ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis [BSc2 Econometrics/Economics]

'What's in the news?'

Name student: Eva van Rooijen

Student ID number: 443081

Supervisor: Velden, M. van de

Second assessor: Cavicchia, C.

Date final version: 5-7-2020

BSc2 Thesis - What's in the news?

Eva van Rooijen

July 2020

Abstract

An important step in cluster analysis, is the selection of the number of clusters. Several measures have been introduced to aid the process of choosing the number of clusters. Wang (2010) introduced cluster stability as a method for the consistent selection of the number of clusters. The performance of this method was shown for low dimensional simulated datasets. However, cluster analysis also has relevant applications for high dimensional data. Topic modeling for example, aims to group text documents based on latent semantic structures. This paper extends the cluster selection method via crossvalidation to the clustering of a corpus of news articles. We show that cluster selection via crossvalidation results in 22 topics. We interpret these topics and compare our results to another common topic selection method, topic coherence. We conclude that selecting the number of topics via topic stability results in interpretable, stable and coherent topics.

Cluster analysis, Cluster Selection, Topic Modeling, LDA

Contents

1 Introduction

Unsupervised machine learning methods such as dimensionality reduction and clustering take a high-dimensional data set (X) and transform it either to fewer dimensions or cluster the observations. When X would be a set of n observations in m dimensions, dimensionality reduction aims to reduce m feature while clustering reduces the n observations into a number of clusters (K).

A common problem in clustering observations is how to select that number of clusters, K. The figure below shows an example for three clusters. From the original plot on the left it is not clear that there are three clusters. For example, one could also claim there are two clusters. First, the dense set with a centre in (x1, x2) = (3, 0). All other observations form a more sparse second cluster.

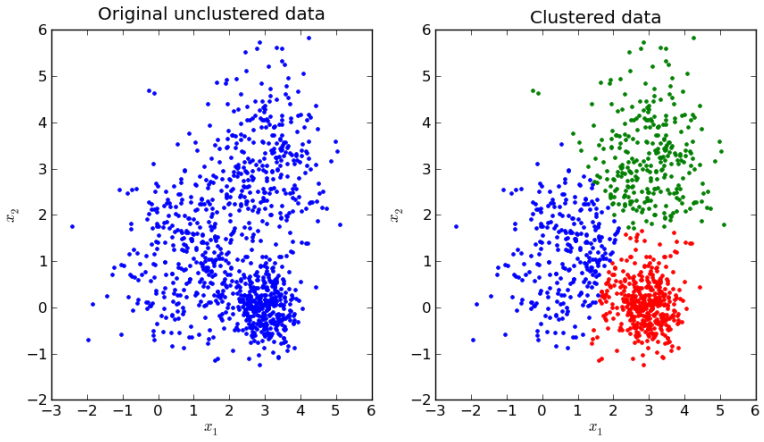

Figure 1: Clustering

'Hard' clustering methods assign a single label to every observation. 'Soft' clustering yields a probability distribution over the K possible clusters instead. For example, consider the clustering of customers into market segments based on customer data. The cluster mapping takes data on a particular customer such as demographic data (age, gender, country) and sales data on historical purchases. After training, it presents us with either a single market segment (hard) or a range of probabilities such as 20% to fit in segment 1, 65% probability to be in segment 2 and a 15% probability to fit in segment 3.

These probabilistic methods can be trained following the Expectation-Maximization algorithm. This algorithm is described by two steps. In the Expectation step, we find the expected cluster for every observation based on the model parameters. Second, in the Maximization step, model parameters are updated by maximizing the likelihood of the function given these labels.

This algorithm can be applied to train several clustering models. For example, to the popular k-means model based on distances. In the expectation step, cluster labels are assigned by minimizing the distance between the observation and cluster centroid. After assigning labels, cluster centroids are updated given the results from the expectation step.

Regardless of the clustering algorithm, selecting the number of clusters has been an important problem in cluster analysis. This is mainly because there is no single objective measure to evaluate model performance. In supervised learning, one could measure the sum of squared errors (SSE) as the sum of squared differences between the prediction and the actual label. For example when predicting sales, the real value of (historical) sales is known. Trained models can be evaluated by this measure of performance.

Crossvalidation is a method to calculate this performance measure on previously unseen data. The performance of a model should not be evaluated on the same data it was trained on to avoid the risk of overfitting (Hawkins, 2004). With crossvalidation, the sample is usually split in three subsamples. One of the subsamples is used to fit the model parameters. A second subsample could be used to fit hyperparameters such as the regularization parameter. Finally, a third subsample is used to evaluate the model performance.

Wang (2010) proposes to use crossvalidation for consistent cluster selection via the maximisation of cluster stability. Crossvalidation is used to calculate the stability of cluster mappings by comparing the cluster labels assigned to samples in a third subsample of the data. The first set of cluster labels is assigned by a model that was trained on the first subsample. The second set of cluster labels by a model trained on only the second subsample. For a total sample of n observations, both training samples contain m

observations. This leaves *n-2*m* observations to calculate the cluster stability. Finally, the number of clusters that maximises this stability is selected.

In Wang (2010) this cluster selection method is shown to perform well on simulated datasets. However, it is not applied on high-dimensional data such as text documents. In this paper, the cluster selection via crossvalidation is used to select the number of latent topics in a set of news articles. Similar to cluster analysis, topic modeling is a text-mining method for grouping documents based on hidden topics. Again, the selection of the number of topics is a common issue. To assess the use of crossvalidation cluster selection method for topic modeling, we raise the following research question:

How does crossvalidation with topic stability perform as a topic selection method?

To answer this question, two subquestions are posed: *How does topic selection via crossvalidation compare to other common topic selection methods?* and *Does topic selection via crossvalidation (with topic stability) lead to interpretable topics?*

The application of topic modeling to news articles can uncover interesting details about the coverage of news. For example, it was applied to analyse the framing of news articles on a specific government policy for assistance to artists and arts organisations (DiMaggio et al., 2013). Other interesting application of topic modeling include: recommending scientific articles (C. Wang & Blei, 2011), analyzing financial reports, twitter messages (Hong & Davison, 2010), customer service emails and more.

The remainder of this thesis is structured into four sections. In the next section, the methodology is discussed for topic modeling and selecting the number of topics. The collection and processing of the news articles is discussed in section 3. Topic selection results are presented in section 4. Finally, conclusions on the performance of crossvalidation as a topic selection method are drawn in section 5.

2 Methodology

In the next sections, clustering algorithms are denoted as $\Psi(\cdot; k)$ and trained clustering mappings are denoted by $\psi(x)$. The definitions below are taken from (J. Wang, 2010).

Definition 2.1. A clustering algorithm $\Psi(\cdot; k)$ with a given number of clusters $k \geq 2$ yields a clustering mapping $\psi(x)$ when applied to a sample z^n.

