Labeling News Headline Topics through Unsupervised Learning

Matthew Radovan
Stanford University
mradovan@stanford.edu

Chris Cross
Stanford University
chrisglc@stanford.edu

Sasankh Munukutla
Stanford University
sasankh@stanford.edu

https://github.com/matthewr6/news-understanding

December 12, 2019

1 Introduction

Seventy percent of Americans are fatigued by the amount of news available and as a result struggle to keep up to date with current events.[4] Moreover, feeling overwhelmed by the amount of news is more common among those who follow news *less* closely.[4] This inspired our project to make news more understandable through effective topic modelling on news headlines.

Our project also aims to identify the key current and relevant issues, allowing readers to easily understand specific topics. Presently, on critical issues like climate change, we see an overload of information. Our project aims to allow readers to get a clear understanding on these critical issues, without being inundated by unrelated articles: easing time to comprehension of current issues.

2 Related Work

Topic modelling is the task of identifying which underlying concepts are discussed within a collection of documents, and determining which topics each document is addressing. Topics are collections of words that are likely to appear in the same context. By discovering these hidden patterns of word use and connecting documents that exhibit similar patterns, a topic model has emerged as a powerful technique for finding useful structure in collections of documents.

The problem space of topic modelling and article labelling has been explored extensively by recent NLP literature over the last few years. Latent Dirichlet Allocation (LDA) is a generative Bayesian probabilistic model designed to assign a mixture of topics to an article [2]. Some works also use word embeddings to automatically label the topic of articles using vector spaces and distance metrics to evaluate topic saliency. Lau et al. used candidate labelling and a ranking algorithm to decide between predefined topics when interpreting a given article [5]. More general models, especially Google Cloud NLP [3], are significantly more powerful when identifying key topics; however, they require massive amounts of data and computing power for training.

3 Data Collection

Our core dataset is a list of strings, where each string is the combination of a news article headline and its description snippet. The news articles were obtained from the News API [6]. Each day had a set of articles published on that day, and our dataset consisted of the time period from September 8, 2019 to November

20, 2019, which totalled 8282 articles relating to politics (nonunique articles happened infrequently but were not pruned). For the reasons outlined in section 1, all articles in the dataset covered political topics.

In the remainder of this paper, referring to "article", "article headline", or "article snippet" refers to this combination of article headline and description snippet. See Figure 3 for examples of article snippets in the dataset.

4 Overview of Methodology

In order to extract keywords from clusters, we first needed to cluster our unlabeled article snippets. However, there is no definitively best method to cluster short texts. So, rather than creating a full end-to-end model encompassing preprocessing, clustering, and keyword extraction, we decided on a "mix-and-match" approach where we could create separate preprocessors, clustering models, and keyword extractors that were not dependent on each other. As a result, we were able to connect any given preprocessor, clustering model, and keyword extraction method in sequence to define pipeline and test out an approach; this allowed us to easily compare approaches without needing to create and train a new model for every approach.

5 Preprocessing

Preprocessing was performed across all article snippets in order to essentially transform the text into meaningful features that can be understood by our clustering algorithms. Aspects of the text, such as punctuation and stop words (e.g. 'and', 'but', 'it'), add noise that unnecessarily adds variability to our results. Preprocessing involves removing these elements from our training set. However, aspects of the text that may or may not be useful for our task, such as non-nouns words and words that are short, experimented upon to see if these features are relevant for our clustering algorithm. Short examples are given in each subsection; see Figure 4 for full examples.

5.1 Simple

We removed all punctuation and obtained the unique words from the article's text. However, because this approach leaves all words in, it tends to cause our models to have much higher dimensionality, leading to higher time and a potential decrease in clustering ability. All following preprocessors, with the exception of the noun tagger, expanded upon this preprocessor. The dimensionality of this preprocessor was 19410. An example input is "UK Supreme Court hears government side in vital Brexit case", corresponding to the output of "uk supreme court hears government side in vital brexit case".

5.2 Stopword removal

Next, in an attempt to lower the dimensionality of the simple preprocessor, we built a preprocessor using the *nltk* Python package that removes punctuation and stopwords (common English words) in the text. This approach retains nearly all information to maintain clustering ability, while removing irrelevant information to lower dimensionality and thus increase speed and potentially clustering ability. The dimensionality of this preprocessor was 19275. The example input described in the Simple preprocessor gives the output of "uk supreme court hears government side vital brexit case".

5.3 Noun tagging

Continuing to keep only relevant information, we used *spaCy* (a natural language parsing library for Python) to automatically identify all nouns and prune article snippets to only include those nouns. The motivation behind this was that nouns are always the subject of a sentence, so using nouns only should

create strong clusters based on the articles' subjects. We also lemmanize across each of the words in order to focus on just the roots of every word - for instance, a word like "walking" is shortened into "walk". This further reduces the variability in our dataset without really losing any valuable information. The dimensionality of this preprocessor was 1237. The example input described in the Simple preprocessor gives the output of "UK Supreme Court government side Brexit case".

