Topic modeling and sentiment analysis on news media sources

Lisette Solis, Jackie Glasheen, JP Martinez, Megan Moore

Abstract:

The objective of this project was to identify topics in news articles and explore differences in the amount of coverage and sentiment of the coverage published by different news sources. We identified the topics of articles through clustering. We explored variations of k-means, hierarchical and BertTopic clustering, and ultimately, the most clearly defined and coherent clusters were produced by the BertTopic model. In addition we performed sentiment analysis using a pre-trained BERT model and nltk VADER model on the different article clusters. The main findings of our analysis are 1) there are differences between the amount of coverage which sources give to different topics which in some cases may be tied to the sources political bend, and 2) while most articles in our dataset were classified as neutral, some sources have a much wider range in the sentiments of the articles published on different topics. This is meaningful because credible news articles should be written in neutral, factual language.

Introduction:

As people typically get their news from only a few news sources, differences in the amount of coverage on an issue and in the positive or negative sentiment in coverage of different stories can have important implications on public opinion. This is especially relevant in the current polarized political climate. This project leveraged a large dataset containing articles from the politics section of several US news sources to implement a series of embedding processes, clustering methods, and sentiment evaluation methods in order to understand effective methods for measuring topics and sentiment in the news. All source code and visuals can be found in the LingoQuartet repository on GitHub: https://github.com/meganhmoore/LingoQuartet.

Related Work:

To inform our selection of methods, we first found the prior work related to the space of efficient clustering methods for topic modeling as well as sentiment analysis, particularly for news article topics.

The work by Bisandu et al. evaluated a range of similarity measures for clustering techniques on text data. The findings from this paper informed our selection of cosine similarity as both an effective and computationally efficient method for determining article relationships.

The work by Bojanowski et al. regarding combined word2vec and char n-gram embeddings for document classification informed our decision to test a non-contextual embedding method with lower dimension against a contextual embedding like BERT with higher dimension.

The work by Burscher et al. evaluated k-means clustering and sentiment analysis as a method of producing “news frames” of reporting on nuclear power, and suggested methods of improving clustering that results in improved results. This research tested limiting text to the headline of news articles and the lead of the article which seemed like an effective method of preprocessing articles to a more concise, but still effective representation of the news topic. This informed our decision to use k-means clustering as well as to test a subset of each news article composed of the title and first sentences of the article up to 512 characters.

The work by Yadav et al. implemented sentiment analysis on financial news using unsupervised methods. For their domain, they built off of Turney et al’s work. This seminal work in sentiment classification used unsupervised methods to classify positive and negative sentiment of reviews using phrases containing adverbs or adjectives and used PMI-IR to calculate semantic orientation.

Yadav et al. however, found that this existing technique was not effective in determining sentiment given that financial language is very particular and may contain context that pre-trained models have not been exposed to. For the financial news, they found that noun-verb combinations yielded the best sentiment interpretations. Given that this work seemed more aligned with the news article text we would be working with, it was useful to inform our work to explore fine-tuning and to preemptively understand the likely differences in review-based sentiment and news sentiment.

Description of Solution and Work Done - Methods:

Data:

The data source was originally 512,978 articles (2GB of text data), from this we took a subset of 37,044 articles from 7 different news sources from 2020 alone. Since the goal of the project was to capture clear, separable topics with identifiable sentiment we wanted to 1) restrict the time frame to a range in which we could define expected topics and 2) define the sentiment of those topics in that time frame knowing that public sentiment and news interpretation of issues changes over time. Using a wider time range risks more sentiment variation but increases the size of the data set thereby providing our models with more examples to learn from.

The data was obtained from a project that worked on obtaining the articles from the politics section of all of the main news media sources from 2015-2023. The data used was obtained from an early iteration of the process where only a limited number of news sources had finished data collection.

Pre-Processing:

For the clustering of news articles, we wanted the articles to maintain their natural structure and context, so we did not remove punctuation, stop words, etc. as might be common in other NLP tasks. Instead, we only removed certain non-text character codes (like unicode characters for newline, tab, etc.), and made the text lowercase. We also prepared representative text for each article to use in the clustering algorithm, consisting of the concatenated article title and text, up to 512 characters.

For the sentiment analysis, we did not change the native uppercasing/lowercasing as casing can be an indicator of sentiment. We also corrected systematic raw-data errors that were impacting sentiment scores, removed articles without text, and removed quotes.

Modeling:

As a first objective of the project, we were interested in clustering the articles to identify the main topics that were covered during 2020. The main idea of clustering is using unsupervised methods to group the vector representation of articles such that similar objects are grouped together into the same cluster. Given that the vector representation of the article will depend on the text, we would expect that the clustering will be representative of news topics.

For the clustering exploration we used three different methods (i) k-means clustering, (ii) hierarchical clustering; and (iii) BERTopic

K-means:

K-means clustering is a hard clustering method, assigning each observation to one cluster –in contrast to other methods of soft clustering where an observation is assigned a probability to be in different clusters–that partitions the data in exactly *k* distinct non-overlapping clusters.

Because the algorithm partitions the data into *k* predefined number of clusters, it has the assumption that the *k* chosen is a good representation of how data should be partitioned. For some applications, the data could give a clear insight on the number of clusters to define, but in many others *k* should be found by testing *k* as a hyper parameter. In our case, we tested different numbers of clusters and evaluated them using the “silhouette score” as described below.

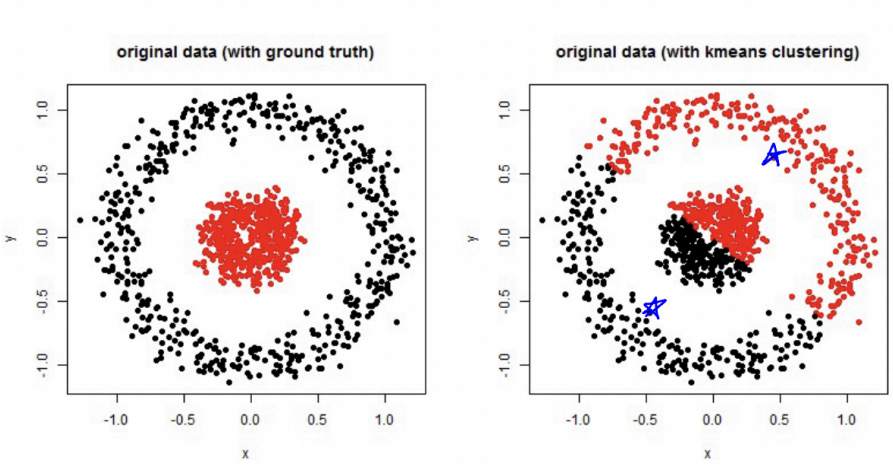
In this method each observation is assigned to the cluster to which the squared distance to the cluster centroid is minimized. One common algorithm to perform k-means clustering is Lloyd's algorithm which we used to find the clusters.

At a high level, the clustering algorithm perform the following steps (i) initializes a set of *k* centroid means, (ii) assign each observation to the closer cluster centroid; (iii) recompute the mean of each cluster, and repeats steps (ii)-(iii) until reaching the convergence or stopping rule,for example maximum number of iterations.

One relevant component of the algorithm is that while the algorithm usually converges, different initializations could output different results, and the randomization component of the initialization method could impact the cluster creation. For our implementation of k-means clustering we used k++ initialization, which pick a random observation from the data and assign it's value to the mean of the first cluster, then pick the furthest observation from the first point, and so on for the k-clusters, assuring an initialization where the points are sparse between each other. This initialization method might not assure obtaining optimal clusters, but considering that initialization finds k sparse clusters, convergence is faster and the complexity of the algorithm is lower than other initialization methods.

One assumption of k-means clustering is that the data is linearly separable, which limits the ability of the algorithm to identify more complex relationships between the data. For example, Figure 1 shows an example where there is a clear delineation between the data points but k-means finds a linear separation which misclassifies the points.

Figure 1: Separable lineary assumption of k-means clustering



To learn more complex relationships, other methods like kernel k-means have been used, where a kernel representation of the vector is used to have a higher dimensional representation of the vector that will allow to find higher-dimensional relationships of the data.

To cluster observations we first need to convert the articles to embeddings so we can calculate the distances between the different points and the cluster's centroids. For this task we used two different methods, BERT base, uncased embeddings and term frequency - inverse document frequency ("**TF-IDF**") representation of embeddings.

BERT embeddings are built from a transformer-based model that is capable of capturing highly contextualized information. This seemed like a useful component, given that the news articles can often use more standardized "newspeak" and so the context and connotation of the

language will have an impact in parsing apart different topics that may otherwise use similar language.

TF-IDF is a method that tries to retrieve for each word in a document, how important that term is in each document relative to the corpus of documents. The idea is that we want to identify the words that appear more often in a document, but controlling for the fact that some words might appear often in many different documents. The TF-IDF is the product between the Term-Frequency ("**TF**") and Inverse Document Frequency ("**IDF**") where we can formally define TF and IDF as:

$$TF_{i,j} = \frac{(n° \ of \ times \ word \ i \ appears \ in \ document \ j)}{n° \ of \ words \ in \ document \ j}$$

$$IDF_i = log(\frac{(n° \ of \ documents)}{(n° of \ documents \ that \ contain \ word \ i)})$$

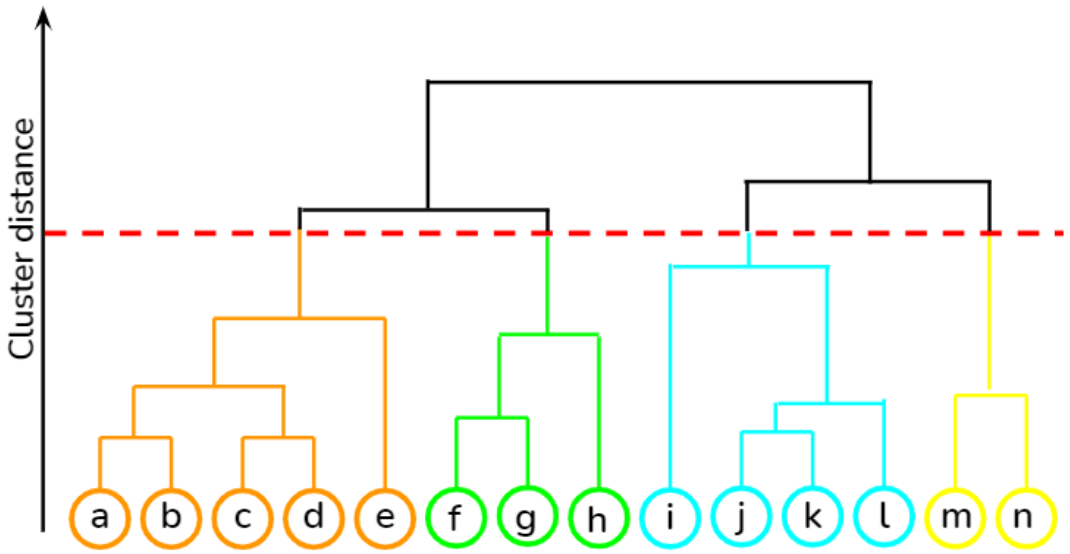
The TF has one value for each word *i* in document *j* and gives the relative frequency of that term in the document, and the IDF is one value per word and it's the logarithm of the number of documents divided by the number of documents containing the word where we can see that words that are more uncommon will be assigned a higher IDF.

For the embeddings we calculate the TF-IDF such that we have a matrix of dimension *n*x*m* where we *n* is the number of documents and *m* the number of words such that we have one vector per document where at position *i* it contains the tf-idf of word *i* for that document. Given that the number of words will highly exceed the number of documents, the matrix representation will be a sparse matrix.

Hierarchical

Hierarchical modeling, also known as agglomerative clustering, approaches clustering very differently than k-means. It starts by assigning each observation to its own cluster and only joining observations together that are closest to it. Then after creating a hierarchy of distance relationships between observations, a threshold is determined at which the observations that are connected by that point will be combined into a single cluster.

Figure 2: Hierarchical clustering representation



This is different from k-means clustering in that there is not a predefined number of clusters that may necessitate forcing observations into a cluster that do not closely align with the rest of the observations. The tricky part, however, is determining the correct cutoff from which to define interpretable clusters.

In order to define the hierarchy, we start with our embeddings. Given that the embeddings are the format for which we determine distances, it is important to select an embedding that effectively combines the topic and contextual impact of the news articles such that similar news topics will be closer together. Therefore we used two different embedding methods, BERT base, uncased embeddings, and FastText embeddings. The K-means section also used BERT embeddings so this section will only go into depth about FastText

The second type of embedding we used was FastText which has been noted as a useful method for embedding when conducting text classification, including in hierarchical clustering contexts. It uses a word-level approach similar to word2vec as well as a char-ngram component that allows it to detect morphologically similar words and perform more robustly on out-of-sample vocabulary than word2vec alone. Notably, however, it is a non-contextual model and does not account for differences that would be learned from the words and sentences around a given token.