Definition 2.2. A clustering $\psi(x)$ is defined as a mapping $\psi : \mathbb{R}^p \to \{1,..., \mathrm{k}\}$

2.1 Topic Modeling

Latent Dirichlet allocation (LDA) is a generative probabilistic model for collections of discrete data (Blei et al., 2003). In this paper, LDA is applied to a collection of news articles. Considering the words used in the articles, each document is represented by topic probabilities. Thus, LDA can be seen as a soft clustering of the news articles.

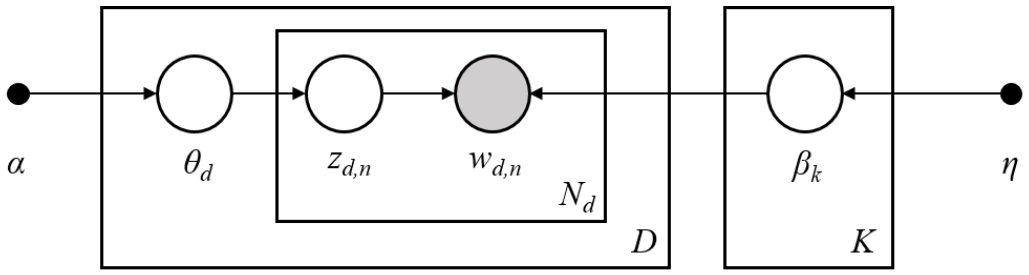

Figure 2: Graphical representation of the LDA model

In Figure 2, the collection of news articles is represented by the largest plate. In total there are D documents. Each document d contains a collection of words, represented by the N_d plate. The remaining plate represents the collection of K topics where K is selected via crossvalidation.

News articles cover a range of topics, such as 'sports' or 'economics'. Some words are more common in articles about sports than articles on economics. In LDA, every topic is represented by a distribution over the words. The word 'game' is for example more likely to appear in the 'sports' topic than the 'economics' topic.

To find latent topics in the text corpus, LDA assumes all documents were generated in three steps. First, the topic probabilities for the document are sampled from a Dirichlet distribution, $\theta_d \sim Dir(\alpha)$. For each word in the document, a topic is sampled from these probabilities through a single-trial Multinomial distribution $z_{d,n} \sim Multinomial(\theta_d)$. For example, the topic distribution for a document is 80% 'sports' and 20% 'economics'. This text is expected to have four times as many 'sports' words than 'economic' words.

The word-distributions of all K topics are modelled by a second Dirichlet distribution, Dir(η). Given a topic (k), a word is sampled from this topic's word distribution $w_{d,n}$ $\sim Multinomial(\beta_k)$. In the example above, this could result in the word 'football' for the 'sports' topic. This generative process is repeated for every word n in document d.

To conclude, the LDA model trains two Dirichlet distributions. Both a topic-distribution for each document and a word-distribution for each topic. The generation of documents follow three steps. First, on a document level, the topic-distribution is sampled. Then, on a word level, the topic of the specific word is sampled. Finally, the specific word is sampled from the topic's distribution of words.

The fitting of the distributions can be done via the Expectation-Maximisation algorithm. In the Expectation-step, words are assigned a topic and documents are assigned a topic as well. Then in the Maximisation-step, the Dirichlet distribution parameters are updated until convergence. More details on this are beyond the scope of this article but can be found in (Blei et al., 2003).

2.2 Topic Stability

The objective in topic modeling is to find latent topics in the text documents. When the complete sample of documents is permuted and split into three subsamples, the latent topics are expected to be the same in all three subsamples. If two topic models were trained on two subsamples of the same dataset, both models should learn the same patterns. In case both models are applied to previously unseen data, they should assign similar topics. This cluster stability is maximised over a range of values for the number of clusters (K).

Wang (2010) introduces a method to calculate the cluster stability via crossvalidation. First, the dataset is permuted and split into three mutually exclusive subsamples: two training sets of size m and a validation sample. Two cluster mappings are trained using the first and second subsample: $\psi_1^{*c} = \Psi(train1^{*c}; k)$, $\psi_2^{*c} = \Psi(train2^{*c}; k)$. Then, both these cluster mappings are applied to the validation subsample. This yields two sets of labels for the validation subsample: $\psi_1^{*c}(validation)$ and $\psi_2^{*c}(validation)$.

Given these labels, the cluster stability is calculated by counting the number of agreements between these two mappings. When one cluster mapping assigns observations a label '2' but another mapping assigns these observations label '3', they agree on the label of the observation. Simply comparing the labels however, would suggest a disagreement since '2' does not equal '3'. Therefore, cluster stability is counted as the number of times both models assign the same label to the pair of (validation) observations.

Figure 3: Splitting of permuted data

2.3 Topic Selection

After C permutations, the optimal number of clusters can either be determined by voting or by averaging. With voting, the optimal $\hat{k}^c$ is determined at each permutation by maximising the cluster stability in this permutation ($\hat{s}^{*c}(\Psi, k, m)$). After C permutations, the final number of clusters $\hat{k}$ is selected as the mode of $\{\hat{k}^1, ..., \hat{k}^C\}$. With averaging however, the performance measures are averaged over all permutations. Then, $\hat{k}$ is calculated by maximising the average cluster stability over all permutations (Algorithm 1).

Algorithm 1: Cluster selection via crossvalidation

Result: Optimal $\hat{k}^*$, number of clusters given clustering algorithm $\Psi(\cdot; k)$

for *c = 1, ..., C* **do**

Permute dataset X, and split X^{*c} into subsamples $train1^{*c}$, $train2^{*c}$, $valid^{*c}$

for *k = 1, ..., K* **do**

fit clustering algorithm $\Psi(\cdot; k)$ on $train1^{*c}$ and $train2^{*c}$ to get

$\psi_1^{*c} = \Psi(train1^{*c}; k)$, $\psi_2^{*c} = \Psi(train2^{*c}; k)$

estimate

$$\hat{s}^{*c}(\Psi, k, m) = \sum_{i=n-2*m}^{n-1} agreement(i)$$

where:

$$agreement(i) = \mathbb{1}[\mathbb{1}\{\psi_1^{*c}(x_i) = \psi_1^{*c}(x_{i+1})\} = \mathbb{1}\{\psi_2^{*c}(x_i) = \psi_2^{*c}(x_{i+1})\}]$$

end

$$\hat{k}^{*c} = argmax_{2 \leq k \leq K}\ \hat{s}^{*c}(\Psi, k, m)$$

end

$$\hat{k}_{voting} = mode\{\hat{k}^{*1}, ..., \hat{k}^{*C}\}$$

$$\hat{k}_{averaging} = argmax_{2 \leq k \leq K}\ \{C^{-1} \sum_{c=1}^{C} \hat{s}^{*c}(\Psi, k, m)\}$$

In the appendix, our reproduction results of the simulation study in (J. Wang, 2010) are presented. We find similar results and extend the study by varying the distance between clusters.