5.4 Word length

We then built a preprocessor that removes all words below a certain length. This one was motivated by the slow speed of the noun preprocessor. By removing all words with length lower than the median, we created a preprocessor that inherently kept more unique, and likely more important words, without having to use another model to tag nouns in a word. In the case of our dataset, the median word length was 5, so we removed all words with length less than 5. The dimensionality of this preprocessor was 14395. The example input described in the Simple preprocessor gives the output of "supreme government brexit".

6 Clustering Models

After featurizing each of the article headlines, we then fed these features into our clustering algorithms in order to group articles - presumably according to common topics. Clustering algorithms are defined as algorithms that group a set of data points such that points within each cluster are similar to each other and points from different clusters are dissimilar. K-Means is the fundamental clustering algorithm - also taught in class - that mainly focuses on clustering numerical data points. The Latent Dirichlet Allocation model (LDA), on the other hand, is a clustering algorithm that is more suited to textual data - and is also the predominant method used today for clustering of texts by topic. The biterm model (BTM) and topic keyword model (TKM) are algorithms developed in the past few years that are similarly used for textual data, but seek to solve many of the problems that are inherent to the LDA.

6.1 K-means

The K-Means model is similar to the one in lecture, using a slightly modified version known as K-Means++ which results in faster convergence. It uses a simple count vectorizer to create a sparse feature vector where each word maps to the count of that word in the article title and outputs an integer determining which cluster a given input is assigned to. Here is the general approach for K-Means:
1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.

2. Repeat until convergence:

$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$$

$$\text{For each } j, \text{ set } \mu_j := \frac{\sum_{i=1}^{m} 1\left\{c^{(i)} = j\right\} x^{(i)}}{\sum_{i=1}^{m} 1\left\{c^{(i)} = j\right\}}$$

K-Means++ [1] is faster than K-Means, as rather than randomly initializing the centroids, the centroids are initialized such that they are well-separated, thus reducing the time for convergence. This is achieved by initializing the clusters using a weighted probability distribution of the existing data points, where data points that are further away from the current set of data points used as initial centroids have a higher probability of being chosen as the next centroid.

One consideration for this model is that it may suffer from the curse of dimensionality, as the dimensionality of preprocessed data ranged from 12377 to 19410, so although the model is not slow, we qualitatively as well as quantitatively analyzed the model's output to see if it makes sense. Another is that the number of clusters must be manually set, which we did through cluster score index analyses as well as qualitative analysis of article snippet clusters and cluster visualizations.

6.2 Latent Dirichlet Allocation

Instead of assigning a given article to a specific model based on sparse vector clustering - as in the case of the K-Means, the Latent Dirichlet Allocation model instead represents each article as a mixture of topics. The LDA[2] takes as input the words that are taken from a given article and tries to generate a distribution of topics from these words. The LDA then picks a given generated topic and generates a distribution of words associated with this given topic. From there, it then backtracks by using a given article's words and tries to find the topic distribution that best fits that article's distribution of words. Thus, the LDA would then output the most likely topic for a specific article given the keywords of an article.

Mathematically speaking, the model can be generally described as follows: for each article n in the corpus, we choose an $\theta \sim Dir(\alpha)$. Then, for each word w_n, we assign it to a topic $z_n \sim Multinomial(\theta)$ and choose another word w_n from $p(w_n|z_n, \beta)$ - which is a multinomial probability conditioned over topic z_n. After all of this, we can generate a topic mixture distribution across a set of topics for an article: $p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p\left(z_n|\theta\right) p\left(w_n|z_n, \beta\right)$. As one may notice, we are generating a joint distribution for each articles - which leads to our major problem regarding the LDA. The LDA does not generalize well to short documents - in this case, article snippets. Given that it's a statistical inference model that relies heavily on a large amount of data with dense features in order to fully explore the feature space and develop a fully comprehensive distribution of topics for each article, the LDA's accuracy becomes dubious when such articles are sparse.

6.3 Biterm Model

The Biterm model[9] tries to solve many of the problems that the LDA encounters with regard to short documents. Essentially, the Biterm model uses a significantly less complex model in order to better deal with inherently sparse data points. Instead of inferring the mixture of topics over each of the documents, it models the mixture of topics over the entire word corpus. Additionally, instead of inferring a topic from a single word, as in the case of the LDA, the Biterm model infers topics from biterms (which essentially represent the co-occurrence patterns of a range of word pairs in the corpus). Similar to the general approach of N-grams, this method reduces the noise of analyzing individual words from the case of the LDA by incorporating more context. Thus, the Biterm model is a simpler model that still captures the nuanced relationships between topics and words in a short sequence of text.

The generative process of the BTM is as follows: similarly to the LDA, we draw a word distribution $\phi_z \sim Dir(\beta)$ for each given topic z, as well as a topic distribution $\theta \sim Dir(\alpha)$ for the entire collection of articles. Then, for each biterm b in the biterm set B, we draw a topic assignment $z \sim Multi(\theta)$ and two words, w_i and w_j drawn from $Multi(\phi_z)$. Then, we can represent the joint probability for a biterm $b = (w_i, w_j)$ as $\sum_z \theta_z \phi_{i|z} \phi_{j|z}$, with the likelihood for the entire corpus being $P(B) = \prod_{(i,j)} \sum_z \theta_z \phi_{i|z} \phi_{j|z}$. From this, we can maximize over likelihoods to determine the optimal topic for a given article. As one can see, the probability distributions exist over the entire corpus rather than being constrained to individual articles. However, the main drawback of this is that we need to perform Gibbs Sampling to optimize over parameters θ and ϕ, drastically increasing the training time compared to other clustering algorithms.