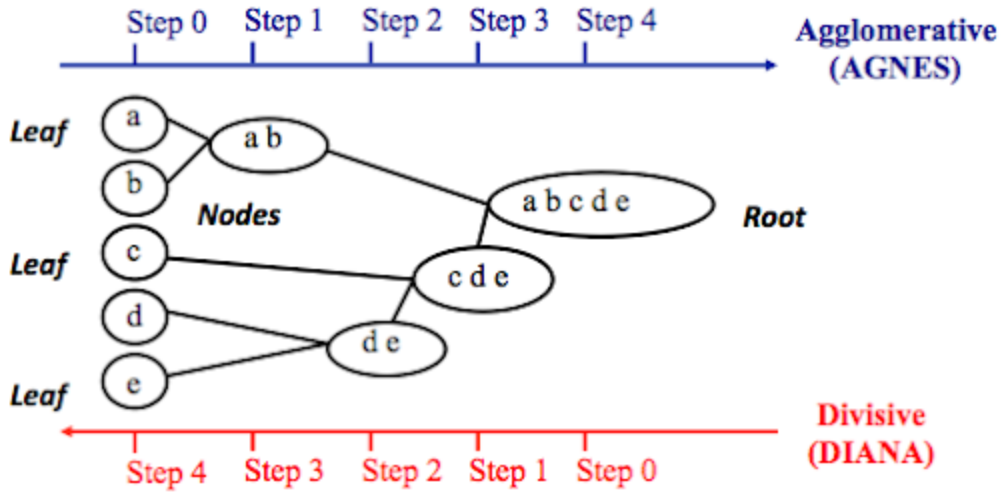
After creating the two types of embeddings, we compute the cosine similarity scores. The math for computing the cosine similarity score is as follows:

$$cos\,(\theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^{\,2}}\sqrt{\sum_{i=1}^{n} B_i^{\,2}}}$$

Where A and B are two different embedding vectors. By taking the dot product ($A \cdot B$) and dividing by the euclidean norms of each vector ($||A||$ $||B||$), this effectively normalizes the magnitude of the two vectors and isolates the angular differences between the two vectors in the hope that, given the embedding method, the similar topics will have similar angles and overlapping vectors (though potentially different magnitudes).

Then, once the cosine similarity (and dis-similarity) are computed, we use the ward clustering algorithm to define the hierarchy of similar news articles. This algorithm minimizes within cluster variance. At each step it merges the two clusters that have the smallest between-cluster distance. In doing this, if the embeddings were effective, it will hopefully mean that similar articles will end up in similar clusters. The ward algorithm produces a linkage matrix from which we can determine the step/distance for which to cutoff and define separate clusters from.

Figure 3: Agglomerative and Divisive Comparison



For this project we built ward cluster hierarchies for both FastText and BERT embeddings. Then we tested a range of distance cutoffs, from which we assessed the silhouette score (discussed in the analysis) and tf-idf top words by cluster to determine the most appropriate fit for the news article corpus.

BERTopic

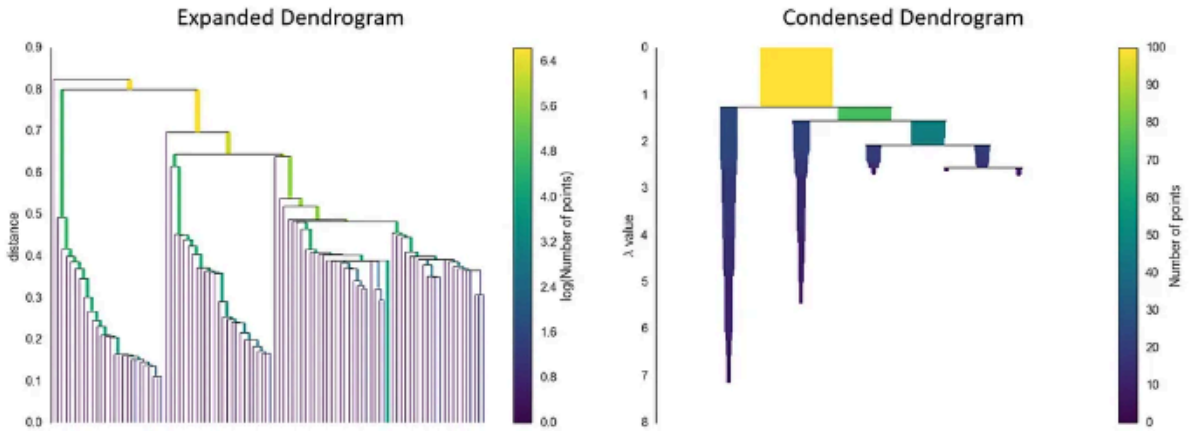
In addition to the manual implementation of K-means and hierarchical clustering, we also utilized BERTopic for an alternative topic modeling implementation. BERTopic is a modular software package that performs topic modeling on unlabeled text data. The model follows similar steps as manual clustering implementations, the key difference being that it is “off-the-shelf”. BERTopic uses default settings/data models that have been found to work well for topic modeling tasks, making it relatively adaptable and capable of achieving coherent topic clusters without much fine tuning.

In our implementation, raw article text is transformed into dense vector representations using sentence-transformers. In particular, "all-MiniLM-L6-v2" was used, which generally works well at capturing semantic similarity across sentences, while being 5x faster than the full base model. Then, UMAP is implemented to reduce the dimensionality of the embeddings to make them compatible with clustering. UMAP captures both the local and global high-dimensional space in lower dimensions.

Then, the embeddings are clustered hierarchically with HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise). At a high level, density clustering algorithms identify clusters by looking for high-density areas of embeddings surrounded by low density areas. Using a density based algorithm circumvents the k-means bias towards circular clusters (which occurs by virtue of clustering data by assigning it to the nearest centroid). In our use case, the handling of outliers was also quite important. Because articles may be written on a large range of topics, we do not ideally want to force all articles into clusters, as an article that is unique in topic (“noisy” data) being forced into a cluster would make the cluster less coherent as a topic representation.

Traditional hierarchical models (like the ward distance clustering algorithm used above) merge iteratively to form clusters based on points closeness to each other. But, HDBSCAN is not an agglomerative model. The high density hierarchical model builds a hierarchy based on a minimum spanning tree, which connects all the data points such that the total edge length is minimized. We then extract clusters from that tree by allowing tree splits that result in a *small* number of data “falling out” of the cluster – this is implicitly removing low density areas. This process is known as condensing the dendrogram. Clusters are then defined if they are greater than the minimum cluster size, which defaults to 10.

Figure 4: BERTopic Dendogram



Once the clusters are formed, we find topic representations for the clusters. This is accomplished by first making a bag-of-words representation of all articles in the cluster. To define clusters in terms of what makes them unique from other clusters, the algorithm performs class-TF-IDF across clusters (similar to the manual implementations above). The most important words per cluster are extracted to get descriptions of topics. We tested creating topic representations using n-grams, but found the single-word representations more coherent upon visual inspection.

We also experimented with fine tuning representations by implementing KeyBERTInspired, which aims to improve coherence and remove stopwords from the topic representations. KeyBERTInspired considers the semantic relationship between keywords within a given cluster and calculates similarity between candidate keywords and the topic embeddings. Given the interpretability of the starting clusters, we did not implement KeyBERTInspired in the final model.

Topic Analysis

After assigning each article to a cluster, we need to evaluate the performance of the clustering process.

As a first step, we computed TF-IDF to retrieve the tokens that are more salient in each cluster. To do this, we perform TF-IDF as explained on the k-means clustering section but at the cluster level, where we concatenated the text of each article on a cluster, getting one document per cluster and then running TF-IDF to retrieve the words more salient to that cluster.

This method works as a good check of our own understanding and perception of a cluster making sense and checking if the articles contained in the cluster represent indeed a topic that was covered in 2020, particularly for checking for the clusters with more articles assigned if it resembles a topic that might have been widely covered during that year.

Besides a heuristic check of clustering outputting reasonable results we wanted to quantitatively evaluate the accuracy of the clusters. Given that we don't have a true label for which cluster each article belongs to, we can't measure accuracy of the clusters by comparing the assigned cluster by the algorithm to a true label and instead we relied on unsupervised learning metrics that measure quality of cluster creation, in particular, the Silhouette Score.

The Silhouette Score is a method to test performance of clustering by obtaining a metric of how well defined the clusters are by measuring for an observation how similar the observation is to the other observations in its cluster, and how different the observation is with respect to the observations that are not part of the cluster. Then, to test the overall accuracy, the average Silhouette Score is calculated.

Formally, we can define the silhouette score as the ratio between, the difference between the average distance between point *i* and the points outside of the cluster *i* is assigned, b_i, and the

average distance between point *i* and the points inside its cluster, a_i,, and the maximum between b_i, and a_i.

$$silhouette\ score\ =\ \frac{(b_i - a_i)}{max(b_i, a_i)}$$

The Silhouette Score ranges from -1 to 1, where a value towards 1 indicates that the clusters are well defined, with a good separation between them, a value towards -1 is indicative of the clustering method likely misassigning the points to the cluster, and a value close to 0 is indicative of the clusters being really close and close to overlapping themselves.

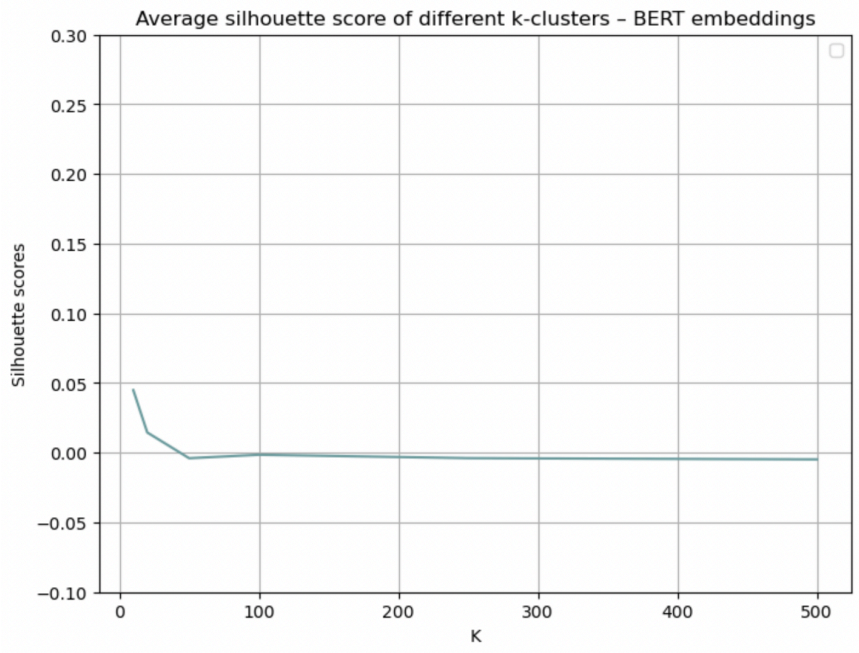
Evaluation of Clustering Methods:

Clustering:

K-means

For k-means clustering we created clusters for different numbers of the parameter *k* which took the values 10, 20, 50, 100, 250 and 500. As explained in the Topic Analysis section we calculated the average silhouette score of each clustering calculation which results for the BERT embeddings are in Figure 6 where the x-axis represents the number of clusters defined and the y-axis represent the average silhouette scores across all observations when partition into the *k* clusters

Figure 6: Average silhouette scores for BERT embeddings



From these results it is possible to observe that the average silhouette score is close to 0 for each of the different *k* values, which as explained above is indicative of the clusters possibly being close to each other and with some possibly overlapping which could occur in k-means if the algorithm reach a stopping condition on the maximum number of iterations before all the points converge.

For the different *k* we obtained the maximum average silhouette scores when *k* is equal to 10 where it takes a value of 0.045, followed by where *k* is equal to 20 where it takes a value of 0.15.

The figures 7-A and 7-B graph the distribution of the silhouette scores for each observation grouped by cluster, where each color in the graph represents one cluster –with taller clusters representing clusters with a larger number of articles assigned to it–, and the vertical red dashed line represents the average silhouette score.