3 Data

In this paper, the cluster selection method is applied to a corpus of news articles. News articles can be grouped into categories such as sports, entertainment, politics etc. By applying the topic selection methods, the goal is to find the number of (latent) topics.

3.1 Data Collection

A sample of news articles is retrieved by using the newsAPI (https://newsapi.org/), this API includes news from several worldwide sources. However, there are certain restrictions on the number of requests and dates. For example, the developer (free) version of the API only allows for dates going back one month. We collected news articles from BBC news.

The newsAPI yields only a limited set of words from all articles. However, it does return the URL of every article. The Newspaper3k Python package https://newspaper.readthedocs.io/en/latest/ takes these URLs and downloads the full text. The final sample includes 318 articles written between May 10, 2020 and June 9, 2020. The dataset can be found here.

3.2 Data Preprocessing

The first step in topic modeling is to clean the raw text files. First, all words are transformed to lowercase. Then, all inflected words are transformed to their root form during lemmatisation. For example, 'making' is reduced to 'make'. After these two steps, stop words such as 'the', 'to' and 'and' (see Table 1) are removed since these words do not carry meaning outside of their context.

The spacy (https://spacy.io/) python package already includes a list of common English stop words. For this specific set of news articles, some additional words are added such as 'said' and 'people'. Besides these stop words, some other common words are presented in Table 2. Words like 'image', 'copyright', and 'caption' are removed as well since they are caused by the web scraping.

Table 1: Top 5 Most Frequent Words

Word	Count
the	9900
to	6295
and	4462
of	4399
in	3744

Table 2: Other Common Words

Word	Count
said	1100
people	781
image	527
copyright	482
caption	471

This is an example of the beginning of an unprocessed news article.

"We bring together three hairdressers from around the world to talk about how their lives have changed because of the pandemic. Marcel in Jerusalem and Marion in Berlin can cut their clients' hair again - but with restrictions. Tamsyn in Johannesburg has only been allowed to open her salon to sell hair products so far. So what is the future of cutting hair while the world is dealing with Covid-19?"

Compared to the processed text below, it can be seen that all words are now lowercase and words such as "her" and "of the" are removed. Also, stemming changed "products" to "product" and "dealing" to "deal".

'bring', 'hairdresser', 'world', 'talk', 'life', 'change', 'pandemic', 'marcel', 'jerusalem', 'marion', 'berlin', 'cut', 'client', 'hair', 'restriction', 'tamsyn', 'johannesburg', 'allow', 'open', 'salon', 'sell', 'hair', 'product', 'far', 'future', 'cut', 'hair', 'world', 'deal', 'covid-19'

3.3 Exploratory Data Analysis

After removing the stop words, the most frequent words are presented in Figure 4. Because of the time sensitivity of the news, these words depend heavily on the time range used in data collection. During the period of our data collection, the coronavirus had a large effect on the news coverage. This can be seen from the high frequency of words such as 'coronavirus', 'lockdown; and 'work'. Also, since these articles are collected from an English newspaper, it is logical that the words 'England', 'UK' and 'BBC' appear often.

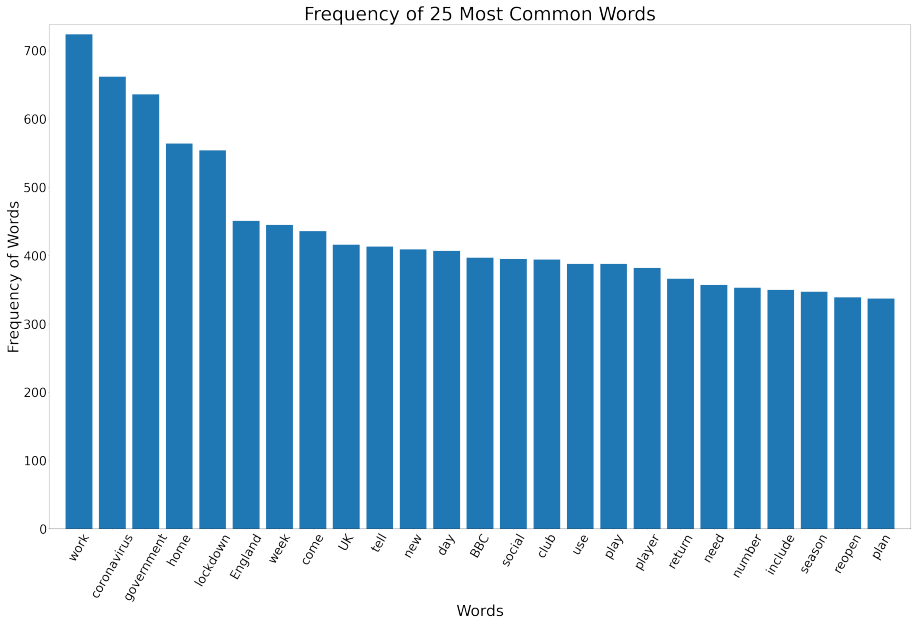

Figure 4: Frequency excl. stop words

Using the TextBlob Python package, the sentiment of texts can be analysed. The sentiment is predicted using a Natural Language Processing model for sentiment analysis. The values range from -1 (negative) to 1 (positive). The same package also predicts the subjectivity of a text which ranges from 0 (objective) to 1 (subjective).

The sentiment distribution of the news articles is shown in Figure 5. This set of news articles is on average rather objective with a subjectivity average of 0.40. Also, the maximum subjectivity score in this set of articles is 0.75 whereas its limit is 1 (Figure 6).