6.4 Topic Keyword Model

The Topic Keyword Model (TKM)[8] also tries to solve problems that the LDA encounters in short documents. Rather than using a bag-of-words model as in the LDA, the TKM factors in the probability of a given word's location in the document. And, instead of Gibbs sampling in the biterm model, the TKM uses expectation-maximization to find the parameters to maximize the probability of the given dataset across all possible topics - that is, maximizing $\prod_d \prod_{i\in[0,|d|-1]} p(d,w,i) = \prod_d \prod_{i\in[0,|d|-1]} p(d)\cdot p(i|d)\cdot p\left(w_i = w|d,i\right)$ where d is a document, i is a word position, and w_i is a word in position i. Then, the actual probability of a topic given a document is based on the probabilities of words in their locations given the topic. And, we defined the cluster assignment to be the topic with the highest probability.

Another core part of TKM is the topic keyword extraction. However, although faster than the biterm model due to using EM rather than Gibbs sampling for training, the TKM was found to cluster poorly regardless of preprocessing method and number of clusters. As a result, no further analysis was performed.

7 Keyword Extraction Methods

In line with our goal to extract keywords that are descriptive of their clusters, we defined "descriptive" keywords to be words that are both relevant and unique to a cluster. That is, a keyword should not be common across all clusters, but should be common within the cluster it is associated with. We created a set of clustering model agnostic keyword extraction methods to be run on any article cluster assignments given in order to continue with our mix-and-match approach.

7.1 Count

As an initial keyword extractor, we created a simple count-based approach. In each cluster, words were ranked based on the number of times they appeared in the cluster, and the top 25 words were chosen to represent the cluster. This approach was chosen first because it was the simplest logical way to extract potentially useful words from a cluster. Due to its simplicity, it only considers half of our definition of descriptive keywords: it finds words likely to be relevant within its cluster, but does not consider the distribution of words across clusters.

7.2 TF-IDF

Compared to the count-based measures, the Term Frequency–Inverse Document Frequency score (TF-IDF) [7] focuses on the relevance of certain words, not just their frequency. Word counts are replaced with TF-IDF scores across the entire corpus. First of all, TF-IDF measures the number of times that words appear in a given document (term frequency). However, since words such as "and" or "the" appear frequently in all documents, those must be systematically discounted. Therefore, we normalize these frequency scores by how often they appear in other documents (inverse-document frequency). The more documents a word appears in, the less valuable that word is as a signal to differentiate between documents.Thus, we are left with only words that are both frequent AND distinctive for a given document. Each word's TF-IDF relevance is a normalized data format that also adds up to one.

7.3 Pseudo TF-IDF

Instead of using the TF-IDF, we created a custom keyword extractor based on the motivation of TF-IDF - scale up a word's importance based on its frequency within the cluster, and then scale down the word's importance based on its frequency in other clusters. In this extractor, we ranked the keywords in a given cluster c by defining the word rank in a cluster r_{wc} as

$$r_{wc} = p_{wc} * \frac{p_{wc}}{\sum_{i=1}^{|C|} p_{wi}}$$

where p_{wc} is the normalized freqency of a word w in a cluster c and C is the set of clusters. The initial p_{wc} term is intended to factor in relevance of the word within its cluster, and the $\frac{p_{wc}}{\sum_{i=1}^{|C|} p_{wi}}$ term is to tcompare the relevance of the word within its cluster to the relevance of the words across all clusters. Then, given these word rankings, we took the top 25 words in each cluster as the cluster keywords.

7.4 Logistic Regression Coefficients

Initially, our idea was to train a multinomial Naive Bayes classifier on the cluster assignments and use the probabilities of features given classes $P(x_i|y)$ to find the best possible combination of $N = 25$ words (25 for consistency with other extractors) that would give the highest probability of a document. That is, given a set of words $W, |W| = 25$ and cluster y, find $\arg\max_W \prod_{i=1}^{|W|} P(W_i|y)$. However, given the high dimensionality of our data, we discovered that the optimization process took too long.

Drawing on the idea of training models and using the learned parameters to infer keywords, we implemented a logistic regression model, similar to the logistic regression algorithm we learned in class. We implemented a multinomial version with cluster assignments as labels k, parameters θ, and input vector x. The logistic function is

$$\phi = p(y = k|x; \theta) = \frac{1}{1 + \exp(-\theta^T x)} = g\left(\theta^T x\right)$$

After training, the coefficients of θ indicated which words are most salient in determining the correct cluster assignment, as they would have the greatest influence on predicting a cluster. Thus, for each cluster, we selected the 25 features with the highest values (the highest θ coefficients) as the method to extract the keywords. 25 was chosen for consistency with other keyword extraction methods.

