirectional Auto-Regressive Transformer) to generate a concise summary per-cluster. BART is a transformer that uses a bidirectional (BERT-like) encoder and an auto-regressive (GPT-like) decoder [6]. This implies that the encoder's attention mask is fully visible, while the decoder's attention mask is causal. It is trained by corrupting text with arbitrary transformations (including changing length) and then learning a model to reconstruct the original text. By randomly shuffling the order of original sentences and replacing arbitrary length spans of text (including zero length) with a single mask token, it forces the model to reason more about overall sentence length than compared to traditional BERT. It is particularly effective when fine-tuned for text generation. Specifically, we used a BART model that has been pretrained on the CNN-Dailymail dataset consisting of 1024 hidden layers and 406M parameters [7].



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 3. BART Architecture from (Lewis et. al., 2019) [6]

4 Experiments/Results

4.1 Implementation Details

4.1.1 Data Sources

We experimented and tested on two primary datasets. The first dataset consists of 15,000 articles from September 2021, scraped using **NewsAPI**, and 70,000 articles from January 2021 - November 2021, scraped using **The Guardian API**. The second dataset consists of over 325,000 articles, sourced from the **US Newsstream dataset**, which features top newspapers, wires, broadcast transcripts, blogs and news sites in full-text format via the **Proquest** platform for the month of September.

4.1.2 Data Preparation

To prepare the data, in addition to standard pre-processing such as converting to lowercase, removing punctuation, special characters, stopwords, trailing spaces, etc. we apply **word lemmatization**

to remove prefixes and suffixes, as well as **source-specific pre-processing using regular expressions(eg. removing HTML tags, etc.)** For the above pre-processing and tokenization, we employ the **NLTK** library.

4.1.3 Algorithms

For embedding, clustering and summarization, we implement the following algorithms and compare their performance with the proposed methodology.

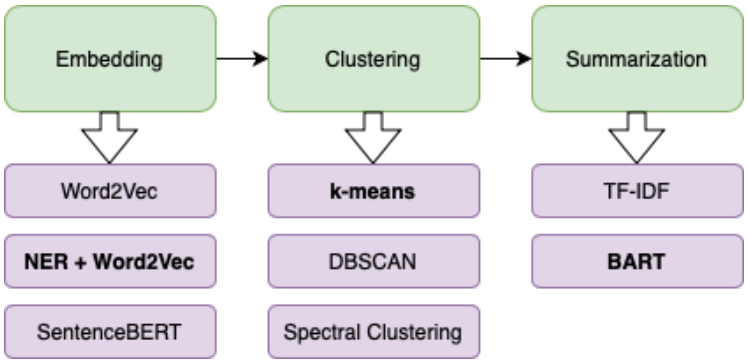


Figure 4. Algorithms Evaluated

4.1.4 Training Details

To extract and visualize named entities, we use the Entity Recognizer, implemented in **spaCy**, that performs statistical NER. To generate word embeddings, the Word2Vec model, implemented in **Gensim** is trained on named entities to generate word vectors of size **100**. For mini-batch k-means, we use **scikit-learn** and set the batch size to **500**. We tuned k (the number of clusters) to obtain the optimal value.

4.1.5 Evaluation Metrics

To evaluate the results of clustering, we employ two quantitative metrics: **inertia** and **silhouette coefficient**. The inertia, also known as the sum-of-squares within a cluster, measures the closeness of data points of a cluster.

$$\textbf{Inertia: } \sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_i||^2) \, ,$$

where C = cluster, μ_i = cluster centroid

The silhouette coefficient measures how similar a point is to points in the same cluster in comparison to data points in other clusters. Other than quantitative measures, we also evaluate the results of clustering by comparing the cluster assignments from k-means to the cluster assignments from clusters designed by an industry professional.

4.2 Experiments

4.2.1 Embedding

To generate text embeddings, we experimented with 3 methods: **Naive Word2Vec, Word2Vec+NER** and **SentenceBERT**.

Naive Word2Vec generates word embeddings for all the words in the article. SentenceBERT on the other hand builds upon BERT by using Siamese and triplet networks to generate semantically sound sentence embeddings. Figure 5 portrays the performance of different embedding methods in terms of inertia and silhouette coefficient.