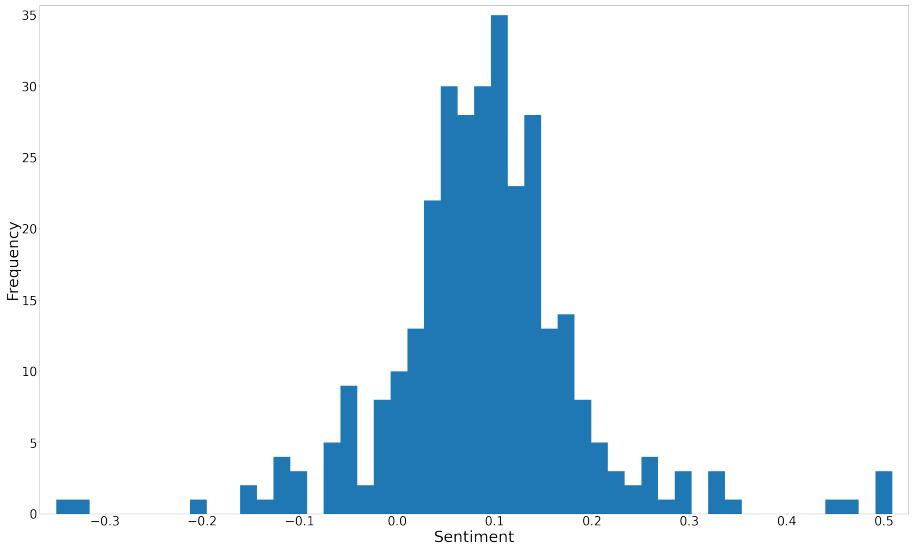

Figure 5: Sentiment of News Articles

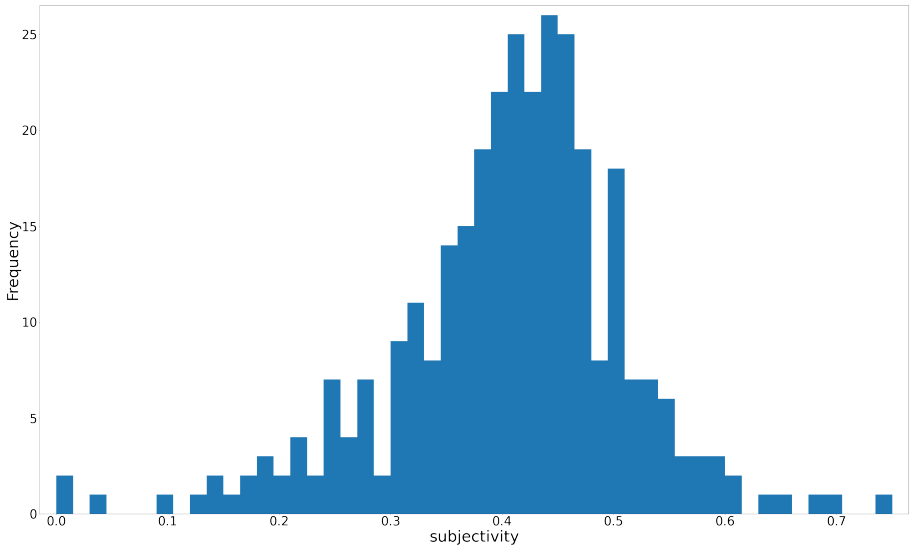

Figure 6: Subjectivity of News Articles

On average, a news article contains 518 words before removing stop words. The word count distribution is shown in figure 7. After removing stop words and other preprocessing steps, each document is transformed into a set of on average 248 tokens. These tokens are the lowercase, lemmatised and 'non-stop' word used to train the topic model. These steps reduce the token count per document to roughly half of its original word count.

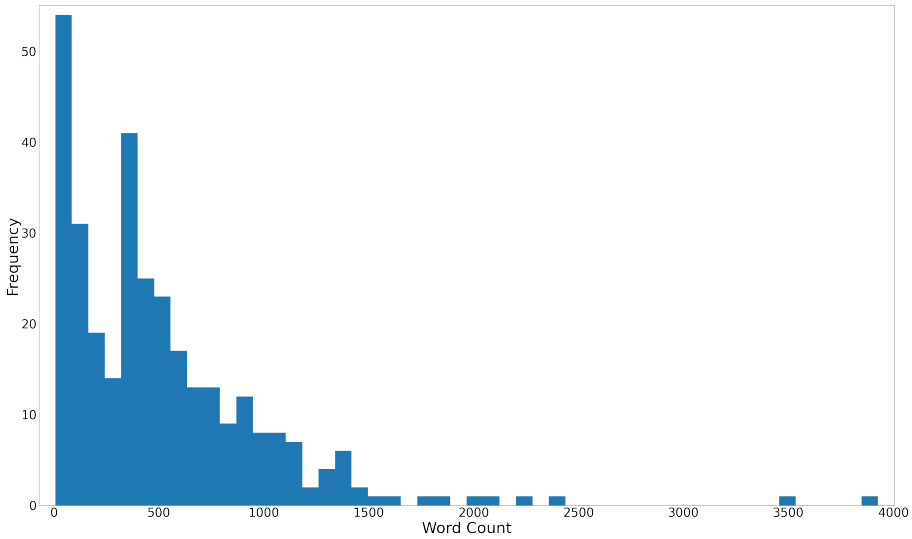

Figure 7: Word Count

The descriptive statistics are also summarised in table 3 below.

Table 3: Descriptive Statistics

	Mean	St. Dev.	Max.	Min.
Sentiment	0.090	0.103	0.508	-0.35
Subjectivity	0.406	0.105	0.75	0
Words per document	518	496	3927	7
Tokens per document	248	232	2113	4

4 Results

4.1 Topic Selection

To answer the research question (*'How does crossvalidation with topic stability perform as a topic selection method?'*), topic selection via crossvalidation is compared to topic selection via topic coherence. Topic coherence measures the semantic similarity between the most likely words per topic. Maximisation of coherence of topics is a common method for selecting the number of topics (Newman et al., 2010). Several measures have been developed (Slimani, 2013) to measure this semantic similarity. More details however, fall beyond the scope of this article.

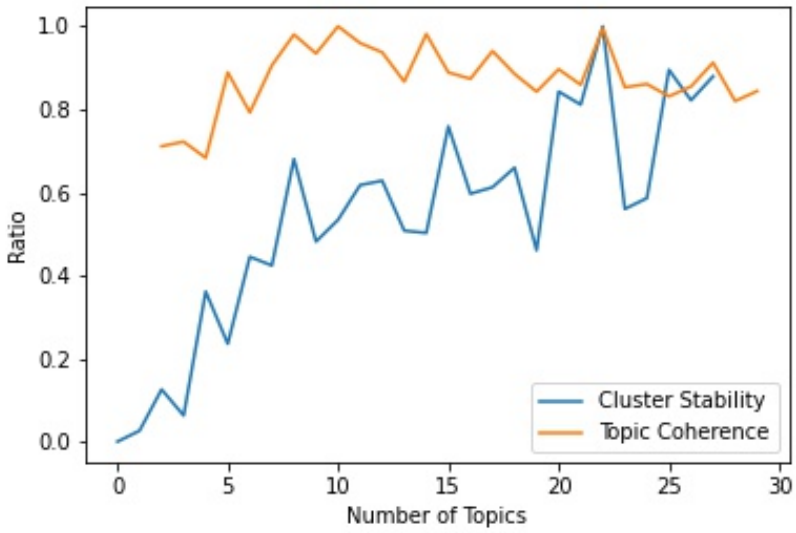

Figure 8: Topic Coherence and Topic Stability

To compare the cluster stability results with topic coherence, both are scaled to a [0, 1] range and plotted in Figure 8. The peak in coherence at K = 22 coincides with a peak in stability measured via crossvalidation with averaging. Considering the interpretability of the topic model, selecting 22 topics seems rather large. This larger number of topics could represent a set of subtopics or combined themes such as articles combining economics and politics.