7.5 Centroids

All of the keyword extraction methods discussed prior iterated over clusters and selected relevant keywords from each cluster. Instead, our centroid-based keyword extractor was motivated by reversing the order of this process: instead of selecting clusters and then selecting keywords from clusters, we built an extraction method that reversed the order, assigning keywords to clusters only after selecting all "interesting" keywords.

Specifically, all articles were converted to vectors through a bag-of-words method and a cluster's centroid was defined to be the average of the cluster's vectorized articles. Then, "interesting" keywords were defined as keywords corresponding to the dimensions of greatest standard deviation over the distribution over centroids. That is, the nth selected word w_n was defined as

$$w_n = \underset{i:1 \le i \le |W|, i \notin \{w_m : m < n\}}{\arg\max} \sigma_w$$

where W is the set of all features (words) and σ_w is the standard deviation of the centroids in the dimension of w.

Because 10 clusters was determined to be the optimal number of clusters and previous keyword extractors took 25 words per cluster, we chose 250 keywords for consistency.

After keywords were extracted according to this method, relevance of a given keyword to a given cluster was calculated simply by calculating the normalized frequency of the keyword in the cluster. That is, the cluster assignment c_w of word w is defined as

$$c_w = \arg\max_C \frac{1}{|C|} \sum_{i=1}^{|C|} C_{iw}$$

where C is a cluster as defined by a set of processed article snippets, and C_{iw} is the number of times word w occurs in the ith article snippet of C.

8 Results

Because determining the best set of "descriptive keywords" for a given topic is inherently subjective, a major portion of our analysis was qualitative - analyzing cluster contents and their relation to the keywords. Then, in addition, we came up with a metric to help concretely define how descriptive the cluster keywords were with regard to their cluster contents.

8.1 Clustering

Because we considered four candidate clustering models, we had to determine which model would produce the best article snippet clusters. And, in all models, the number of clusters was a parameter that had to be manually chosen.

To first choose the best model out of the candidate models, we created scatterplots of the first three principal components of the data and colored the points by cluster label (see Figure 7 for sample plots). The biterm model operated on the scale of tens of hours on an Intel Xeon E5520 due to its Gibbs sampling process, so we decided it wasn't a feasible model. Of the three remaining models (K-means, LDA, and TKM), both the LDA and TKM models produced clusters that were extremely unintuitive both through looking at the graphs and the individual articles in their clusters. So, the K-means model was selected as the best model for clustering the article snippets. See Figure 3 for examples of articles in clusters created by the models.

8.1.1 Quantitative Analysis

After deciding on K-Means, the hyperparameter we had to tune is the number of clusters k. After empirical tuning, $k = 10$ was found as the best choice for number of clusters.

We utilized the Davies-Bouldin Index, which is a metric for assessing the performance of clustering, taking into account tightness of clusters as well as the difference between clusters, with a lower score indicating better clustering. For $k = 10$ we had an average Davies-Bouldin score across all preprocessors of **4.91**, compared to 4.96 for 8 clusters and 4.89 for 12 clusters. See Figure 1,9 for graphs of cluster scores. Although $k = 12$ clusters had a marginally lower score, we ultimately chose 10 clusters, as after qualitatively assessing the clusters, we found $k = 10$ to have more tight and unique clustering (discussed later). This is later confirmed in our accuracy analysis.

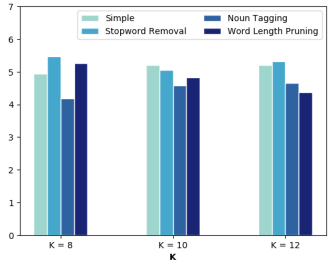


Figure 1: Davies-Bouldin index for K-means models across multiple values of K and multiple preprocessors

Another interesting observation is that good accuracy does not necessarily imply good clustering or vice versa. For example, the centroid preprocessor had the highest accuracy as wee see in the Cluster Keywords

analysis in the next section, whereas the noun tagging preprocessor has the best clustering (lowert Davies-Bouldin Index of **4.57**) for $k = 10$ clusters. Similarly, the stopword removal preproccesor had the lowest accuracy as wee see in the Cluster Keywords analysis in the next section, whereas the simple preprocessor had the worst clustering (higest Davies-Bouldin index of **5.21**).

We also reran all these results using $k = 8$ and $k = 12$ clusters for completeness. Another fascinating insight is that the best preprocessor in terms of lowest Davies-Bouldin Index varied by cluster size. For example, for 12 clusters the word length preprocessor had the lowest Davies-Bouldin Index, whereas, for 8 and 10 clusters the noun tagging preprocessor had the lowest Davies-Bouldin Index.

8.1.2 Qualitative Analysis

Analyzing the best clustering using the noun tagging preprocessor, multiple interesting examples of articles in the same cluster were found. For example, articles related to Brexit were put in the same cluster: "pm john boris johnson u.k. supreme court case court johnson law parliament" and "uk election boris johnson corner brexit misstep" (snippets shown post-preprocessing).

Another interesting example is "trump bidens ukraine president trump talk impeachment report ukraine investigation joe biden" and "trump nixon respect president transcript call president dirt rival joe biden breath michael dantonio trump richard nixon", two articles in the same cluster (again using the noun tagging preprocessor), which relate to the ongoing Trump–Ukraine scandal.