1. **Inertia** We observe that for NER+Word2Vec, we get the lowest inertia, a value close to zero. This indicates that the distance between the points within a cluster is low i.e. similar

data points have been grouped together. Whereas, for the other two algorithms, the inertia value is significantly higher.

2. **Silhouette Coefficient** We observe that for NER+Word2Vec, the silhouette coefficient is comparable to Word2Vec, while having a much lower inertia. For SentenceBERT, the silhouette coefficient is extremely low, almost nearing zero - an indicator that there are many overlapping clusters.

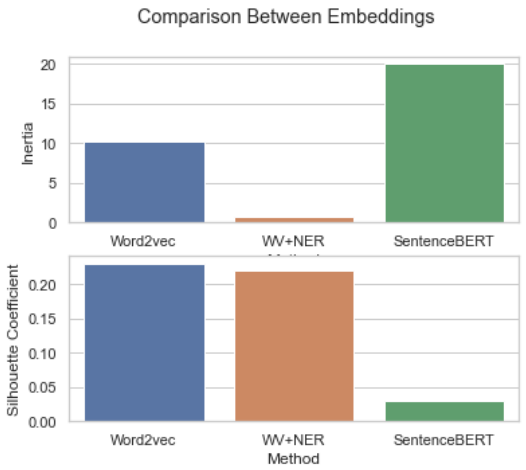


Figure 5. Performance Comparison

One of the reasons of poor performance of SentenceBERT embeddings in k-means clustering is due to the metric optimized. K-means uses Euclidean distance to measure similarity between sentences whereas different clustering techniques can utilize distance metrics such as **Levenshtein distance or cosine similarity** to lead to better performance.

4.2.2 Clustering

For clustering, we experiment with 3 methods: **DBSCAN, Spectral clustering** and **k-means**.

DBSCAN [8] i.e. Density-based Spatial Clustering algorithm identifies separable clusters in the data, such that the cluster is a contiguous region of high point density, distinguishable from other clusters by regions of low point density. We observed that for DBSCAN, most of the articles are clustered into the ”other” cluster, indicated by -1 and most of the other clusters had less than 10 articles.

Spectral clustering on the other hand performs dimensionality reduction using a similarity matrix of data before clustering the data points into groups. It performed well on the smaller datasets but as we increased the size of our input, Spectral Clustering continued to output only a few clusters. Since our goal was to provide enough of a distinction in articles for large datasets, that result would not suffice but could be beneficial in circumstances that would only have a small number of articles or limited coverage from larger news outlets.

As k-means requires a pre-specification of clusters, we ran an experiment to show how our main evaluation criteria are effected by the number of clusters. As we increased the number of our clusters, both our inertia and silhouette coefficients decreased. While a lower silhouette score indicates our clusters may have more overlap, **30 clusters** provided enough of a separation to get the newsworthy events while also keeping the clusters well-separated. Figure 6 shows the trend over the number of clusters.

Additionally, we observed a trade-off between the number of clusters and the quality of cluster summary. As we increased the number of clusters, we observe a fewer quantity of articles per cluster and hence, better summaries. But, as the dataset size increases, the number of clusters would become extremely large. Since our goal is for this algorithm to potentially output ready-to-print summaries, flooding our audience with a high number of news summaries would be the opposite of our goal. Using a smaller k for large datasets causes dissimilar articles to be clustered together. Hence, the

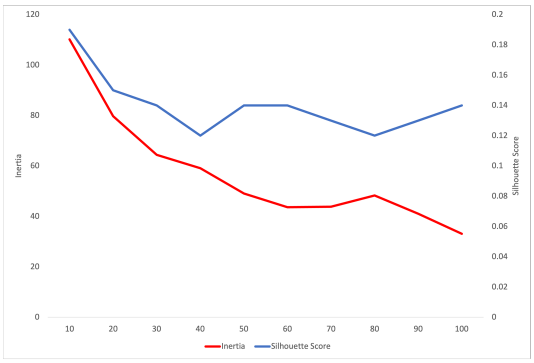


Figure 6. Performance Comparison

clustering parameter **k** needs to be tuned in such a way that the complexity of model and summary quality are balanced.