4.2 Topic Interpretation

As explained before, latent Dirichlet allocation represents each topic by a distribution over the words. Each topic can then be interpreted by looking at the words it is most likely to generate. The most likely words per topic are given in the appendix. From these distributions, we could indeed infer four general themes for the 22 more detailed topics. The following tables show the grouping of the 22 topics based on the probability of certain words. The probability of the specific word w generated by the given topic k is given inside the brackets ($P(w \mid k)$).

Table 4: Top most likely words for sports topics

Topic	Word						
0	"player"	"season"	"club"	"league"			
	(0.016)	(0.007)	(0.007)	(0.005)			
11	"game"	"rugby"	"player"				
	(0.006)	(0.006)	(0.005)				
16	"ufc"	"ferguson"	"fight"	"mcgregor"	"title"	"gaethje"	
	(0.008)	(0.006)	(0.006)	(0.006)	(0.006)	(0.006)	
17	"vettel"	"team"	"ferrari"	"tennis"			
	(0.010)	(0.010)	(0.006)	(0.005)			
21	"league"	"club"	"season"	"play"	"player"	"game"	"football"
	(0.019)	(0.018)	(0.011)	(0.010)	(0.009)	(0.008)	(0.006)

From table 7 it becomes clear that topics 0, 11, 16, 17 and 21 are in general 'sports' topics. For example, topic 16 is about fighting sports. The UFC is short for 'Ultimate Fighting Championship' and Ferguson, McGregor and Gaethje are Mixed Martial Arts fighters. Topic 11 is specifically about rugby sport, topic 17 on racing and topic 21 probably on football. To conclude, topic 0 is likely to cover more general sports.

Table 5: Top most likely words for social topics

Topic	Word				
6	"ireland"	"death"	"coronavirus"	"hospital"	
	(0.010)	(0.008)	(0.008)	(0.007)	
9	"child"	"scrabble"	"game"	"school"	
	(0.007)	(0.006)	(0.006)	(0.006)	
12	"content"	"moderator"	"facebook"	"harmful"	"coronavirus"
	(0.016)	(0.015)	(0.014)	(0.010)	(0.007)
14	"case"	"virus"	"lockdown"	"coronavirus"	
	(0.008)	(0.007)	(0.005)	(0.005)	
18	"reopen"	"school"	"social"	"distancing"	"lockdown"
	(0.025)	(0.014)	(0.011)	(0.010)	(0.010)

Common words in these topics are "coronavirus", "lockdown" and "school". These topics cover the impact of the epidemic on society. For example, topic 9 and 18 are about schools. Topic 12 is about the impact of content spread through Facebook and potential harm. Finally, topic 6 and 14 are probably on the number of cases and its impact on hospitals and lockdown.

Table 6: Top most likely words for political topics

Topic	Word				
5	"china"	"mask"	"face"	"coronavirus"	"government"
	(0.017)	(0.014)	(0.012)	(0.005)	(0.005)
7	"argentina"	"government"	"debt"	"standard"	"food"
	(0.008)	(0.007)	(0.005)	(0.005)	(0.005)
8	"school"	"health"	"government"		
	(0.005)	(0.005)	(0.004)		
10	"work"	"government"	"lockdown"	"coronavirus"	
	(0.008)	(0.008)	(0.007)	(0.007)	
13	"coronavirus"	"president"	"brazil"	"russia"	"putin"
	(0.010)	(0.008)	(0.007)	(0.007)	(0.007)
15	"government"	"work"	"england"	"test"	"johnson"
	(0.013)	(0.009)	(0.008)	(0.006)	(0.006)
19	"work"	"lockdown"	"transport"	"government"	"police"
	(0.011)	(0.007)	(0.007)	(0.006)	(0.006)
20	"scotland"	"minister"	"government"	"sturgeon"	"england"
	(0.014)	(0.013)	(0.010)	(0.009)	(0.007)

The main difference between the previous group of social topics and these political topics is that the political topics are more likely to contain words like "government". Also, several countries such as China (5), Russia (13), Argentina (7) and Brazil (13) are included. These topics are more likely to cover international political relations or the political situation in countries. Topic 7 for example, is likely to cover the political situation in Argentina. Topic 8, 10, 15, 19 and 20 seem to cover politics in the United Kingdom. Boris Johnson (15) is the Prime Minister of the United Kingdom and Nicola Sturgeon (20) is the First Minister of Scotland.

Table 7: Top most likely words for economic topics

Topic	Word					
1	"old"	"coronavirus"	"run"	"world"	"money"	"recession"
	(0.006)	(0.006)	(0.005)	(0.005)	(0.005)	(0.004)
2	"scheme"	"furlough"	"business"	"job"	"employer"	
	(0.020)	(0.012)	(0.009)	(0.009)	(0.009)	
3	"fight"	"england"	"world"	"garden"	"work"	
	(0.008)	(0.007)	(0.006)	(0.005)	(0.004)	
4	"work"	"pay"	"woman"	"employer"	"hospital"	"pregnant"
	(0.010)	(0.008)	(0.007)	(0.006)	(0.005)	(0.004)

Four topics remain to be interpreted. Considering the word distributions, these topics tend to cover more economic concepts such as "business", "employer" and "money". Topic 2 and 4 are more likely to mention "work" and cover news related to furlough (leave of absence) or woman in the workforce. Topic 1 is more likely to cover the economic impact of the coronavirus on the world. Topic 3 however seems to be ambiguous.

By regrouping the 22 selected topics into these four groups, it becomes clear that the fitted topic model indeed returns a larger set of subtopics. However, looking back at figure 8, both topic stability and coherence actually show some of the worst values for 3-5 topics. Topic selection via any of these methods would not select four topics. Selecting a small number of topics (3-5) results in either very unstable topics or non-coherent topics. Instead topic selection results in a much more detailed topic analysis.

5 Conclusion

In this paper, we analysed the following research question: *How does crossvalidation with topic stability perform as a topic selection method?*. We use crossvalidation to select the number of latent topics in a set of news articles. Given this number, the latent Dirichlet allocation model clusters documents based on these latent topics. This model assumes text documents are generated by sampling from two Dirichlet distributions. Each document is represented by a topic-distribution and each topic by a distribution over the words.

Topic selection via crossvalidation selects the number of topics such that the stability of the topics is maximised. Stable topics are reproducible when the topic model is trained on different articles drawn from the same source. For our set of news articles, cluster stability was maximised for 22 topics.

We compared this result to topic selection via topic coherence. Topic coherence is a common method to select the number of topics. It selects the number of topics based on the semantic similarity of the words with the highest probability per topic. We find that topic coherence is maximised for 22 topics, this shows that topic selection via topic coherence and stability lead to the same result. Therefore, crossvalidation with topic stability as a topic selection method performs equal to the common topic selection method.

To answer our research question, we also assess the interpretability of the selected latent topics. Each topic can be represented by a set of words ordered by the probability to be generated from the given topic. This representation shows the most likely words per topic, which are semantically similar and allow us to interpret what this topic is about. We regrouped the 22 topics and found that most were more detailed subtopics of more broad topics such as 'sports', 'politics', 'economics' and 'social'. For example, we found five topics on sports where some only covered a single sport such as rugby or fighting.