On the other hand, we see unrelated articles from the worst clustering (using the simple preprocessor) in the same cluster. For example, "pm vs pm john major argues against boris johnson in uk supreme court case the court is considering whether johnson broke the law in suspending parliament", and "what happened to kamala harris if you asked any smart democrats on june 28 who they believed in their hearts of hearts would be the 2020 democratic presidential nominee 9 in 10 would have said kamala harris", were two articles in the same cluster, which are unrelated (one is about Brexit and the other is about the democratic presidential nominee Kamala Harris).

Another example, again using the simple preprocessor, is "gig economy in crosshairs after decade of free-wheeling growth the wall street journal gig economy in crosshairs after decade of freewheeling growth the wall street journal uber and lyft face 2 big threats to their business model after new california law yahoo finance california passes gigeconomy bill aimed at wages at uber lyft and other fir" and "the latest jury pool asked how excops job affects views prosecutors have begun questioning potential jurors in the murder trial of a white former police officer who fatally shot an unarmed black neighbor in his own home", which are two articles in the same cluster that are unrelated (one is about regulations for ride-sharing services and the other is about police brutality).

Looking at articles within the actual clusters allows us to build intuition and understand how our various preprocessors work and shows that our qualitative interpretations of the optimal preprocessing and clustering methods are in line with our quantitative results. See also Figure 5 for a summary of these examples.

8.2 Cluster Keywords

8.2.1 Quantitative Analysis

In order to create a quantitative measure of how descriptive the extracted cluster keywords were, we decided to look at the ability of a simple logistic regression classifier to predict the cluster from stripped-down article snippets containing only keywords. We trained a logistic regression classifier using the preprocessed article snippets as input and cluster labels from the clustering model as labels. Then, we created a derived dataset from the training dataset by using just the keywords (removing all words that are not keywords). For instance, with an input of "action thunberg teenage climate activist testifies before congress" and a set

of keywords {"protests", "climate", "politics", "coverage"}, the derived article snippet would simply be "climate". All keywords across all clusters were included in the article, because if a keyword is descriptive, it should be strongly linked to one and only one cluster label.

We reran the logistic regression classifier on this derived dataset to measure accuracy (called re-classification accuracy in the remainder of this discussion). This accuracy is a metric to measure how descriptive the keywords are of their specific cluster.

Looking at the accuracy graphs (Figure 2, 10), the word length preprocessor was the best preprocessor in terms of re-classification accuracy. In addition, the centroid extractor coupled with the word length preprocessor had the best overall accuracy of **99.42%**. Generally, the centroid extractor was the best extractor, but the best extractor did vary by preprocessor. For example, the logistic regression extractor was best on the stopword removal preprocessor. Intuitively, the word length preprocessor being the overall best makes sense, as the length of words are likely very salient for news headlines: longer words tend to help differentiate topics as they are more significant. And, it's reasonable that the centroid extractor is best, as it first chooses candidate words by finding words that distinguish the clusters the most (as a sort of PCA that aligns along the original dimensions) rather than choosing words cluster-by-cluster, which lessens the chance of choosing words that give high cluster separability.

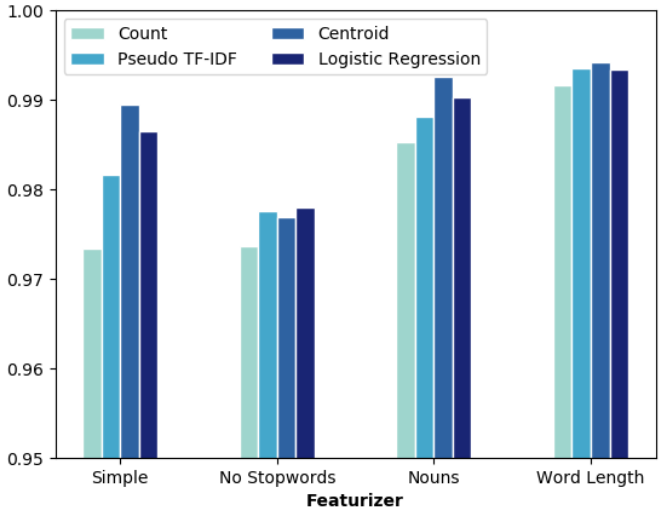


Figure 2: Accuracy of logistic regression classifier on keyword-based derived dataset across different preprocessors and keyword extractors.

We also reran all these results using $k = 8$ and $k = 12$ clusters for completeness. It was interesting to see that $k = 10$, our choice for the number of clusters, had slightly better accuracy on average on the whole across all extractors and preprocessors of **98.54%** compared to 98.47% for 8 clusters and 97.93% for 10 clusters, confirming the validity of choosing 10 clusters. Another interesting observation is that variability of accuracy performance of preprocessors across different cluster sizes. For example, for 8 clusters Nouns was the best preprocessor and for 12 clusters Simple was the worst preprocessor, whereas for 10 clusters, the stopword removal preprocessor was the worst preprocessor.