Figure 7-A: Distribution of silhouette scores for BERT embeddings with k = 10

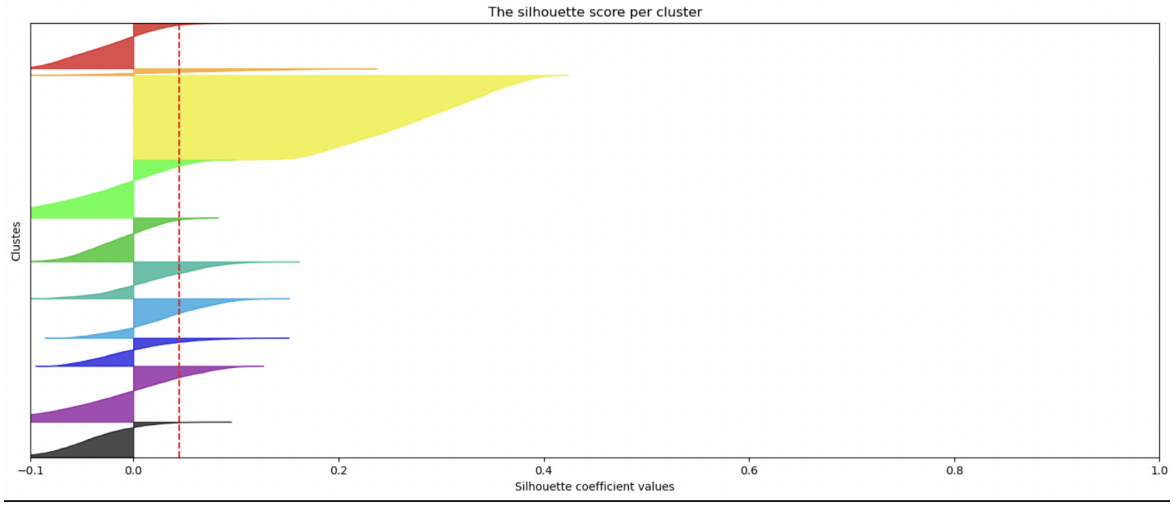
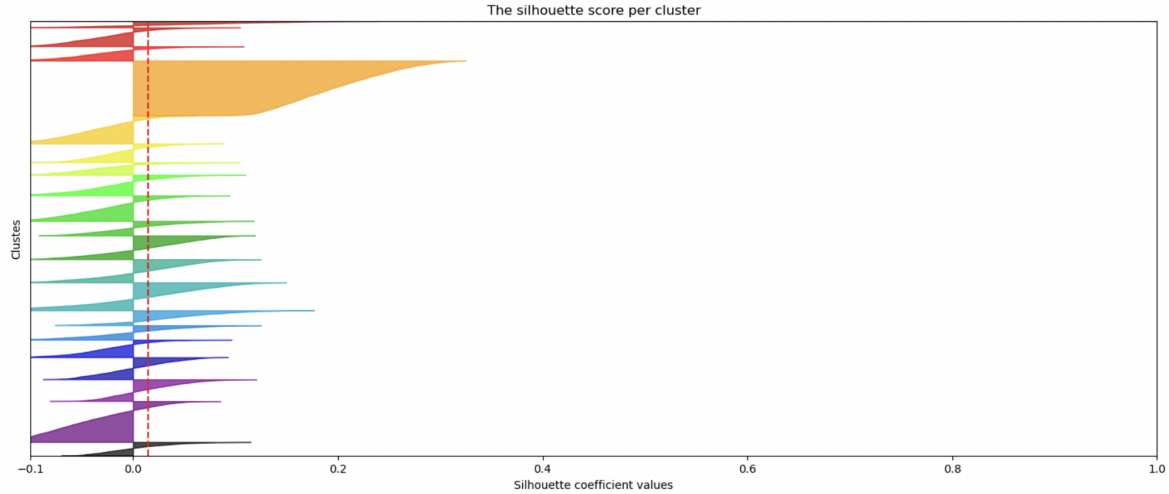


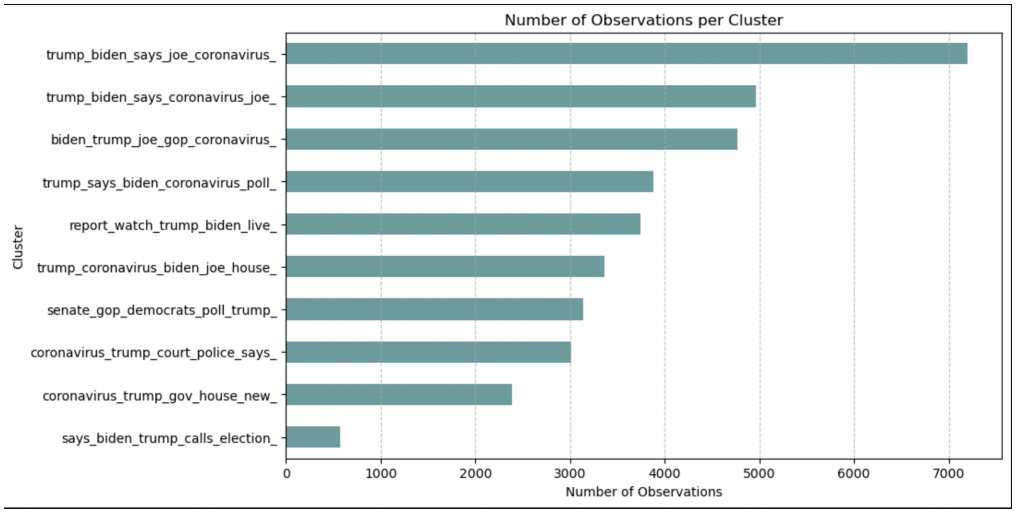
Figure 7-B: Distribution of silhouette scores for BERT embeddings with k = 20



From this graph we can observe that for the cluster with more observations, for both 10 and 20 clusters, this one seems to be more robust, which makes sense as by being the bigger cluster, for many observations closer to the cluster centroid, the distance for the points inside the cluster might be smaller, and those outside the cluster, considering the cluster is large, they might be far away from the other points. For other clusters it is possible to see a larger variance, where a significant amount of observations end up with a negative silhouette score, which again, might be indicative of the algorithm terminating earlier and not at the moment of convergence.

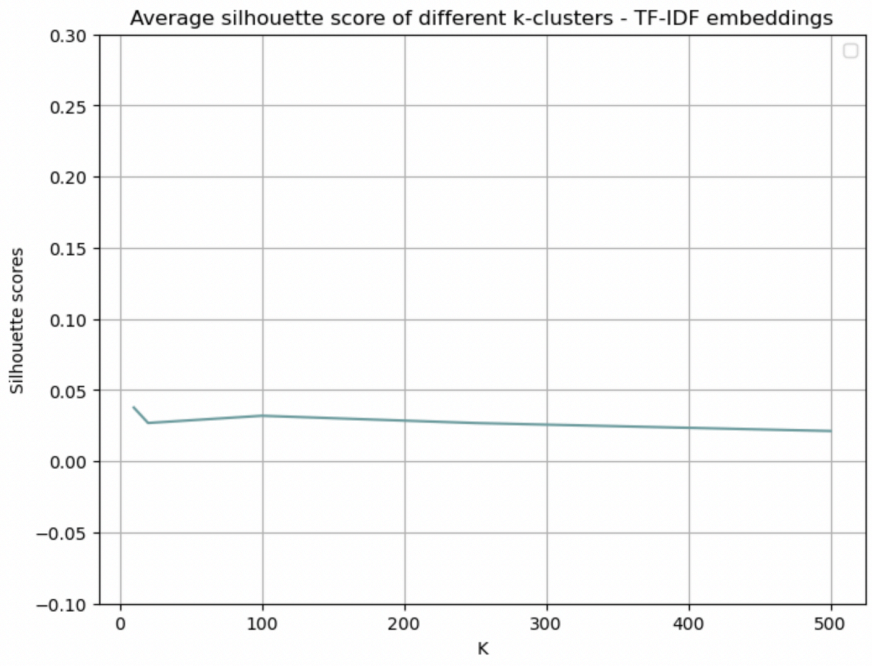
Certainly, given the large number of articles in the dataset, we would expect that classifying the articles in only 10 or 20 clusters to summarize the topics of the corpus of text we have, would be insufficient. On the following image we have the number of observations, and the main features of each cluster obtained for the top 10 clusters where it's possible to observe that each cluster has a large number of observations, with the main cluster having assigned over 7,000 articles, and where it's possible to observe repetition on the key words of different clusters for which we can't get a clear sense of the represented topic.

Figure 8: Number of observations per cluster for k=10 with main words for BERT embeddings



For the TF-IDF embeddings we obtain the following figure that plots the average silhouette score for the different values of *k*:

Figure 9: Average silhouette scores for TF-IDF embeddings

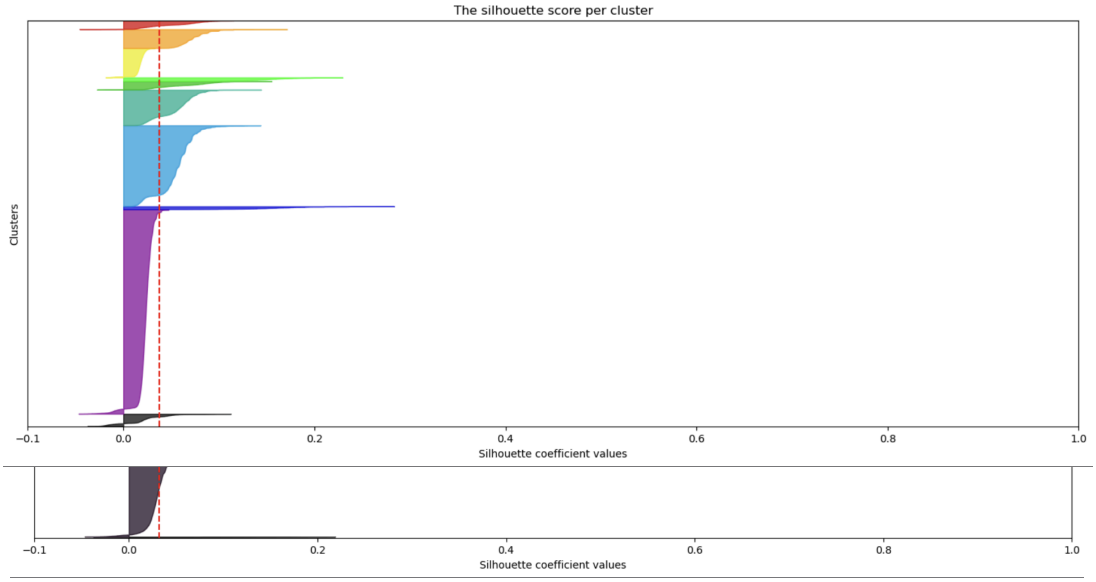


As in the BERT embeddings, the average silhouette scores for the different number of clusters is close to 0 which is indicative of close clusters that might be close to overlapping with one

another. The two number of *k* that had a higher silhouette scores were 10, where the average was 0.378 and 100, where the average value was 0.320

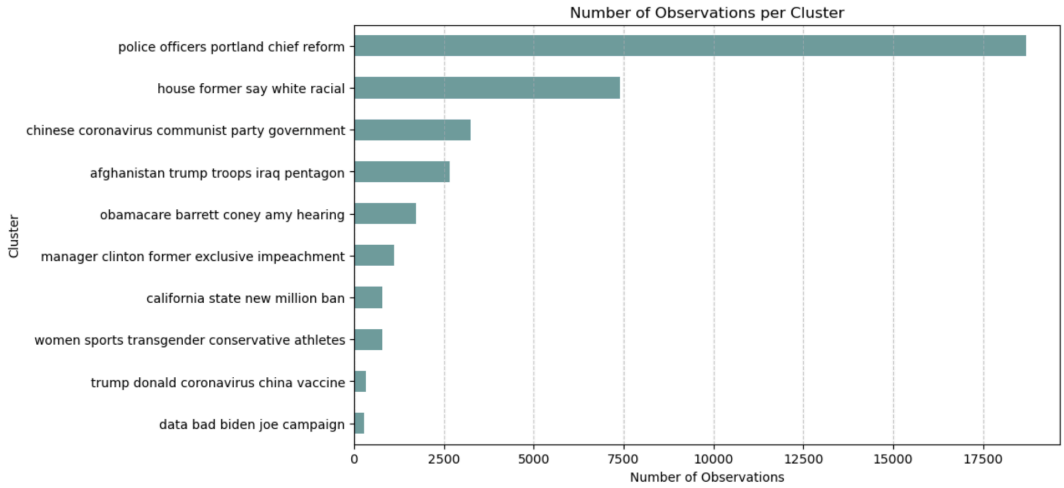
When looking at the distribution of silhouette scores for *k*=10 and *k*=100, we can observe that for *k*=10, most observations have a value greater than 0 but most observations are concentrated close to 0, while for k=100 we observe more variation with a larger number of articles getting a negative silhouette score.

Figure 10-A: Distribution of silhouette scores for TF-IDF embeddings with k = 10



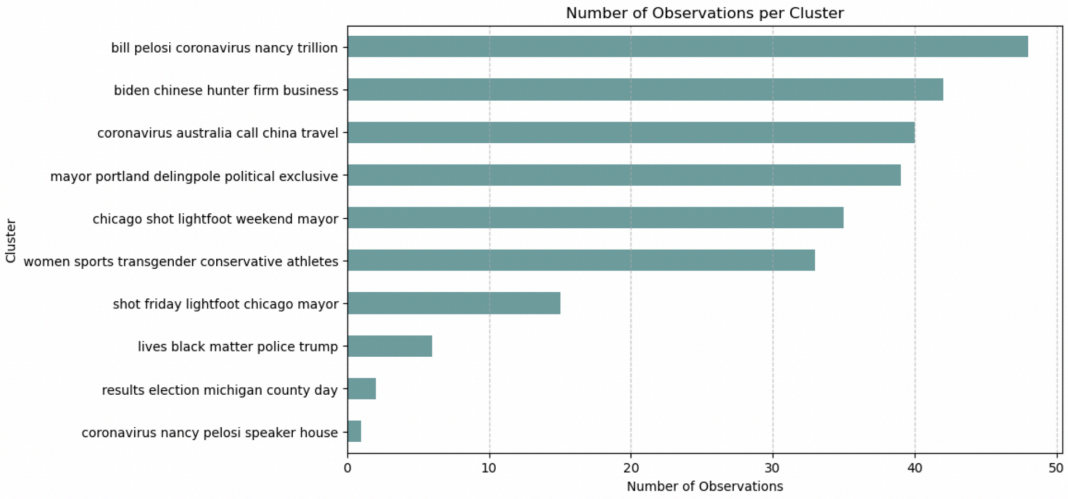
Finally, the next Figures graph for the 10 topics with more observations, the number of articles in each cluster and the top 5 words obtained through TF-IDF. Again, as in the k-means using BERT embeddings, only 10 clusters is too little to capture the main topics, and while the cluster top features could make sense, it's unlikely that they are capturing each topic correctly and we would expect many observations to not be necessarily related to what could be understood as the cluster topic.