To conclude, we find that crossvalidation with cluster stability can also be applied to select the number of latent topics in a set of news articles. It results in coherent topics that can be interpreted as detailed specialisations of more general news categories.

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

DiMaggio, P., Nag, M., & Blei, D. (2013). Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of us government arts funding. *Poetics*, *41*(6), 570–606.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York, NY, USA, Springer New York Inc.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, *44*(1), 1–12.

Hong, L., & Davison, B. D. (2010). Empirical study of topic modeling in twitter, In *Proceedings of the first workshop on social media analytics*.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, *24*(7), 881–892.

Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010). Automatic evaluation of topic coherence, In *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*.

Slimani, T. (2013). Description and evaluation of semantic similarity measures approaches. *arXiv preprint arXiv:1310.8059*.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411–423.

Wang, C., & Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles, In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining*.

Wang, J. (2010). Consistent selection of the number of clusters via crossvalidation. *Biometrika*, *97*(4), 893–904. http://www.jstor.org/stable/29777144

A Appendix

A.1 Additional Results

Word-distributions for all 22 topics

Sports (0, 11, 16, 17, 21)

(0, '0.016*"player" + 0.014*"day" + 0.011*"mother" + 0.009*"keegan" + ' '0.008*"newcastle" + 0.007*"season" + 0.007*"club" + 0.006*"anna" + ' '0.005*"league" + 0.005*"like"'),

(11, '0.007*"world" + 0.006*"game" + 0.006*"rugby" + 0.005*"player" + ' '0.004*"post" + 0.004*"loan" + 0.004*"medium" + 0.004*"social" + 0.004*"tom" ' '+ 0.004*"stiller"'),

(16, '0.008*"ufc" + 0.006*"ferguson" + 0.006*"fight" + 0.006*"mcgregor" + ' '0.006*"come" + 0.006*"title" + 0.006*"gaethje" + 0.005*"group" + ' '0.005*"party" + 0.004*"event"'),

(17, '0.011*"test" + 0.010*"vettel" + 0.010*"team" + 0.006*"end" + ' '0.006*"ferrari" + 0.006*"england" + 0.005*"home" + 0.005*"tennis" + ' '0.005*"aqe" + 0.005*"school"'),

(21, '0.019*"league" + 0.018*"club" + 0.011*"season" + 0.010*"play" + ' '0.009*"player" + 0.008*"game" + 0.007*"final" + 0.006*"premier" + ' '0.006*"football" + 0.006*"city"')

Economics (1, 2, 3, 4)

(1, '0.006*"old" + 0.006*"coronavirus" + 0.005*"run" + 0.005*"neanderthal" + ' '0.005*"world" + 0.005*"find" + 0.005*"money" + 0.004*"klein" + 0.004*"try" ' '+ 0.004*"recession"'),

(2, '0.020*"scheme" + 0.012*"furlough" + 0.009*"business" + 0.009*"job" + ' '0.009*"employer" + 0.008*"sunak" + 0.008*"cost" + 0.007*"chancellor" + ' '0.007*"scottish" + 0.007*"£"'),

(3, '0.008*"fight" + 0.007*"england" + 0.006*"world" + 0.005*"garden" + ' '0.005*"week" + 0.005*"strip" + 0.004*"look" + 0.004*"school" + 0.004*"team" ' '+ 0.004*"work"'),

(4, '0.010*"work" + 0.008*"pay" + 0.007*"woman" + 0.006*"employer" + ' '0.006*"robot" + 0.006*"government" + 0.005*"hospital" + 0.005*"attack" + ' '0.004*"pregnant" +

0.004*"tell"'),

Politics (5, 7, 8, 10, 13, 15, 19, 20)

(5, '0.017*"china" + 0.014*"mask" + 0.012*"face" + 0.010*"care" + 0.008*"use" + '
'0.007*"intensive" + 0.006*"wear" + 0.005*"need" + 0.005*"coronavirus" + ' '0.005*"government"'),

(7, '0.009*"moth" + 0.008*"argentina" + 0.007*"government" + 0.005*"debt" + '
'0.005*"payment" + 0.005*"standard" + 0.005*"food" + 0.005*"interest" + ' '0.004*"santos" + 0.004*"dos"'),

(8, '0.007*"child" + 0.006*"company" + 0.005*"po" + 0.005*"school" + ' '0.005*"health"
+ 0.005*"parent" + 0.005*"choir" + 0.005*"covid-19" + ' '0.005*"new" + 0.004*"government"'),

(10, '0.008*"work" + 0.008*"government" + 0.007*"lockdown" + 0.007*"coronavirus"
' '+ 0.006*"home" + 0.006*"england" + 0.006*"new" + 0.005*"guidance" + ' '0.004*"use"
+ 0.004*"need"'),

(13, '0.011*"country" + 0.010*"coronavirus" + 0.010*"virus" + 0.010*"case" + '
'0.008*"president" + 0.008*"price" + 0.008*"mr" + 0.007*"brazil" + ' '0.007*"russia"
+ 0.007*"putin"'),

(15, '0.013*"government" + 0.009*"work" + 0.009*"mr" + 0.008*"england" + ' '0.007*"uk"
+ 0.007*"home" + 0.006*"test" + 0.006*"coronavirus" + ' '0.006*"johnson" + 0.006*"wales"'),

(19, '0.011*"work" + 0.007*"lockdown" + 0.007*"transport" + 0.006*"government"
+ ' '0.006*"home" + 0.006*"police" + 0.006*"london" + 0.006*"travel" + ' '0.006*"social" + 0.005*"distancing"'),

(20, '0.016*"stay" + 0.014*"scotland" + 0.013*"minister" + 0.010*"government" + '
'0.010*"home" + 0.009*"sturgeon" + 0.008*"message" + 0.008*"different" + ' '0.008*"uk"
+ 0.007*"england"'),

Social (6, 9, 12, 14, 18)

(6, '0.010*"ireland" + 0.008*"death" + 0.008*"coronavirus" + 0.007*"hospital" + '
'0.007*"northern" + 0.007*"message" + 0.007*"covid-19" + 0.006*"patient" + ' '0.005*"report" + 0.005*"executive"'),

(9, '0.009*"dalglish" + 0.007*"child" + 0.006*"scrabble" + 0.006*"game" + ' '0.006*"school" + 0.005*"play" + 0.005*"police" + 0.005*"hamilton" + ' '0.005*"cup" + 0.005*"parent"'),

(12, '0.016*"content" + 0.015*"moderator" + 0.014*"facebook" + 0.010*"harmful" + ' '0.007*"coronavirus" + 0.007*"human" + 0.007*"detect" + 0.007*"ai" + ' '0.006*"social" + 0.006*"information"'),

(14, '0.008*"case" + 0.007*"virus" + 0.006*"bbc" + 0.006*"week" + ' '0.005*"lockdown" + 0.005*"scotland" + 0.005*"coronavirus" + 0.004*"state" + ' '0.004*"new" + 0.004*"report"'),

(18, '0.025*"reopen" + 0.014*"school" + 0.013*"allow" + 0.011*"june" + ' '0.011*"social" + 0.010*"distancing" + 0.010*"lockdown" + 0.008*"open" + ' '0.007*"shop" + 0.006*"child"'),

A.2 Simulation Study

To find comparable results to (J. Wang, 2010), the following dataset with two clusters and 200 observations is simulated.