A further evaluation was done on our clustering by using t-stochastic neighbour embeddings (t-SNE) to represent all the named entities present in a cluster, as displayed in Figure 7. For the first cluster, we notice entities such as **hollywood, california, disney**. The summary for that cluster was about the Disney Resort. For the second cluster, we notice entities such as **republicans, trump, gop**. Looking at the summary for the cluster, we see that it is about a bill being passed in Congress.

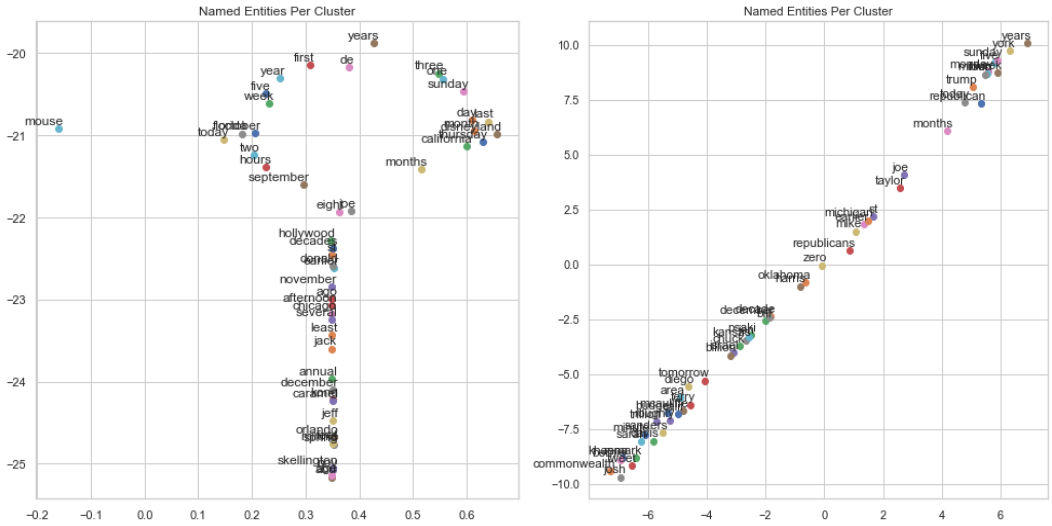


Figure 7. T-SNE

4.2.3 Summarization

For summarization, we compare two algorithms: An extractive approach via **TF-IDF** and an abstractive approach using **BART**. An extractive approach simply concatenates the most important tokens/sentences without understanding the meaning behind the sentence (i.e. looking at the 'numbers'). It extracts the best sentences from the dataset without generating new tokens. An abstractive approach aims to generate a meaningful summary from the data. When generating tokens, it does not use the same tokens from the data but rather generates new tokens based on how/which dataset the model was trained.

TF-IDF (term frequency-inverse document frequency), as shown in Equation 1 measures how common a token is in a document relative to all the documents in a corpus. The term frequency (TF) portion looks at the proportion of tokens that appear in a single document d. Document frequency looks at the proportion of the number of documents N in a corpus D that contain such token t. The TF-IDF score tells you how relevant a token is in a specific document, with higher leaning scores being more relevant and lower scores less important.

$$\text{TF}(t, d) = \frac{\text{count}(t \in d)}{\text{total \# of tokens in d}} \tag{1}$$

$$\text{IDF}(t, D) = \log\left(\frac{N}{\text{count}(t \in d : d \in D)}\right) \tag{2}$$

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{3}$$

The summary is generated by selecting the top n sentences with the highest TF-IDF scores. In the code, we also specified a threshold s for the score to ensure that only high scoring sentences i.e. sentences that appear often in other articles are selected. Hence, well-defined clusters tend to produce more sentences for summarization than a poorly defined cluster.

Compared to BART, TF-IDF yielded poor results. Even though it extracted the most important sentences, the sentences all came from different articles - yielding an incoherent summary. Extractive summarization seems ineffective when performed on a large corpus of documents.