Figure 11-A: Number of observations per cluster for k=10 with main words for TF-IDF embeddings



For k=100 the topics look more separable and more specific about a certain topic which from a reader point of view make more sense. Certainly, the topics are not necessarily the topics we would expect as the main topics for 2020, which might be a consequence of the news media sources that are included in the dataset with most of them having an editorial line towards right-wing policies.

Figure 11-B: Number of observations per cluster for k=10 with main words for TF-IDF embeddings

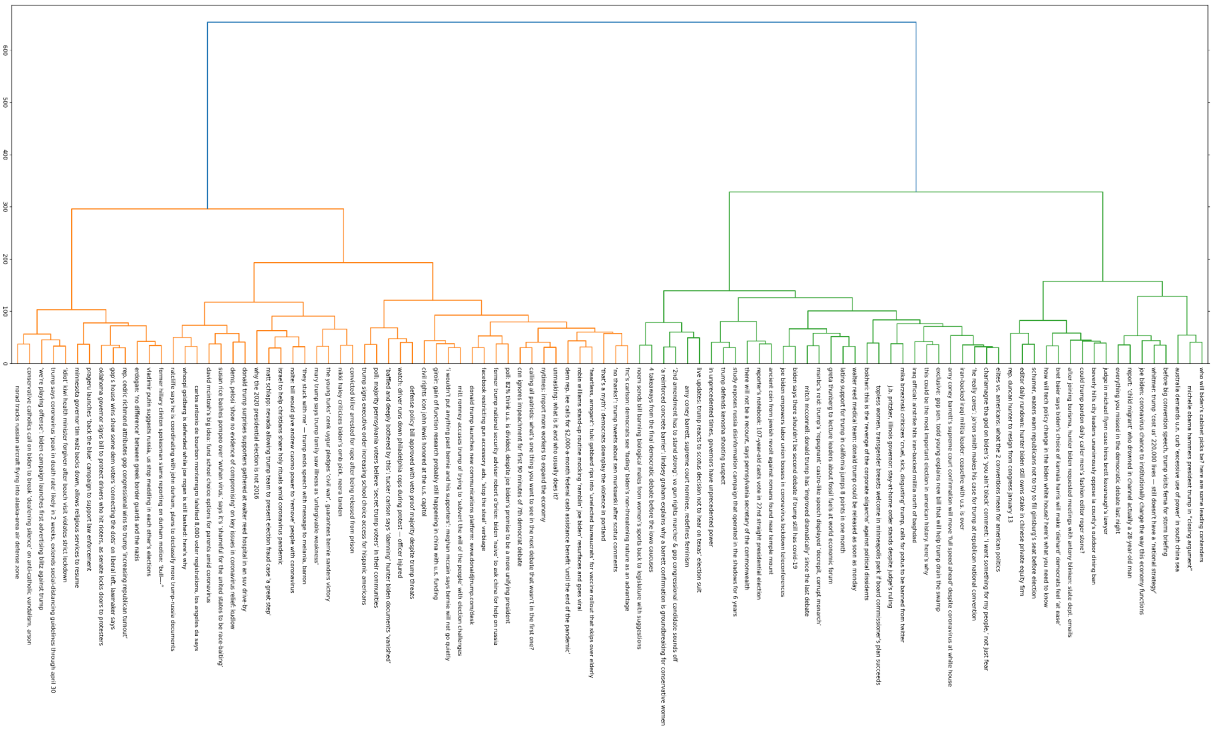


As seen on these results, it seems that k-means on tf-idf help us retrieve better clusters compared to BERT embeddings for this application. Our hypothesis is that while tf-idf is a simple model to estimate embeddings, by focusing on the salient features of a document the vectors are more highly represented about only those main words which should be associated with the topic the article is covering.

Hierarchical

For the hierarchical clustering of FastText we see the following dendrogram for the lastp=100 clustered observations produced by the ward clustering algorithm (the articles themselves are not necessarily meaningful at this point but understanding the shape and distances at which clusters are forming is informative):

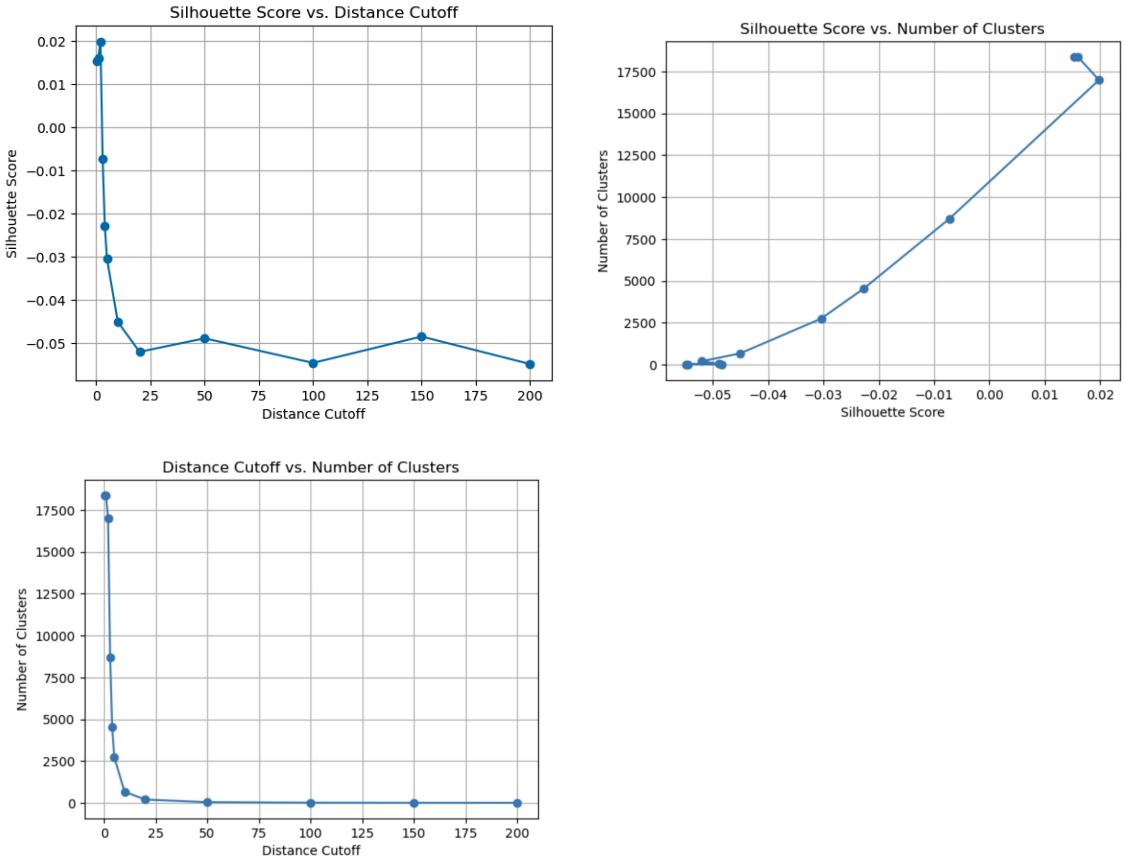
Figure 12: Hierarchical clustering result for FastText



Like the other methods, we use silhouette score at each of the different cutoff thresholds to determine how separable each set of clusters is. Additionally we assess the number of clusters produced and the top tf-idf words per cluster to gather a sense of concrete, different topics that are meaningful.

The FastText produced the following metrics:

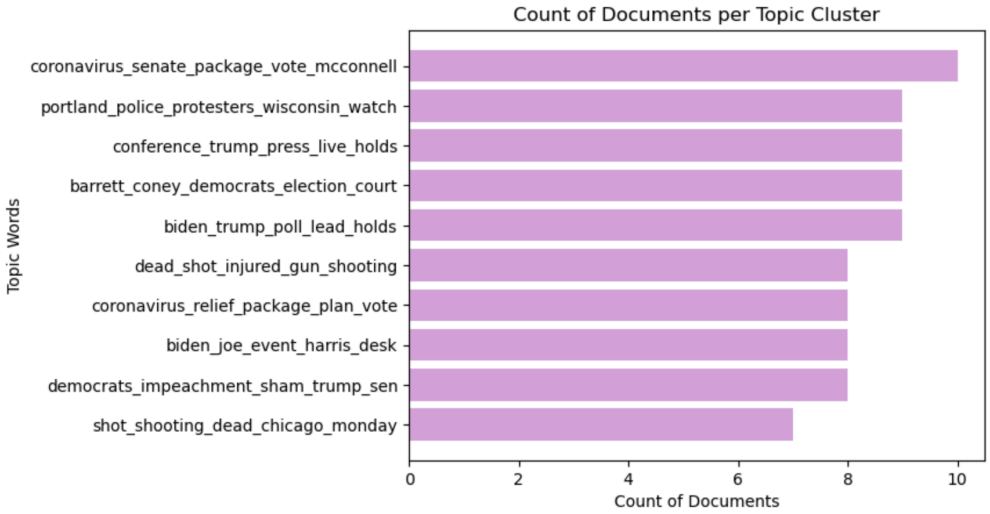
Figure 13: Relationship between Silhouette Scores with distance cut-off and n° of clusters for FastText



From these we see that silhouette score is highest when the distance cutoff is lowest which produces the largest number of clusters. This is not very meaningful because we want to strike a balance between multiple news articles and common topics. Therefore the cutoffs for distances of 3 and 4 seemed to fall between producing too many small clusters, and having some topic separability given the slightly higher silhouette score (though it was a very small margin of variation in silhouette score indicating that these clusters were not particularly separable).

Using TF-IDF we then assessed the top words for the largest clusters produced and found the following:

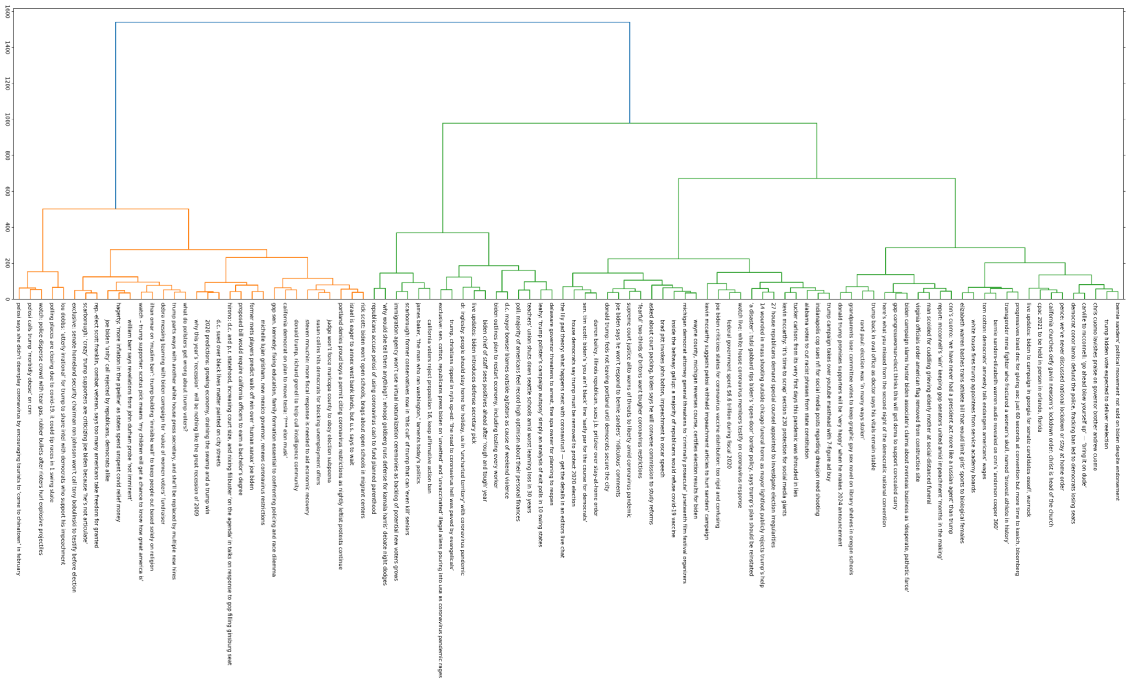
Figure 14: Number of observations per cluster for hierarchical clustering using FastText



Clusters created at cutoff distance 3 seemed to show the most separable topics, and had a silhouette score of -0.0072 with 8,711 clusters. The graph above shows that the largest cluster contained 10 documents and interpretable top words seemed to surface such as discussions around the senate vote on a coronavirus package, police protests in Portland, and Amy Coney Barrett's nomination to the Supreme Court.