Algorithm 2: Simulation Data following (Tibshirani et al., 2001)

set random seed = 0 for reproducibility

set t equal to a series of 100 equally spaced values between [-0.5, 0.5]

for *i = 0, ..., 100 (cluster 0)* **do**

$y_i = 0$

$x_{1,i} = t_i + \mathcal{N}(0, 0.1)$

$x_{2,i} = t_i + \mathcal{N}(0, 0.1)$

$x_{3,i} = t_i + \mathcal{N}(0, 0.1)$

end

for *i = 101, ..., 200 (cluster 1)* **do**

$y_i = 1$

$x_{1,i} = t_i + \mathcal{N}(0, 0.1) + 10$

$x_{2,i} = t_i + \mathcal{N}(0, 0.1) + 10$

$x_{3,i} = t_i + \mathcal{N}(0, 0.1) + 10$

end

This dataset shows two elongated clusters in three dimensions. In the figure below, it can be seen that this example does not contain any difficulties such as overlapping clusters. Also, the distance between the clusters is quite large. In this simulation study, this distance is increased to extend the simulation experiments for less well separated clusters.

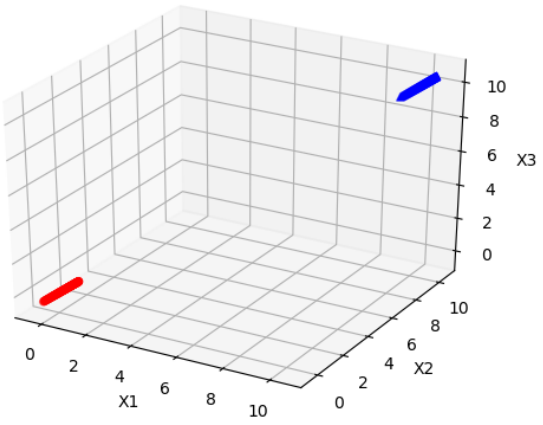

Figure 9: Simulated data for example (e) in Tibshirani et al. (2001b)

These observations are clustered by applying the k-means model (Hastie et al., 2001).

With the same specifications, we find the same results as (J. Wang, 2010) both with voting and averaging. The specifications of the simulation are set as follows. The number of splittings (permutations) is set to be 100. The total sample size (n) is equal to 200, 100 observations for both labels 0 and 1. The sample is split into two training samples of size 67 (m) and a validation sample of size 66. Finally, the simulation is run 50 times.

Table 8: cluster selection results for 50 simulation runs

	Selected K								
Method	2	3	4	5	6	7	8	9	10
CV_v	50	0	0	0	0	0	0	0	0
CV_a	50	0	0	0	0	0	0	0	0

Table 8 shows how both crossvalidation with averaging and voting select the correct number of clusters (2) 50 out of 50 times. These results are a replication of the results in (J. Wang, 2010). However, the clustering of this simulated data is quite an easy case. As can be seen in 9, the simulated clusters are very well-separated.

To test this crossvalidation cluster selection method for clusters less separated, the distance between both simulated clusters is varied between 0 and 10. Previously, the distance was set at 10. Table 9 shows that for all distances except a distance of 1 the selected number of clusters remains 2. However, when the distance is set to 1, the crossvalidation selection method selects 4 clusters instead of 2.

Table 9: cluster selection results for varying distances

	Distance										
Method	0	1	2	3	4	5	6	7	8	9	10
CV_v	2	4	2	2	2	2	2	2	2	2	2
CV_a	2	4	2	2	2	2	2	2	2	2	2

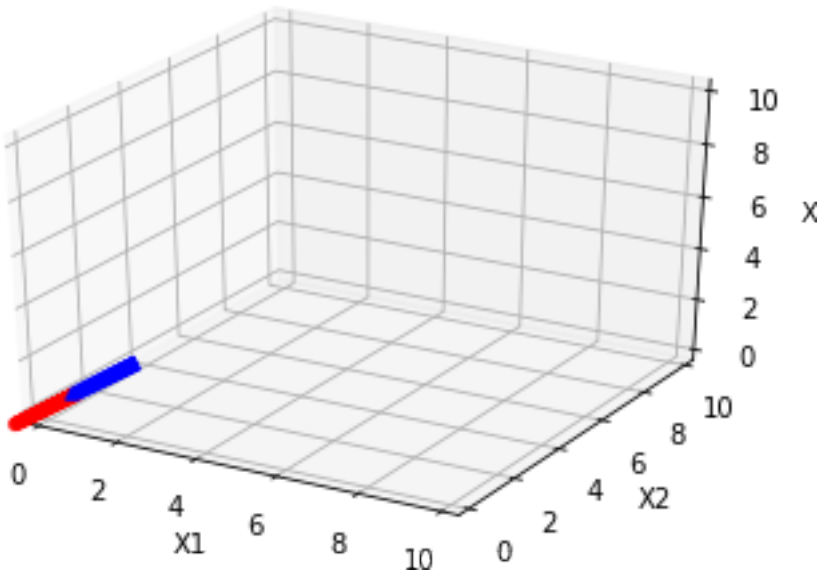

Figure 10: Simulated data where distance is set to 1

A.3 Python Code

In this section, we present all the code required to run the analyses in the paper.