BART on the other hand, yielded stellar results. It generated cohesive and meaningful sentences based on the dataset partially due to using a BART model pretrained on a news (CNN-Dailymail) dataset. From the summaries it generated, one could easily identify what the articles in a particular cluster were about. An example of a generated summary from cluster 5 can be found in Figure 8.

```
5:
 Gov. That leaves very limited options. Bronsons office released a statement after the meeting. That never happened.
It was a move that shocked and angered parents. Iowas Republican Gov. Four firefighters were injured while responding
to the scene. He did not elaborate on the initial investigation into those reports. They were both in incubators beca
use they were born underweight. Neither woman was identified. But everything I know is American. Some families have a
lready experienced slow-rolling family separations under the policy. Nursing homes were limiting admissions. "Weve be
en planning all along. Carrington said. ------ Hill reported from Albany. A similar condition can occur in adults. Su
bmit them at: FactCheck@AP.org. Read more here: What is the mu variant of the coronavirus? What can employers do if w
orkers avoid COVID-19 vaccines? Do I need a booster if I got the Johnson & Johnson vaccine?
```

(a) TF-IDF

```
5:
[{'summary_text': " In January, Alaska had the highest per capita coronavirus vaccination rate in the nation . Now, h
ospitals are overwhelmed with COVID-19 patients, and the states largest hospital is rationing care . Vaccine hesitanc
y and the delta variant have pushed the states fragile and limited hospital system to the breaking point . Of Alask
a's 120 ICU beds, 106 were filled as of Thursday -- leaving only 14 beds available statewide ."}]
```

(b) BART

Figure 8. Summaries generated from cluster 5 using (a) TF-IDF and (b) BART

As shown, the TF-IDF generated summary is incoherent and does not mimic human writing. While one can tell the main topic is about the coronavirus and hospitalizations (**Keywords**: coronavirus, mu variant, vaccine, Johnson & Johnson, incubators, etc.), there still remains key sentences that the algorithm deemed important but are off-topic (e.g. firefighters responding to a scene, family separation under a policy). Whereas in the BART generated summary, one could identify that the articles in this cluster fall under the category: coronavirus and hospitalizations, particularly in Alaska. Furthermore, this generated summary is well-structured and coherent as if a human wrote it. In general, abstractive methods seem to perform better in multi-document summarization than extractive methods.

5 Conclusions and Future Work

5.1 Usages in Production

First and foremost, our method aims to solve the issue of the deluge of news articles every day. In the Guardian dataset, the 70,000 articles comprised of over 70 million words. At an average reading rate of 250 words per minute, it would take over 4,700 hours to read all the articles in the dataset. Even if we take just a week's worth of articles, it would take almost 100 hours to read those. With our summarization algorithm, and our cluster size of 30, we can reduce that 100 hours to just 30 minutes per week to get the top headlines. Not only would the reader get the top events, they would also get a base understanding of what the article would be about through BART's summarization. In a newsroom environment, this procedure allows a journalist to read through the summaries and

find all the articles that were about a certain topic. This would allow newsroom staff to easily create deep-dive pieces, automatically having their data collection done and only have to focus on the thought-provoking writing.

5.2 Shortcomings

One of the biggest shortcomings of our current method is that our algorithm primarily uses Named Entity Recognition to generate the summary of the cluster of articles. One issue we observed was its bias towards named entities related to data and time. For articles related to less popular events that do not involve celebrity names or locations, most of the named entities are related to the time and day of the event, which affects the clustering results, and hence the summarization. While we attempted using event based extraction and did not receive great results, we might try and develop a combination of event based extraction with named entity recognition to create clusters that could also recognize the different sides to a story. For example, our current algorithm mostly clusters stories regarding the COVID-19 Vaccine together but in the future, we might be able to better differentiate between opposing sides viewpoints on vaccine distribution. It could be possible to originally cluster according to named entity recognition and then split those topical clusters via event extraction if the clusters are large enough. Our current solution of specifying the number of clusters via k-means would not be flexible to this addition so we might have to re-explore using DBSCAN or Spectral Clustering.