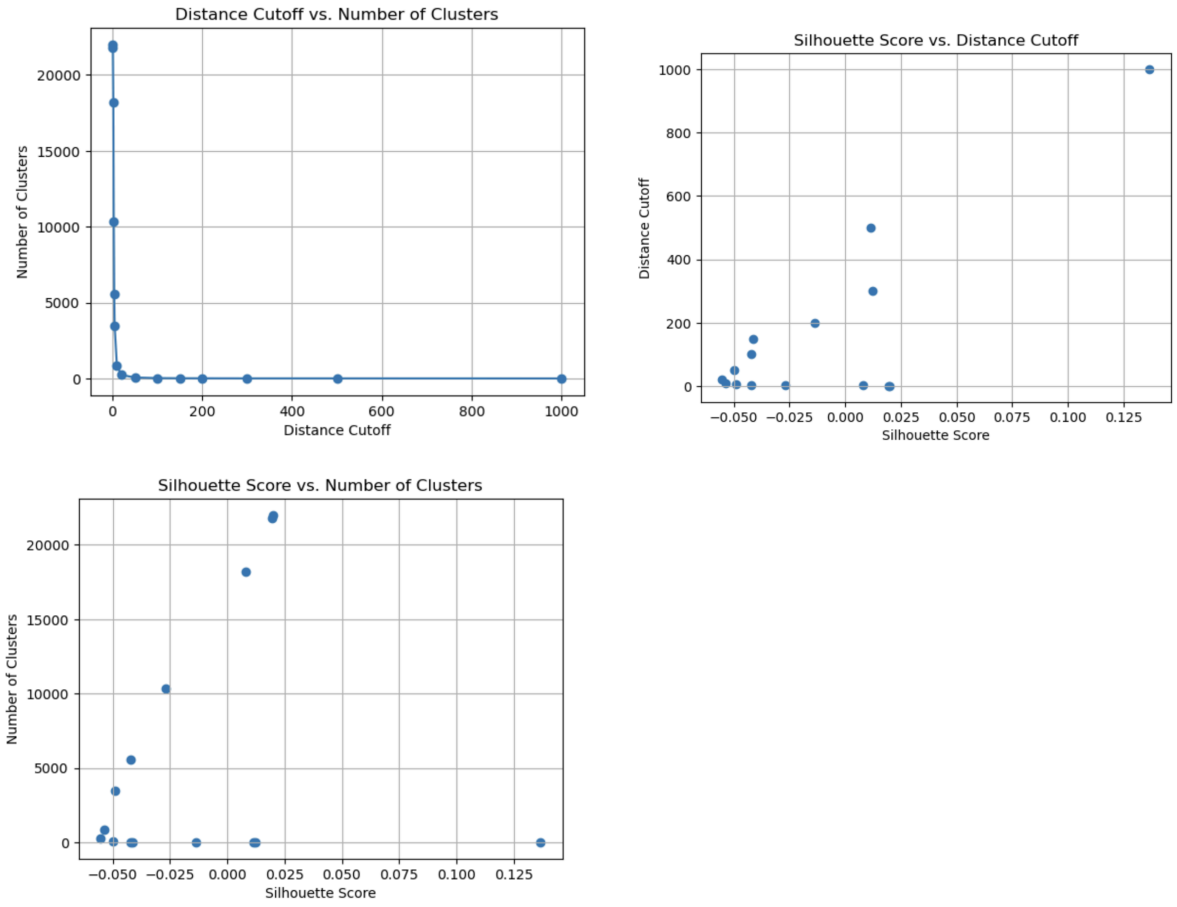
Our BERT Embeddings produced the following dendrogram for lastp=100:

Figure 15: Hierarchical clustering result for BERT embeddings



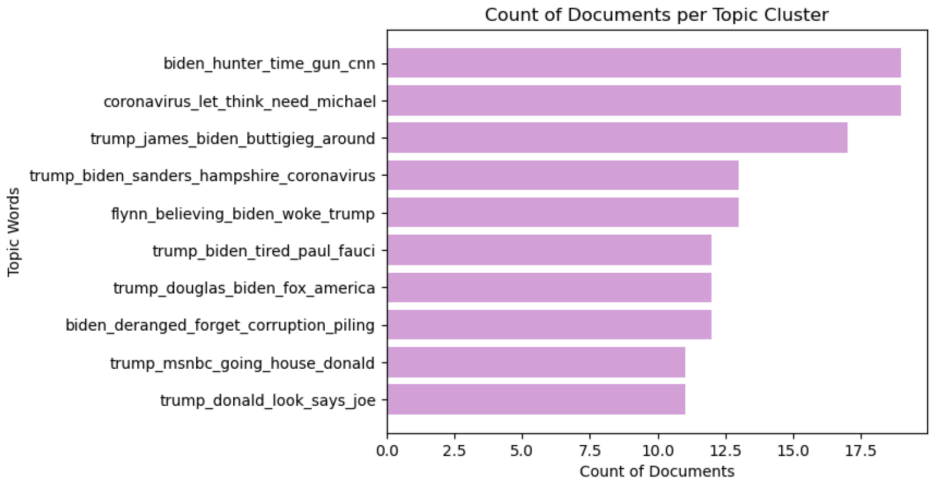
Again we ran the same analysis as with FastText and found that for BERT the cutoffs produced the following metrics:

Figure 16: Relationship between Silhouette Scores with distance cut-off and n° of clusters for BERT



These results indicated a slightly less consistent trend of silhouette scores, seeing higher scores when there are very very few clusters or very very many clusters. That being said, cutoffs of 2, 3, and 4 seemed to strike the best balance between not completely independent clusters, and a higher silhouette score. From assessing the top words, a distance of 1 seemed to parse apart more discrete topics, but all topics seemed flooded with ‘biden’ and ‘trump’. The average silhouette score for the distance 2 clusters was 0.0079, however at first glance this seems to produce less meaningful and varied news topics than the FastText analysis produced.

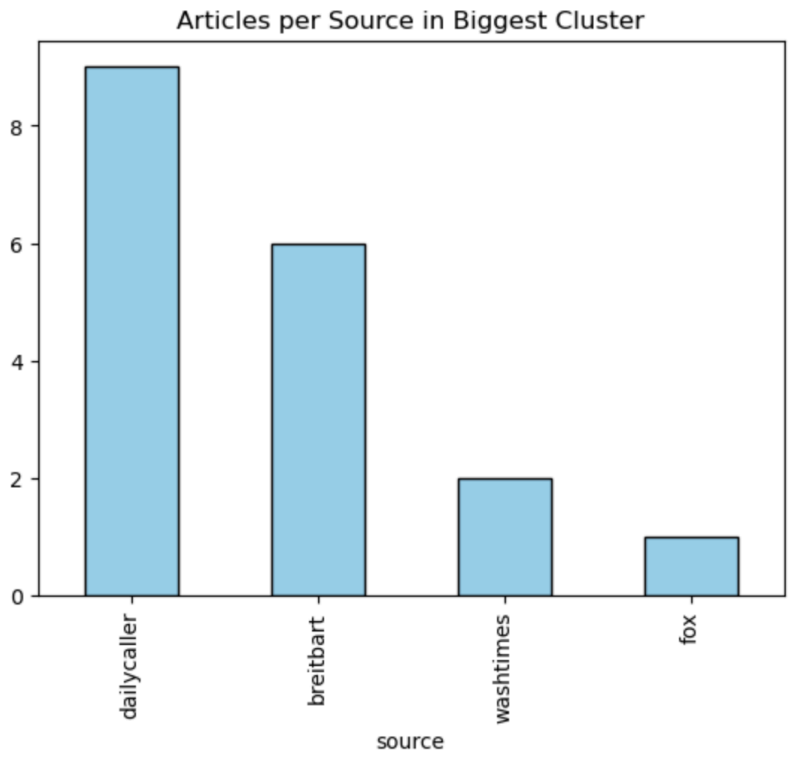
Figure 17: Number of observations per cluster for hierarchical clustering using BERT



A hypothesis for why FastText may have produced more interpretable clusters is that its lower dimensionality provided an advantage in that despite its embeddings not capturing as much contextual information, the word relationships from word2vec and the ability to relate morphologically similar words was sufficient to capture the most important aspects of the news articles that made them similar, without getting lost in the high dimensionality imposed by capturing contextual information, but making clusters harder to define. To gather a sense of this, the maximum distance that the FastText embeddings contained was 653 whereas the BERT embeddings had distances up to 1,541.

To further assess if linguistic trends used by a given source may be getting gathered rather than actual topic similarities, a quick assessment of source distribution in a top cluster shows that there are 4 different news sources covering the biggest topic (albeit conservative news sources):

Figure 18: Number of observations per news source for hierarchical clustering



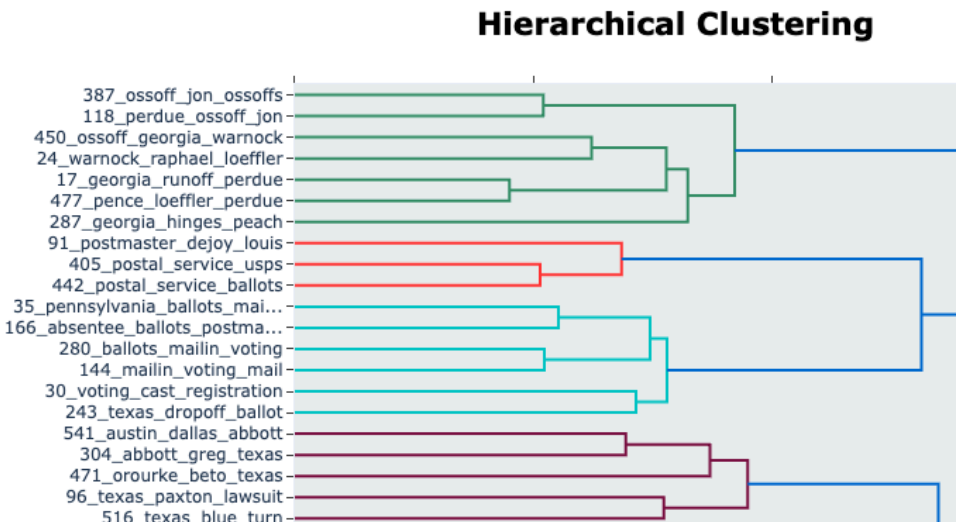
That being said, for the sentiment component of the project, FastText would likely be less useful since context is critical to determining the sentiment of a document and word variation likely has less impact.

BERTopic

The BERTopic model implementation produced about 550 clusters, with cluster sizes ranging from 10 to 664. There were 11,736 (32%) outliers identified that we excluded from subsequent sentiment analysis.

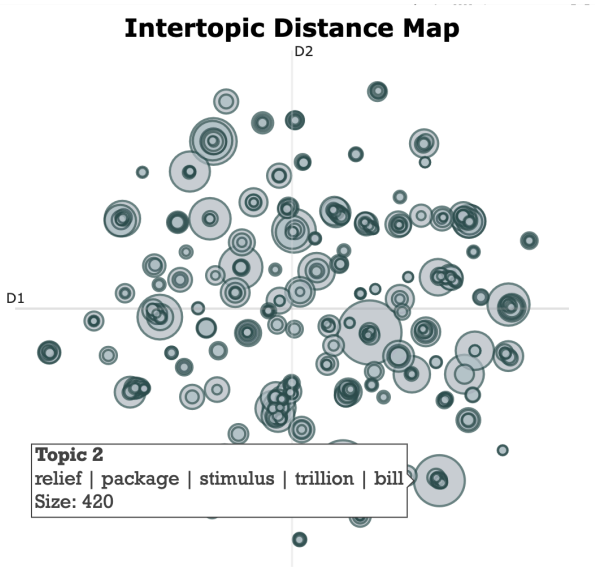
The HDBSCAN resulted in clusters that had similar topic representations in adjacent tree leaves and nodes. As shown below in a portion of the model's hierarchical clustering results, the first two clusters have to do with "jon ossoff", in the next two "warnock" show up, and then there is a cluster mentioning "runoff". This part of the hierarchy has clear ties to different subtopics in the Georgia runoff senate elections. The next group of clusters seem to be about mail-in voting, also election related. This suggests that the hierarchy is being constructed as intended, with similar clusters appearing close to each other in the tree structure.

Figure 19: Representation of BERTTopic hierarchical clustering



However, this chart also highlights that further investigation is necessary to determine the ideal number of clusters. While it is possible that the clusters do represent distinct sub-topics and their representations don't capture those distinctions, it's also possible that the articles are very similar and that the clusters should be merged. Indeed, when the clusters are viewed in a 2D representation, it is both clear that the model produced a large variety of sizable clusters, and that many of those clusters are overlapping.

Figure 20: BERTopic clustering mapping



To further evaluate the clusters, we investigated the similarity of topics. To do this, we calculated the cosine similarities of the cluster topic embeddings. In looking at cluster pairs, we find that some topic embeddings are quite similar to each other, with a distance of close to 1. In the second example in the table below, the topic representations are even the same for topic1 and topic2.