8.2.2 Qualitative Analysis

Analyzing the most accurate clusters (world length preprocessor + centroid extractor) we see some interesting examples of keywords for clusters. For example, {'democratic', 'presidential', 'candidates', 'bloomberg', 'kamala', 'andrew', 'gabbard', 'buttigieg'} were the keywords for one of the clusters. It is easy to see that this is with regard to democratic candidates for the upcoming US presidential election. Another interesting example is {'donald', 'ukraine', 'volodymyr', 'whistleblower', 'zelensky'} (again from the word length preprocessor and centroid extractor), which clearly relates to the ongoing Trump–Ukraine scandal.

On the other hand, we see vague keywords from the least accurate clusters (simple preprocessor + count extractor). For example, {'news', 'world', 'breaking', 'exclusive',' politics'} were the keywords for one cluster, which doesn't give us any intuition for the topics in it. Another bad example, again from the simple preprocessor and count extractor, is {'to', 'the', 'a', 'in', 'trump', 'president'}, which is very general to most current political topics.

From our quantitative analysis, with examples given here (and summarized in Figure 6, we observed that our qualitative decision for the best keyword extraction method was in line with our quantitative results.

8.3 Error Analysis

As we ran through our experiments, we immediately notice clear differences in performances between models. Unsurprisingly, the LDA performed poorly (which is expected, given what we know about the model's problems with clustering short texts). Surprisingly, the K-means model worked considerably better than the TKM - which may be due to the fact that the short length of headlines made it possible for the K-means to cluster articles. We also recognize that the BTM model was far too slow in our experimental setting. Namely, given that we had several thousand examples, the Gibbs Sampling protocol took close to 15 hours on an Intel Xeon E5520 (by comparison, the LDA, K-Means, and TKM each took around a minute to compute on an Intel i7-7700HQ). Another caveat is the improvement of using the word length feature extractor over more complex feature extractors that we expected to work better. This may be due to the fact that - at least in the context of news articles - larger words will better encompass the nuances of an article. For instance, while the word "democrat" may be a good all-around guess for an article's topic, "whistleblower" would be more emblematic of the article's content.

Throughout our experimental paradigm, we encounter a variety of challenges and potential sources of error. One bottleneck exists in the dimensionality of our dataset. Due to the inherent complexity of textual data, the feature space becomes exponentially large (hence, why topic modelling has been dominated by the LDA in recent years). While we eventually recognize that the nature of our problem does constrain the problem a bit (by clustering short headlines rather than large documents), the short nature of these texts adds a significant amount of variability to the text features that we extract.

8.4 Summary

Through analyzing various combinations in our mix-and-match pipeline of preprocessors, clustering models, and keyword extractors, we determined that the **median word length preprocessing → 10-cluster k-means → centroid-based keyword extraction** combination worked best. For our specific case of politics-related news articles, the median word length preprocessor likely worked best due to the the observed tendency of important words to be longer in the article snippets.

8.5 Next Steps

One key hand-tuned parameter was the number of clusters in the clustering models, as it was determined through both quantitative and qualitative analysis of the model's cluster assignments. And, the number of clusters may change based on the current topics in the news at any given time. Because the topic keywords are inherently dependent on the cluster, the keywords are also strongly dependent on the number of clusters. So, one major next step is to determine a method to mathematically determine the optimal number of clusters while keeping the cluster assignments reasonable when qualitatively analyzing them.

Another interesting problems to solve is to consider how keywords differ by source and to calibrate our models accordingly. For example, a neutral source may use less sensational terms compared to far-left or far-right news source.

Appendix

Acknowledgements

We would like to express our gratitude for the open source NewsAPI and scikit-learn, which enabled us easily to collect the data for our project and implement our models. We would also like to give special thanks to Chuma Kabaghe, our mentor for the project, for her very helpful feedback and support throughout the entire project.

References

[1] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1027–1035. http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022. http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf.

[3] Google Cloud. *Google Cloud Natural Language*. https://cloud.google.com/natural-language/.

[4] Jeffrey Gottfried and Michael Barthel. 2018. Almost seven-in-ten Americans have news fatigue, more among Republicans. (5 July 2018). https://www.pewresearch.org/fact-tank/2018/06/05/almost-seven-in-ten-americans-have-news-fatigue-more-among-republicans/.

[5] Jey Han Lau, David Newman, Sarvnaz Karimi, and Timothy Baldwin. 2010. Best topic word selection for topic labelling. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, 605–613. https://arxiv.org/pdf/1612.05340.pdf.

[6] NewsAPI. *News API - A JSON API for live news and blog articles*. https://newsapi.org/.

[7] Juan Ramos and others. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. Piscataway, NJ, 133–142. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf.

[8] Johannes Schneider and Michail Vlachos. 2018. Topic modeling based on keywords and context. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 369–377. https://arxiv.org/pdf/1710.02650.pdf.

[9] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 1445–1456. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.4032&rep=rep1&type=pdf.