Additionally, we noticed that our current methodology generates poor summaries for clusters with some dissimilar articles. For example, if articles about two seemingly unrelated named entities are clustered together, the summary generated only talks about one of them and may even produce different summaries if the sentences are shuffled. If we plan to only use this algorithm to get the top 10 best news articles, this wouldn't impact our solution as those articles would be the most covered and, therefore, the most important events. However, for specialized articles for which there may have been very little coverage by large news outlets, those articles would get lost in a summarization of a different event.

5.3 Future Development

Some of the areas for future development would be finding alternatives to the full body of the article for summarization, algorithmically restricting the timespan of the cluster, and translation into multiple languages. While we decided to use the full body to get a detailed summarization of the article, in the future, we may find a significant reduction in time to train the model by using just the first and last paragraph - where the author likely puts the salient points of the article - or even using the headline to get the key idea. In our current procedure, we have tested the model using almost a full year, a month, and a week to generate clusters. It may be worthwhile to explore a method to choose the time horizon of the cluster, potentially creating new clusters when the latest set of articles do not closely match the previously allocated clusters. In the event that this algorithm is used for a daily summary of events, we could try and create clusters that include the news for that day as well as a few that update the previous day's summary with any new information. Finally, while we did use BART's language translation model, we did not have a method to test its successful application - something that can be explored in a future endeavor.

6 Lessons Learned

One of the main problems we encountered was data accessibility. While we found a few APIs that allowed us to scrape news websites and sources, there were constant limitations on the amount of data we could scrape. While NewsAPI would have been the ideal source for us to build our entire dataset, due to license restrictions, we were limited to pulling a maximum of 2,000 articles each day, only going as far back as 30 calendar days. Additionally, the license restricted access to the body of articles, which had to be scraped using URLs. We weren't able to build a cohesive dataset this way so had to rely on using a single source, the Guardian, for much of our methodology. While we were able to get the entire US Newsstream database for September, the dataset could only be accessed through ProQuest's AWS instance and there were limitations on the packages we could download to help us explore the data and output the summaries. Again, due to license restrictions, we were not able to pull the full articles locally or upload to the NYU HPU Cluster.

Another challenge in this project was the evaluation of our algorithm. As this was an unsupervised learning task, and none of our datasets had the ground-truth labels for evaluation, we were limited in the metrics we could optimize and instead, had to manually evaluate many of our clustering and summarization algorithms. For summarization, as there were no reference summaries, well-defined metrics like BLEU and ROUGE were out of the equation. Hence we could only approximate how well the cluster summaries were using human evaluation. Our best bet was to manually go through all generated cluster summaries and compare them to each other, marking down which summary was 'better' (i.e. more cohesive).

References

1. How Many Stories do Newspapers Publish Per Day?. The Atlantic; 2016. Available from: https://www.theatlantic.com/technology/archive/2016/05/how-many-stories-do-newspapers-publish-per-day/483845/.
2. Carta S, Consoli S, Piras L, Podda AS, Recupero DR. Event detection in finance using hierarchical clustering algorithms on news and tweets. PeerJ Computer Science. 2021;7:e438.
3. Fan A, Grangier D, Auli M. Controllable Abstractive Summarization; 2018.
4. A Text Summarizer & A Mind Reader By Facebook — How Will They Make A Difference?. Analytics India Mag; 2020. Available from: https://analyticsindiamag.com/a-text-summarizer-a-mind-reader-by-facebook-how-will-they-make-a-difference/.
5. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013.
6. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, et al.. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension; 2019.
7. Fine-Tuning the BART Large Model for Text Summarization. Medium; 2021. Available from: https://towardsdatascience.com/fine-tuning-the-bart-large-model-for-text-summarization-3c69e4c04582.
8. Ester M, Kriegel HP, Sander J, Xu X, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd. vol. 96; 1996. p. 226-31.

Appendix: Group contributions

- Saumya Didwania:
  - Dataset Sourcing
  - Evaluating Alternate Clustering Procedures
- Jonathan Lu:
  - Text Pre-Processing
  - Summarization Deployment
- Saumyaa Shah:
  - Web Scraping
  - Article Embedding + NER
  - Clustering
- Shared Tasks:
  - Poster Presentations
  - Paper Write-up