Figure 21: Distance between cluster topic embeddings

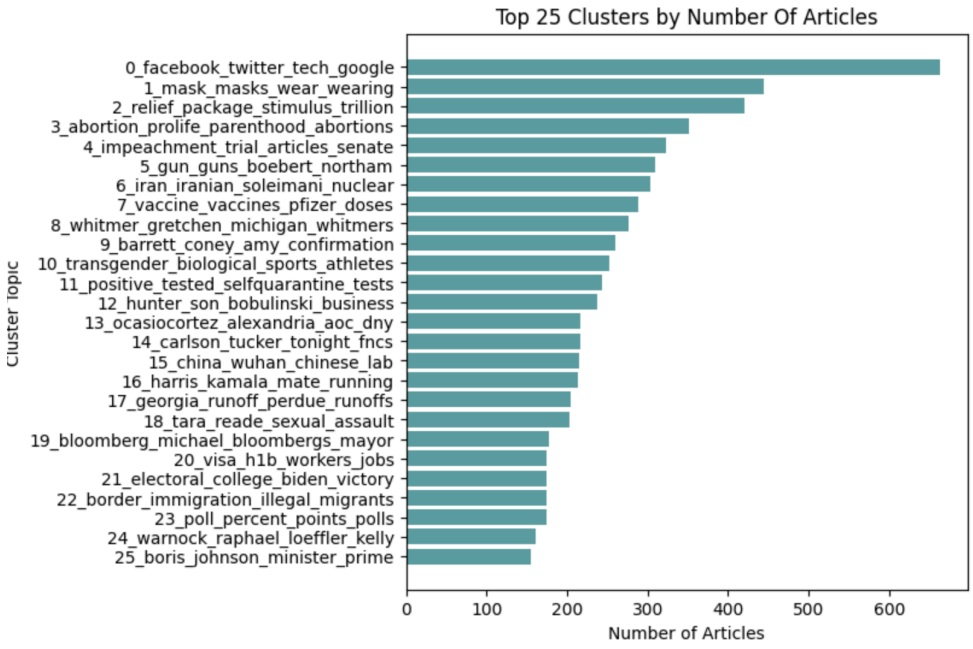
	topic1	topic2	distance
13162	23_poll_percent_points_polls	81_poll_battleground_leads_points	0.915709
92688	169_fauci_anthony_dr_infectious	37_fauci_anthony_dr_infectious	0.907186
89952	164_pelosi_nancy_speaker_testing	26_pelosi_nancy_speaker_relief	0.885869
105165	191_portland_oregon_protesters_lasers	524_portland_oregon_courthouse_fence	0.876341
168208	307_nursing_homes_patients_cuomo	347_nursing_cuomo_deaths_homes	0.875318
2161	2_relief_package_stimulus_trillion	525_relief_agreement_axne_schumer	0.870911
90016	164_pelosi_nancy_speaker_testing	90_pelosi_nancy_speaker_union	0.866566
17795	31_biden_joe_coronavirus_wilmington	354_teleprompter_joe_teleconferencing_virtual	0.862779
115623	211_sanders_nevada_bernie_ivt	82_sanders_bernie_ivt_vermont	0.862674
121053	221_briefings_conferences_lamestream_airing	62_briefing_task_force_white	0.862090
12014	21_electoral_college_biden_victory	23_poll_percent_points_polls	0.851751
144508	264_sanders_bernie_socialist_uygur	82_sanders_bernie_ivt_vermont	0.851580
115907	211_sanders_nevada_bernie_ivt	366_hampshire_sanders_bernie_primary	0.850876

We experimented with methods of adjusting the number of clusters, both by implementing hard cluster number targets and by setting probability thresholds for cluster definitions. Automatic topic reduction reduces the number of topics iteratively when topic pairs are found that exceeds a minimum similarity of 0.9, and continues to allow outliers. While we show above that not many topics had a similarity level above .9, implementing the automatic topic reduction resulted in many small clusters, and then a single large cluster that did not have a coherent topic representation. Additional fine tuning of the similarity threshold is needed to make automatic topic reduction function as expected.

In testing hard cluster number cutoffs, when we set a low number of clusters (50), we noticed topic representations became less identifiable as news events and had more stop words throughout. While higher levels of clusters (like 100, for example) did result in seemingly coherent topic clusters, we were concerned that noise in clusters would make sentiment analysis harder to interpret. So, for our use case, we decided a large number of somewhat overlapping clusters was preferable to a smaller number of noisy clusters.

Despite the concern about having too many clusters that were somewhat overlapping, the implementation provided many coherent topic groupings that were sufficiently large to perform sentiment analysis. In the chart below, we find many of the large clusters seem to correspond with relevant 2020 news topics.

Figure 21: Number of articles and main words for top 25 BERTopic clusters



Sentiment Analysis:

VADER Sentiment Analysis

The goal of sentiment analysis is to assign polarity and emotional tone of statements.To assign sentiment to our clusters, we implemented VADER (Valence Aware Dictionary and sEntiment Reasoner), which is a lexicon and rule-based sentiment analysis tool that classifies sentiment on a positivity/ negativity scale. In order to assign an overall sentiment score to text, the software maps words and other lexical features to VADER's valence scores, which are then summed and then normalized to -1 (very negative sentiment) to +1 (very positive sentiment) for the text. The valence scores were developed by averaging scores assigned by contractors; therefore, VADER's scores are reflective of typical human assessment.

Literature recommended using -.5 and .5 as threshold for labeling sentiment as positive and negative. In our analysis, we also added thresholds at -.3 and .3 to capture somewhat negative and somewhat positive sentiment. This was done to create greater variation in our data since we thought a simple positive and negative ranking would not be granular to understand the difference in sentiment across sources. For instance, it is likely that much of the coverage on the BLM protests in 2020 contains language about protests and police brutality that is scored negatively. We worried that on a binary scale all this coverage would be labeled negative so we would not be able to understand the differences in coverage across sources. We also make use of the continuous VADER compound score in our analysis for this reason.

In addition to scoring lexical features, VADER considers the following heuristics.

- Semantic orientation. Sentiment scores should account for negation statements, intensifiers, and shifts in polarity (“love you, but…”) of words based on the context.
- Capitalization
- Punctuation such as the difference between ending a sentence with a period or an exclamation point.
- Degree modifiers such as “very” or “not much.”

A notable limitation of VADER is that the model is based on social media training data. Social media data likely has different features than published news data. It likely uses more informal language, uses emojis, capitalizations, and punctuation in more expressive ways (i.e., great vs GREAT!!). Further, sentences are possibly less complex than sentences in news articles, both in terms of structure (i.e., fewer subtle shifts in polarity) and in the nuance of the way sentiment is expressed in the news. Another important limitation is that given VADER is a rule-based model with sentiment scores assigned to words, it does not have a way to learn unseen terminology, which is likely the case when encountering varied news topics. Finally, in cases where context plays a significant role in sentiment of a word, VADER may not interpret the sentiment accurately. VADER failed to properly classify the following more complex examples.

Text	**VADER Classification**
"Trump Didn't See It Coming: Coronavirus Deaths Increased Tenfold This Month"	Somewhat Positive Compound Score: 0.2732
'Donald Trump: Facts Are on Our Side in Election Fight, but Time Isn't'	Neutral Compound Score: -0.2023
Delingpole: Dawn Butler MP Is Not a Victim of Police Racism	Negative Compound Score: -0.5083

BERT

BERT is also commonly used for sentiment analysis. The main advantages of using BERT is that transformer models have better contextual understanding than rule-based models like VADER. This helps BERT better understand complex linguistic structures such as negation and polarity shifts. BERT is also better able to learn new vocabulary than VADER which only recognizes lexical features which have been mapped to scores in its dictionary.

One challenge of using BERT for our team was that since our data is unlabeled, we could not fine-tune a pre-trained BERT model to our specific dataset. The pre-trained models available

were either trained on very different datasets such as product reviews[1], or returned a binary or tertiary label.[2] We attempted to address this issue by creating labels for our data using VADER and fine tuning a pre-trained BERT model using these labels. The goals was to improve the VADER model with BERT's improved contextual understanding and ability to learn new vocabulary. Given more time it would have been interesting to experiment with the pretrained BERT models to see if they outperformed the fine-tuned VADER model despite not being fine-tuned with this dataset.

BERT model

We used the pretrained 'distilbert-base-uncased' tokenizer and 'Distill Bert For Sequence Classification' model with the default parameters except that we modified the model to predict labels from 1-5 and to use cross-entropy loss. This is important because cross-entropy loss is better suited to multi-class classification than mean-square loss which is typically used in this model when predicting binary labels.

Pre-trained Model Configuration:

- Number of attention heads: 12
- Number of transformer layers: 6
- Activation Function: non-linear function, GELU (Gaussian Error Linear Unit)
- Masking: Yes
- Dropout: Yes

We trained the model using the following hyperparameters. These parameters were chosen based off standard values seen in literature and then tuned:

Epochs: 3
Batch size: 16
Learning Rate: 5e-5 with 30 warm-up steps
Weight Decay: 0.01

Model Evaluation

Step	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
500	0.753900	0.630716	0.751000	0.725087	0.708302	0.751000
1000	0.566100	0.514297	0.799750	0.786675	0.781031	0.799750
1500	0.379300	0.501745	0.809250	0.801714	0.809622	0.809250
2000	0.352900	0.437180	0.823250	0.820287	0.825735	0.823250

[1] https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment

[2] https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

2500	0.181300	0.574748	0.822500	0.822728	0.826405	0.822500
3000	0.181000	0.578601	0.828750	0.826200	0.826891	0.828750

Evaluation accuracy = 0.829, Evaluation F1: 0.826

Since the evaluation accuracy is not meaningfully increasing after three epochs we stopped training to prevent overfitting. It is difficult to interpret these results in the context of our use case since we want to improve on the VADER labels and do not want to predict the exact same labels. Without labeled data we cannot test whether this model performs better than VADER alone. However, we found a Medium post testing this process on labeled review data which found that the BERT model fine tuned with VADER was 2% more accurate than VADER labels alone.[3] We expect that our model would have similarly positive but modest improvements in accuracy.

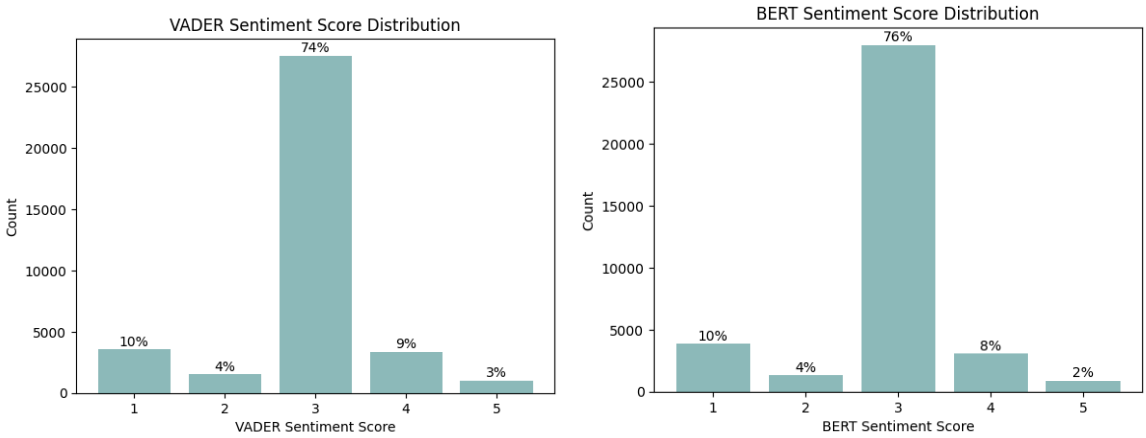
The BERT model changed the label on 3,798 articles in our dataset. When tested on the following example, the fine-tuned BERT model seems to outperform the VADER model:

Text	**VADER Classification**	**BERT Classification**
"Trump Didn't See It Coming: Coronavirus Deaths Increased Tenfold This Month"	Somewhat Positive Compound Score: 0.2732	Neutral (change of -1)
'Donald Trump: Facts Are on Our Side in Election Fight, but Time Isn't'	Neutral Compound Score: -0.2023	Negative (change of -2)
Delingpole: Dawn Butler MP Is Not a Victim of Police Racism	Negative -0.5083	Positive (change of +4)

Overall though, the two models produced very similar results.

Figure 5 A-B: Distribution of sentiments scored for fine-tuned Bert model and Vader model

[3] https://nlpiation.medium.com/is-it-possible-to-do-sentiment-analysis-on-unlabeled-data-using-bert-feat-vader-experiment-357bba53768c



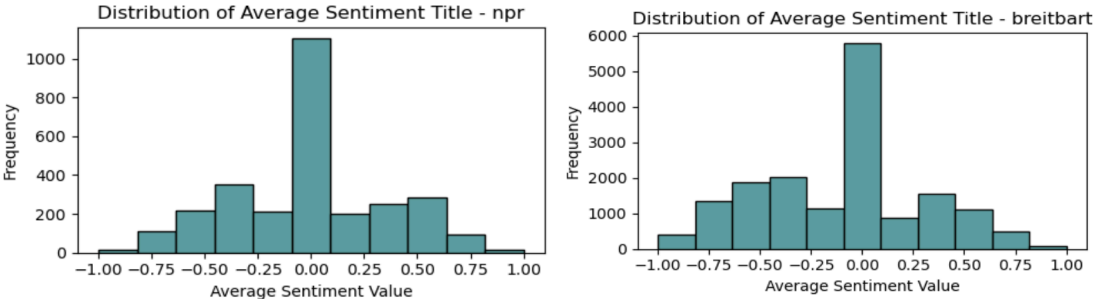
Analysis of News Articles:

Trends in Overall Sentiment

For the sentiment analysis, we proceeded with clusters derived from BERTopic and used VADER to assess sentiment.

To begin, we investigated the baseline distributions of VADER sentiment of news articles. A first key finding is that titles tended to be neutral. This was a somewhat surprising finding, given that we expected titles to contain attention-grabbing statements. Below, the distributions of sentiment for NPR and Breitbart articles peak around 0, or neutral.