Figures

Trump administration officials tell Senate about Big Tech antitrust probes The Justice Department's antitrust division chief, Makan Delrahim, said Tuesday that its probes of big technology companies like Alphabet's Google were a "priority" that could result in either "law enforcement or policy options as solutions."
The only way to deal with Trump the whiner Michael D'Antonio writes that whether or not the House decides to proceed with the impeachment process, Trump is going to play the victim, so it's better for Democrats to proceed with a serious approach to Trump's obstruction of justice than to give the Presi...
China's TikTok social media app has captured the NFL, but not Hong Kong protesters - The Washington Post Social media has been a quintessentially American journey rooted in ideas about creative expression and freedom of speech. But the rise of the Chinese video app TikTok could signal an end to Western dominance of the Internet's megaphones — and an ideological ...
Buttigieg calls Warren 'evasive' on Medicare for all and Pete Buttigieg on Thursday called Elizabeth Warren ëvasiveër for not saying whether her health care plan would raise middle class taxes.Warren, a candidate for the Democratic party's presidential nomination, didn't...
Tension with Iran rising as US backs Saudi 'right to defend itself'; Warning sign for Biden in new poll

Figure 3: Examples of article snippets in dataset.

Original	UK Supreme Court hears government side in vital Brexit case LONDON (AP) — The British government was back at the country's Supreme Court on Wednesday, arguing that Prime Minister Boris Johnson's decision to suspend Parliament just weeks before the country is set to leave the European Union was neither improper...
Simple	uk supreme court hears government side in vital brexit case london ap the british government was back at the countrys supreme court on wednesday arguing that prime minister boris johnsons decision to suspend parliament just weeks before the country is set to leave the european union was neither improper
Stopword removal	uk supreme court hears government side vital brexit case london ap british government back countrys supreme court wednesday arguing prime minister boris johnsons decision suspend parliament weeks country set leave european union neither improper
Noun tagging	UK Supreme Court government side Brexit case LONDON AP government country Supreme Court Wednesday Prime Minister Boris Johnson decision Parliament week country European Union
Word length pruning	supreme government brexit london british government countrys supreme wednesday arguing minister johnsons decision suspend parliament before country european neither improper

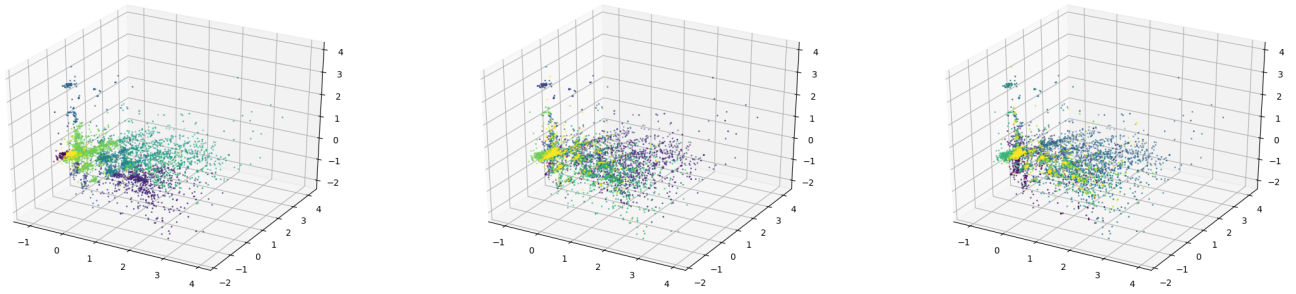
Figure 4: Example of preprocessor outputs.

Preprocessor	Cluster examples
Noun tagging	pm pm john major boris johnson u.k. supreme court case court johnson law parliament uk election boris johnson corner brexit misstep opposition plan johnson man loss uk country time
Noun tagging	trump bidens ukraine president trump talk impeachment report ukraine investigation joe biden trump nixon respect president transcript call president dirt rival joe biden breath michael d'antonio trump richard nixon
Simple	pm vs pm john major argues against boris johnson in uk supreme court case the court is considering whether johnson broke the law in suspending parliament what happened to kamala harris if you asked any smart democrats on june 28 who they believed in their hearts of hearts would be the 2020 democratic presidential nominee 9 in 10 would have said kamala harris
Simple	the latest jury pool asked how excops job affects views prosecutors have begun questioning potential jurors in the murder trial of a white former police officer who fatally shot an unarmed black neighbor in his own home gig economy in crosshairs after decade of freewheeling growth the wall street journal gig economy in crosshairs after decade of freewheeling growth the wall street journal uber and lyft face 2 big threats to their business model after new california law yahoo finance california passes gigeconomy bill aimed at wages at uber lyft and other fir

Figure 5: Examples of articles grouped by cluster, with preprocessor noted. Notice how the noun tagging preprocessor has clusters with strongly related articles, while the articles under the simple preprocessor have much looser relations. The 10-cluster k-means model was used.

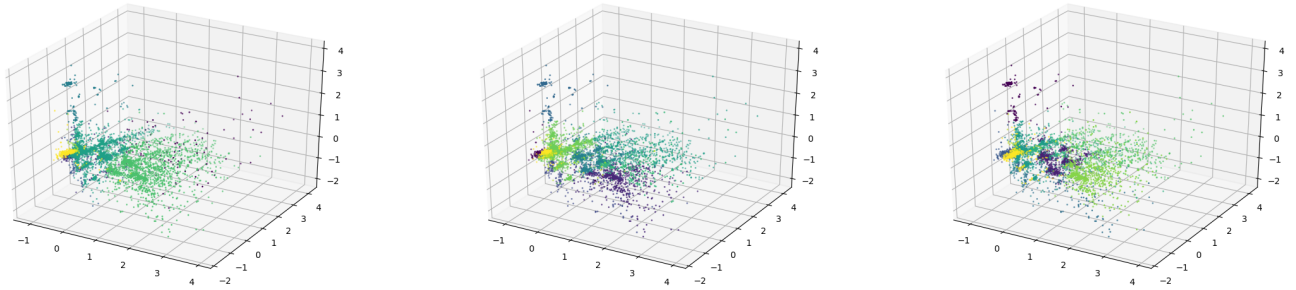
Preprocessor + Extractor	Cluster Keywords
Simple + Count	to, the, a, in, and, of, on, for, trump, is, that, with, president, as, from, his, us, new, he, at, an, be, has, election, it news, and, the, at, world, breaking, top, coverage, national, get, broadcast, video, exclusive, interviews, find, online, abc, for, in, to, of, latest, politics, us, today
Word length + Centroid	democratic, debate, presidential, candidates, candidate, primary, issues, bloomberg, houston, michael, kamala, current, orourke, andrew, nomination, hillary, leading, fourth, patrick, october, gabbard, partys, nights, buttigieg president, donald, trumps, ukraine, ukrainian, former, ukraines, volodymyr, meeting, carter, romney, military, investigate, personal, whistleblower, concerns, according, office, zelensky, attorney, hunter, george, allies, analysis, interview

Figure 6: Examples of keywords extracted from 10-cluster k-means model given preprocessing method and keyword extraction method. Notice the discrepancy in descriptiveness of the keywords in the first versus the second row of the table.



K-means LDA TKM

Figure 7: Comparison of K-means, LDA, and TKM clustering ability on the noun tagging preprocessed data using 10 clusters. Plots were created by taking the first three principal components of the data and coloring the scattered points by cluster label.



K = 8 K = 10 K = 12

Figure 8: Examples of different number of clusters K using the noun tagging preprocessor and K-means clustering.

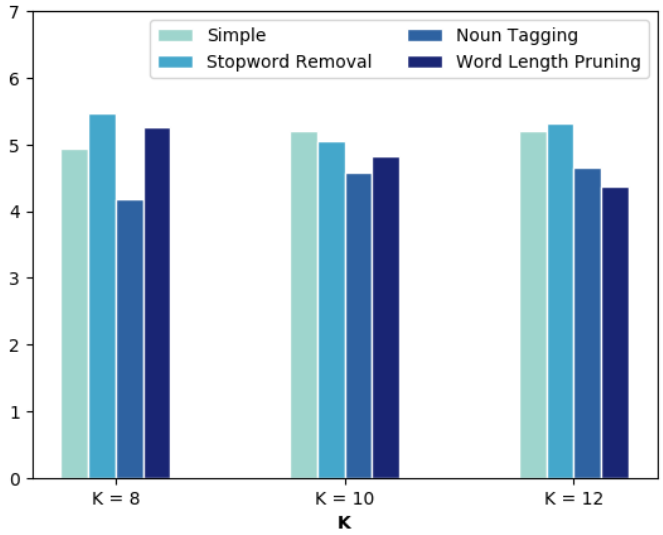
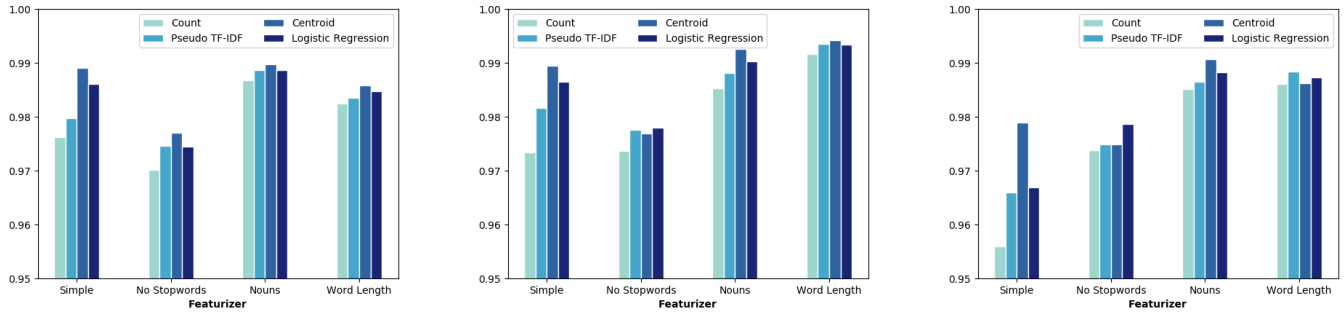


Figure 9: Davies-Bouldin index for K-means models across multiple values of K and multiple preprocessors



K = 8 K = 10 K = 12

Figure 10: Examples of logistic regression-based analysis of keyword extractors using different number of clusters K and different preprocessors. "Featurizer" refers to preprocessor.