Figure 22 A-B: Distribution of sentiment scores for NPR and Breitbart titles

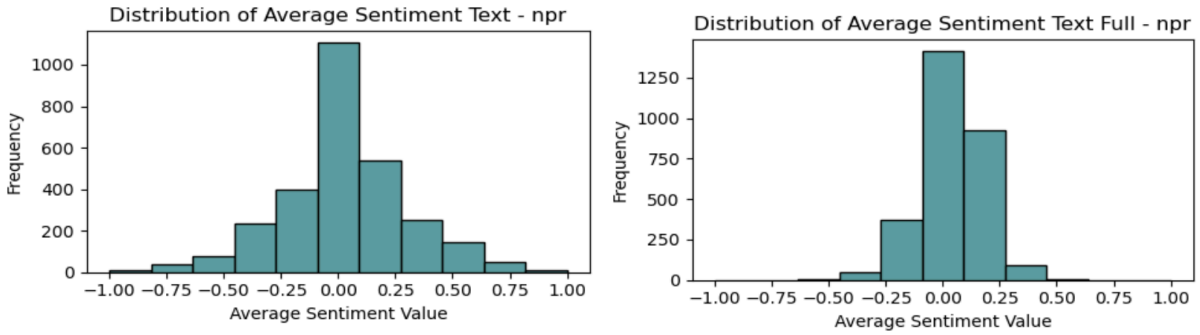


In investigating some scores, we both found many “objective” sounding titles without emotive words, but also recognized that VADER was missing some of the nuanced sentiment within the titles. It seemed unable to recognize some words humans might identify as having sentiment in the context of political news. We think this is a result of the model being developed on social media data rather than news data.

Another key finding is that when sentiment analysis is run on full articles, there is less variation in average sentiment score than when sentiment analysis is run on the first 512 characters of article text. We theorize that the beginning of articles have relatively flashier text than the article

as a whole, possibly to grab the reader's attention. The trend is displayed in the distributions below; sentiment analysis on the first 512 characters of text (left chart) yielded average sentiment score distributions with thicker tails than when sentiment analysis was performed on the whole text (right chart). This also suggests that in some cases, sentiment-heavy sentences may “average out” over the course of an article.

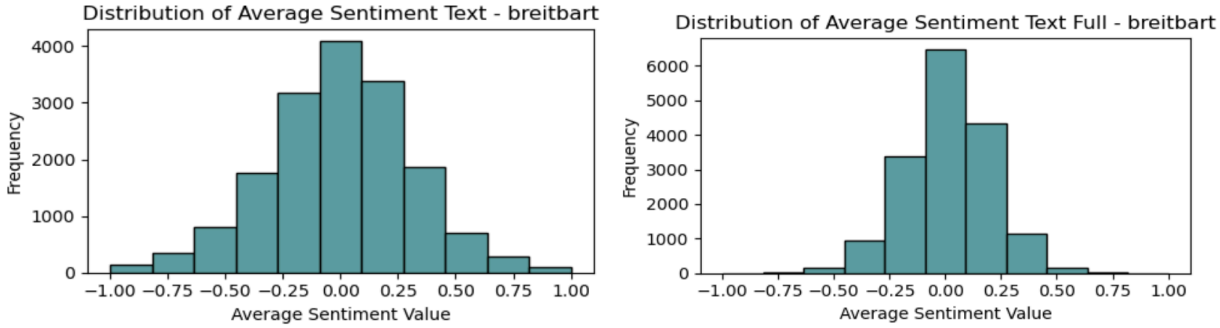
Figure 23 A-B: Sentiment scores distribution for first 512 words and full article text



We also found that while news sources have some variation in overall average article sentiment distributions, the distributions follow similar trends. When comparing the distributions for NPR above and Breitbart below, you can see the shapes of the distribution for sentiment on full article text and 512 characters of text are similar across sources. Given that news sources may talk about some topics positively and others negatively, the sentiments overall made mostly balanced sentiment distributions.

While the general shapes are similar, we do see a larger spread in the distributions for Breitbart: a larger proportion of their articles fall outside the most central “neutral” bar (0.0 average sentiment score), whereas NPR articles have thinner tails outside of the most neutral sentiment bin.

Figure 24 A-B: Sentiment scores distribution for first 512 words and full article text



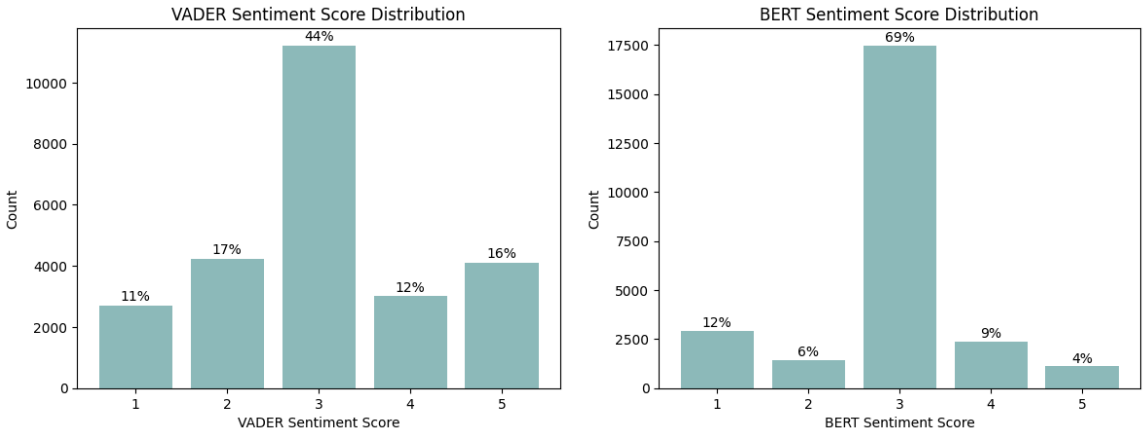
We also investigated the sentiment distributions of the different news sources based on a crude topic assignment– i.e., searching for topic keywords (like Trump, pandemic, etc.). We found these crude topic identifications to be too broad to deduce meaningful insight from the resulting

sentiment scores. For instance, we might expect news sources with political leanings to talk about Trump's twitter ban in a different way than they would talk about Trump's election – simplistically analyzing all Trump articles together does not recognize that sentiment trends may vary by subtopic.

Trends in Topic Clusters

We analyzed the topics created through BERTopic as this produced the most well defined and coherent clusters of the methods that we tried. We dropped all the articles that were coded as outliers which left 25,308 articles in our sample. Interestingly, many of the articles which could not be classified were classified as neutral. This slightly shifted the distribution of the sentiment in our sample. To better understand the difference in coverage and sentiment across topics, we first analyze the full dataset and then we focus on trends in the largest topic clusters.

Figure 25 A-B: Sentiment scores distribution of articles assigned to topic clusters



Differences in Coverage

We found that there is a difference between the amount of coverage that different sources give to different topics. We identified the top topics for each of the sources with more than 30 articles in our dataset based on the percentage of their total coverage devoted to each topic. The results here are likely skewed by the sources in our dataset which are right leaning, so it would be interesting to add more center and left leaning sources; however, we can see that there are meaningful differences between the top topics of NPR and the far right sources in our sample. NPR seems to have more coverage of topics like voting registration and the census which likely speaks to the NPR audience. As people who read primarily center and left leaning sources, it is interesting to see some of the top topics covered by Washington Times, Breitbart, and the Daily Caller are on events that we do not remember, likely because they were not extensively covered in the sources that we read. This speaks to how these differences in coverage can affect public opinion.

Figure 26: Main clusters for each news source

NPR

1. **voting_cast_registration_voter**
2. facebook_twitter_tech_google
3. **census_hansi_bureau_wang**
4. mask_masks_wear_wearing
5. relief_package_stimulus_trillion

Breitbart

1. facebook_twitter_tech_google
2. mask_masks_wear_wearing
3. **gun_guns_boebert_northam**
4. **iran_iranian_soleimani_nuclear**
5. **transgender_biological_sports_athletes**

Fox

1. facebook_twitter_tech_google
2. mask_masks_wear_wearing
3. relief_package_stimulus_trillion
4. vaccine_vaccines_pfizer_doses
5. barrett_coney_amy_confirmation

Daily Caller

1. facebook_twitter_tech_google
2. relief_package_stimulus_trillion
3. **abortion_prolife_parenthood_abortions**
4. **positive_tested_selfquarantine_tests**
5. **tara_reade_sexual_assault**

Washington Times

1. **flags_halfstaff_flag_glory**
2. **cancel_culture_rowling_jk**
3. **haaland_deb_interior_native**
4. **austin_dallas_abbott_texas**
5. **carter_page_rosenstein_surveillance**

*unique topics are in bold

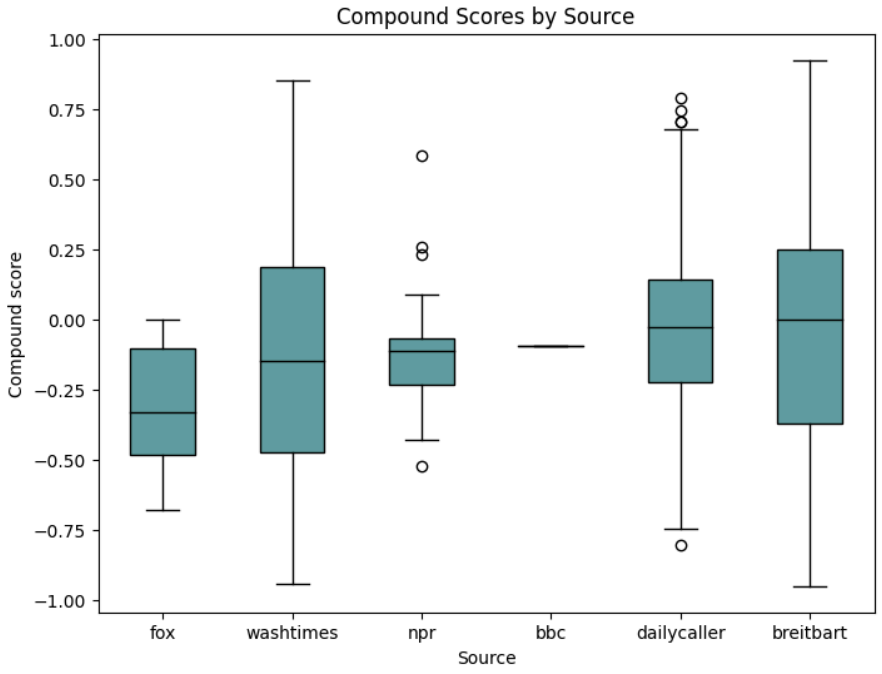
Impeachment

The fifth largest topic cluster was ‘impeachment_trial_articles_senate’ with 323 articles in the cluster. Articles in this cluster were about the articles of impeachment brought against Trump by Congress and impeachment trial. The overall distribution of sentiment for this topic follows a similar distribution to the full dataset with the majority of articles being classified as neutral.

Figure 27: Sentiment scores distribution: Impeachment topic

However, looking at the full spread of sentiment across sources using the continuous compound score as assigned by VADER we observe that while the median sentiment for most of the sources is neutral, the range of sentiment for sources varies widely. This is an important finding because while some bias is inevitable in news, news should ideally be factual and use neutral language. We find that the range of sentiment in NPR's coverage of impeachment is much smaller than Washington Times, The Daily Caller, and Breitbart. Fox also has a smaller spread but is skewed more negatively than the other sources.

Figure 28: Distribution of sentiment scores by source: Impeachment topic



To better understand what negative and positive coverage of this issue is, we pulled the text for the article with the most positive and negative sentiment scores in the cluster.

- The most positive article was from Breitbart with the following sentiment scores:
    - VADER Compound Score: 0.88, VADER: Positive, BERT: Somewhat Positive
    - *"jim jordan: senate impeachment trial to end soon, american people 'don't seem to be tuning in' during a friday interview with fox news channel's "america's newsroom," rep. jim jordan (r-oh) sounded off on the senate impeachment trial.jordan predicted the impeachment trial, which he noted nobody is tuning into, will end in the next week so congress can get back to working for the american people."[t]hese facts are so strong for the president, i feel real confident that hopefully next week we'll get this…"*

- The most negative article was also from Breitbart with the following sentiment scores:
    - VADER Compound Score: -0.95, VADER: Negative, BERT: Somewhat negative

- *“charles hurt: after impeachment farce ends, next comes 14th amendment with the result of the second impeachment charade a forgone conclusion in washington, let us now turn to the final partisan tool remaining in democrats’ torture closet: the 14th amendment.this amendment allows congress to banish from future federal office any current or former politician deemed by congress to have “engaged in insurrection.” its adoption after the civil war was designed to prevent officers or political leaders of the…”*

Both of these text samples use strong language in the positive and negative direction illustrating how strong sentiment in either direction can reflect bias in news coverage.

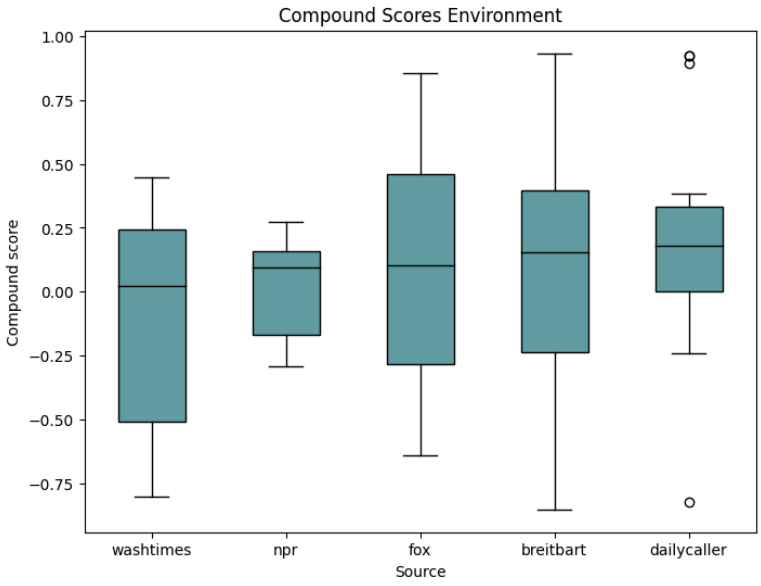
Environment

These trends appeared to be consistent across the other clusters which we looked into. The next topic we examined was the ‘climate_oil_environmental_change’ cluster with 147 articles. Articles in this cluster primarily focused on the Paris Climate Agreement, the Biden administration’s environmental agenda, and energy.

Figure 28: Sentiment scores distribution: Environment topic

Again most of the median coverage is neutral but we observe that some of the sources exhibit a much larger range in coverage. NPR again has the smallest range in the sentiment scores of their articles. Interestingly, the Daily Caller has a much smaller range in sentiment on their coverage of environmental issues, while Fox has a larger range in sentiment than their coverage of the impeachment trial.

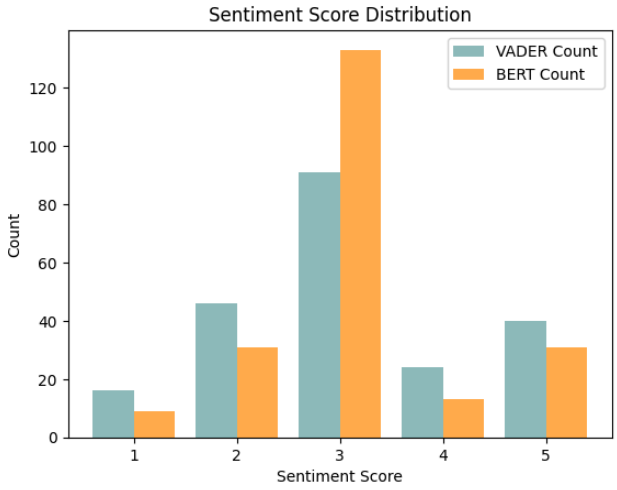
Figure 28: Distribution of sentiment scores by source: Environment topic



Alexandria Ocasio-Cortez

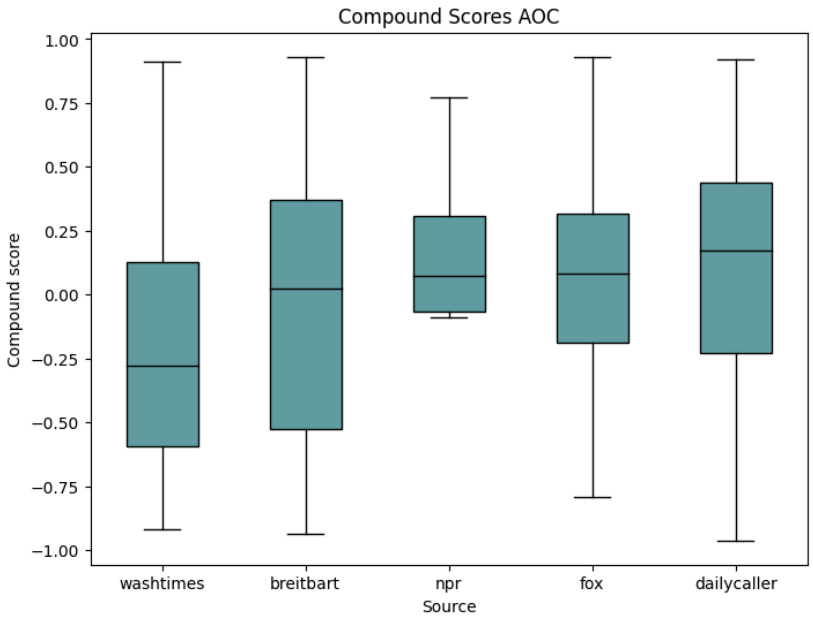
The ‘ocasiocortez_alexandria_aoc_dny’ topic cluster contained 217 articles on the representative from New York. The articles in this cluster followed similar trends to the articles in the impeachment and environment clusters.

Figure 27: Sentiment scores distribution: AOC topic



Notably all right-leaning sources display a large range in sentiment in this cluster. We also noticed that the right-leaning sources published a higher than expected number of articles on Alexandria Ocasio-Cortez, likely reflecting the polarization around her as a political figure.

Figure 28: Distribution of sentiment scores by source: AOC topic



Method difficulties:

Validation: Given that the methods we selected were both unsupervised training, it was difficult to determine what success meant since we couldn't refer to more standard metrics of accuracy, precision and F-score among other supervised training evaluation methods. Silhouette score was a slightly helpful guide to hint at where clustering configurations may be more successful, but ultimately, human-checking and manual validation were required to determine if the models were successful. This has also been echoed in the literature as a challenge.

Future Work:

There are several ways that we could expand on our work, which include expansion of methods used and expansion of the analysis dataset.

Methods:

a. **Clustering:**

For clustering we identified for each method one expansion that might improve the results. For k-means clustering, as mentioned earlier this method has some limitations regarding the assumption that data is linearly separable, and one possible way to expand this method is to use kernel k-means for which we can get a better multidimensional representation of the data to learn more complex relationships.

For hierarchical clustering methods, one expansion is to explore non-flat clustering schemes for hierarchical clustering, which is one way to also deal with more complex structure of the articles as it could estimate distances for structures that are not necessarily on a plane.

For BERTopic, one extension is to try additional Fine-Tuning for the purpose of obtaining a lower number of clusters but that stay meaningful on correctly representing a model.

In addition, there are other clustering methods that could be used to create clusters like fuzzy c-means, spectral clustering, Latent Dirichlet Allocation among others.

b. Sentiment analysis

We could extend the analysis by (i) experimenting with the sentiment analysis with different BERT pre-trained models that were fine tuned on other datasets such as product reviews and Twitter data, and (ii) label data for sentiment to be able to fine tune to news language.

While certainly challenging, as labeling could be reflective of our own biases, having labeled data for this context would be useful as we could be able to fine tune a pre-trained model that could capture the nuances of the news language and be able to capture contextual relationships to correctly address the sentiment of a news article. For example, it could be possible that now certain keywords are captured as negative while reflecting a neutral sentiment on a particular topic which could not be possible to capture on a general sentiment analysis model.

Data:

On data, the main extension we would like to implement is to run the models on the whole corpus of text that contains different news sources as this could let us get a more comprehensive idea of the topics and further explore how different editorial views or level of mainstream of a source affect sentiment. One of the main limitations of the project was that the data for 2020 contained mostly news from right-leaning news sources.

The other extension we would implement in future work would be to extend the number of years to explore the idea of how language and sentiment dynamically change over time.

Description of Effort - reflection

During the course of the project we learned both technical and theoretical implications for the different methods we tried. From a pure technical perspective it was useful further exploring some nuances of working with GPU for training models and running complex pieces of code, and working in a cloud environment.

Regarding methods the project was a good opportunity to learn both about the mathematical representation and implementation of clustering methods, particularly for k-means, hierarchical and BERTTopic clustering which we had not previously covered. In addition, for these methods

we learned about possible evaluation methods to test the performance of the clustering algorithms.

With respect to the sentiment analysis, it was positive to learn more about the specifics of the built-in libraries, their capabilities and limitations, and how we could expand from them on future extension of this project. Fine tuning the BERT model using VADER labels seemed to be only marginally effective as we expected, but we wanted to gain experience fine-tuning a pre-trained model. We initially thought it was very important to have more granular sentiment scores because we expected topics to be overwhelmingly positive or negative. However, since most of the articles were classified as neutral it may have been possible to analyze our data using a pre-trained BERT model that predicted tertiary labels. It would be interesting to compare the results of this model with the BERT model fine-tuned with VADER data.

For this project while in general many people did work on different part of the projects, we can define the main responsible for the following task as following:

- Loading data and initial analysis: Megan
- K-means clustering: JP
- Hierarchical clustering: Megan
- BERTTopic: Jackie
- Preprocessing and initial VADER analysis: Jackie
- Fine-tuning BERT: Lisette
- Analysis of news articles: Lisette (80%) and JP (20%)

Regarding the time spent on each task, our biggest challenge was learning how to run large models. We ran into significant problems with running out of GPU and RAM in Google Colab, and ultimately moved our work into the AWS environment provided.

Processing the data also proved to be time consuming as the VADER model seemed to be affected by specific characters such as ’ which were not well documented in the package as needing to be removed.

We also spent a significant amount of time reviewing clustering theory since this was a new topic for us. To better understand these methods more of our time was spent on reading than coding.

Conclusion:

The average person gets their news from only a few news sources, whether these are traditional sources like NPR or more informal forums. It seems that the information people receive is increasingly filtered through political lenses. In this project, we investigated various methods for uncovering news sources' differential sentiment around news topics. It is important for policy makers and the general public to be aware that news sources present information with sentiment, rather than presenting information with neutrality. Our project implemented various

methods of clustering and sentiment analysis to investigate these topics. We find variance in the distribution of sentiment scores across news sources, suggesting the news sources present the news with a varying level of neutrality. While further work needs to be done to refine our clustering methodology and validate the sentiment evaluations, the analysis presented here is a first step to better understanding how news sources may present news in tendentious ways.

Bibliography

Alam Falaki, A. (2021, April 23). Is it possible to do Sentiment Analysis on unlabeled data using BERT? (Feat. Vader) [Experiment]. Medium. https://nlpiation.medium.com/is-it-possible-to-do-sentiment-analysis-on-unlabeled-data-using-bert-feat-vader-experiment-357bba53768c

Afrimi, D. (2023, May 1). Text Clustering using NLP techniques. Medium. https://medium.com/@danielafrimi/text-clustering-using-nlp-techniques-c2e6b08b6e95

BERTopic. (n.d.). Retrieved from https://maartengr.github.io/BERTopic/index.html

Bisandu, Dr & Prasad, Dr & Liman, Musa. (2019). Data clustering using efficient similarity measures. Journal of Systems Science and Complexity. 22. 901-922. 10.1080/09720510.2019.1565443.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. arXiv:1607.04606. Retrieved from https://arxiv.org/abs/1607.04606

Burscher, B., Vliegenthart, R., & Vreese, C. H. de. (2016). Frames Beyond Words: Applying Cluster and Sentiment Analysis to News Coverage of the Nuclear Power Issue. Social Science Computer Review, 34(5), 530-545. https://doi.org/10.1177/0894439315596385

Chen, Q. (2020, May 10). Contextual word embeddings — Part1. Analytics Vidhya. https://medium.com/analytics-vidhya/contextual-word-embeddings-part1-20d84787c65

Dorfer, T. A. (2022, December 5). Density-Based Clustering: DBSCAN vs. HDBSCAN. Which algorithm to choose for your data. Towards Data Science. https://towardsdatascience.com/density-based-clustering-dbscan-vs-hdbscan-39e02af990c7

Gomede, E., PhD. (2023, October 11). Clustering Text in Natural Language Processing: Unveiling Patterns and Insights. Medium. https://medium.com/@evertongomede/clustering-text-in-natural-language-processing-unveiling-patterns-and-insights-8c3cd137b135

Grootendorst, M. (2021, January 6). Interactive Topic Modeling with BERTopic: An in-depth guide to topic modeling with BERTopic. https://towardsdatascience.com/interactive-topic-modeling-with-bertopic-1ea55e7d73d8

Gulcan, M. (2023, July 25). Hierarchical Clustering with Python: Basic Concepts and Application. Medium. https://medium.com/@muratgulcan/hierarchical-clustering-with-python-basic-concepts-and-application-cd5f5dc95b1f

Hdbscan. (n.d.). The hdbscan Clustering Library. https://hdbscan.readthedocs.io/en/latest/index.html

Hugging Face. (n.d.). bert-base-multilingual-uncased-sentiment. https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment

Hugging Face. (n.d.). Twitter-roBERTa-base for Sentiment Analysis. https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

Koch, K. (2020, March 25). A Friendly Introduction to Text Clustering. Towards Data Science. https://towardsdatascience.com/a-friendly-introduction-to-text-clustering-fa996bcefd04

Maklin, C. (2018, December 31). Hierarchical Agglomerative Clustering Algorithm Example In Python. Towards Data Science. https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Mansurova, M. (2023, September 8). Topics per Class Using BERTopic: How to understand the differences in texts by categories. Towards Data Science. https://towardsdatascience.com/topics-per-class-using-bertopic-252314f2640

Salton do Prado, K. (2017, April 1). How DBSCAN works and why should we use it? Towards Data Science. https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80
Text Clustering.ipynb. (n.d.). https://colab.research.google.com/github/dipanjanS/nlp_workshop_odsc19/blob/master/Module05%20-%20NLP%20Applications/Project02%20-%20Text_Clustering.ipynb#scrollTo=WycevFPGUjCe

VaderSentiment. (n.d.). Welcome to VaderSentiment's documentation. https://vadersentiment.readthedocs.io/en/latest/index.html

Yadav, A., Jha, C. K., Sharan, A., & Vaish, V. (2020). Sentiment analysis of financial news using unsupervised approach. Procedia Computer Science, 167, 589-598. https://doi.org/10.1016/j.procs.2020.03.325.

Github + README + description of who wrote what code: https://github.com/meganhmoore/LingoQuartet