Topic Modeling: Data Collection

This script scrapes news articles from websites. The URLs are retrieved via the newsAPI and requires a personal token (newsapi.org).

```
begin = datetime.datetime(2020, 5, 10)
dataset_big_bbc = []
days = 30
units = days*24
articles_per_hour = []

for days in range(units):
  n = 0
  until = begin+datetime.timedelta(hours=3) # take care of too many requests
  untilStr = until.strftime("%Y-%m-%dT%H:%M:%S")
  beginStr = begin.strftime("%Y-%m-%dT%H:%M:%S")
  try:
    url = ('https://newsapi.org/v2/everything?'
           'sources=bbc-news&'
           'pageSize=100&'+
           'from='+beginStr+'&to='+untilStr+'&'+
           'apiKey=b3847c488b8e4f8cbac4395519965200')
    response = requests.get(url)
    articles_per_hour.append(response.json().get('totalResults'))
    print (response.json())
    for artic in response.json().get('articles'):
      n += 1
      article = Article(artic.get('url'))
```

```
        article.download()
        article.parse()
        dataset_big_bbc.append(article.text)
    except:
        print('ERROR')

    begin=until
```

Topic Modeling: Data Preprocessing

This script contains all necessary functions to transform the collection of raw articles: removing stopwords, lemmatization, lowercase, tokenization.

```
with open('/content/articles_bbc.txt', 'r') as filehandle:
    articles_bbc = json.load(filehandle)

nlp= spacy.load("en")

# extra words to remove which are not informative such as copyright, Getty I
stop_list = ["Mrs.","Ms.","say","'s","Mr.","Mr", "image", "copyright", "capt

nlp.Defaults.stop_words.update(stop_list)

# Iterates over the words in the stop words list and resets the "is_stop" fl
for word in STOP_WORDS:
    lexeme = nlp.vocab[word]
    lexeme.is_stop = True

def lemmatizer(doc):
    # This takes in a doc of tokens from the NER and lemmatizes them.
    # Pronouns (like "I" and "you" get lemmatized to '-PRON-', so I'm removi
```

```
    doc = [token.lemma_ for token in doc if token.lemma_ != '-PRON-']
    doc = u' '.join(doc)
    return nlp.make_doc(doc)

def remove_stopwords(doc):
    # This will remove stopwords and punctuation.
    # Use token.text to return strings, which we'll need for Gensim.
    doc = [token.text for token in doc if token.is_stop != True and token.is
    return doc

# The add_pipe function appends our functions to the default pipeline.
nlp.add_pipe(lemmatizer,name='lemmatizer',after='ner')
nlp.add_pipe(remove_stopwords, name="stopwords", last=True)

doc_list = []
for doc in tqdm(np.unique(articles_bbc)):
  # Passes that article through the pipeline and adds to a new list.
  doc = doc.lower()
  pr = nlp(doc)
  doc_list.append(pr)

words = doc_list
allwords = []
for wordlist in words:
    allwords += wordlist

sentiment = [TextBlob(x).sentiment.polarity for x in articles_bbc_unique]
print('Average sentiment of {} ({})'.format(np.mean(sentiment), np.std(senti
```

```
plt.figure(figsize=(50,30))
plt.margins(0.02)
plt.xlabel('Sentiment', fontsize=50)
plt.xticks(fontsize=40)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.hist(sentiment, bins=50)
#plt.title('Sentiment Distribution', fontsize=60)
plt.show()

articles_bbc_unique[np.argmax(sentiment)]

articles_bbc_unique[np.argmin(sentiment)]

token_count = [len(str(x).split()) for x in doc_list] #.apply(lambda x: len(
#review_len = data_today.astype(str).apply(len)

word_count = [len(str(x).split()) for x in articles_bbc_unique] #.apply(lamb
#review_len = data_today.astype(str).apply(len)

sentiment_unique = [TextBlob(x).sentiment.polarity for x in articles_bbc_uni
print('Average sentiment of {} ({})'.format(np.mean(sentiment_unique), np.st
print('Average word count of {} ({})'.format(np.mean(word_count), np.std(wor
print('Average token count of {} ({})'.format(np.mean(token_count), np.std(t

k2, p = stats.normaltest(sentiment_unique)
if p < 0.05:
  print('reject, no normal')
```

```
plt.figure(figsize=(50,30))
plt.margins(0.02)
plt.xlabel('Word Count', fontsize=50)
plt.xticks(fontsize=40)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.hist(word_count, bins=50)
#plt.title('Count Distribution', fontsize=60)
plt.show()

plt.figure(figsize=(50,30))
plt.margins(0.02)
plt.xlabel('Sentiment', fontsize=50)
plt.xticks(fontsize=40)
plt.ylabel('Frequency', fontsize=50)
plt.yticks(fontsize=40)
plt.hist(sentiment_unique, bins=50)
#plt.title('Sentiment Distribution', fontsize=60)
plt.show()

mostcommon_small = FreqDist(allwords).most_common(25)
x, y = zip(*mostcommon_small)
plt.figure(figsize=(50,30))
plt.margins(0.02)
plt.bar(x, y)
plt.xlabel('Words', fontsize=50)
plt.ylabel('Frequency of Words', fontsize=50)
plt.yticks(fontsize=40)
plt.xticks(rotation=60, fontsize=40)
```

```
plt.show()
```

```
f = open('/content/articles_bbc.txt', 'r')
words = nltk.word_tokenize(f.read())

# Remove single-character tokens (mostly punctuation)
words = [word for word in words if len(word) > 1]

# Remove numbers
words = [word for word in words if not word.isnumeric()]

# Lowercase all words (default_stopwords are lowercase too)
words = [word.lower() for word in words]

# Calculate frequency distribution
fdist = nltk.FreqDist(words)

for word, frequency in fdist.most_common(100):
    print(u'{};{}'.format(word, frequency))
```

Topic Modeling: latent Dirichlet allocation

The latent Dirichlet allocation model is trained via the https://radimrehurek.com/gensim/models/ldamod

```
def split_texts(X, m, n, perm = True):
    ''' Cross Validation: Split Data '''
    if perm:
        X = np.random.permutation(X)
    z1 = X[:m]
```

```
    z2 = X[m:2*m]
    z3 = X[2*m:]
    return z1, z2, z3

def build_model(doc_list, k):
    '''doc_list is z1 or z2'''
    # Creates, which is a mapping of word IDs to words.
    words = corpora.Dictionary(doc_list)

    # Turns each document into a bag of words.
    corpus = [words.doc2bow(doc) for doc in doc_list]

    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                id2word=words,
                                                num_topics=k,
                                                random_state=0,
                                                update_every=1,
                                                passes=10,
                                                alpha='auto',
                                                per_word_topics=True)
    return words, corpus, lda_model

def document_labeler(proc_text, lda_model, words):
    '''input proc_text is z3'''
    corpus_valid = [words.doc2bow(doc) for doc in proc_text]
    dicts = dict(lda_model.get_document_topics(corpus_valid)[0])
    label = max(dicts, key=dicts.get)
    return label
```

```
def lda_predict(texts, lda_model, words):
    '''input proc_text is z3'''
    labels = []
    for doc in texts:
        corpus_valid = [words.doc2bow(doc)]
        dicts = dict(lda_model.get_document_topics(corpus_valid)[0])
        label = max(dicts, key=dicts.get)
        labels.append(label)
    return labels
```

To select the topics, the following script measures cluster stability and coherence for a range of number of clusters.

```
def check_stability(doc_list, n=len(doc_list), m=int(len(doc_list)/3), K=30)
    z1, z2, z3 = split_texts(doc_list, m, n)
    stabilities = []

    for k in range(2, K):
        words1, corpus1, lda_model1 = build_model(z1, k)
        words2, corpus2, lda_model2 = build_model(z2, k)

        labels1 = lda_predict(z3, lda_model1, words1)
        labels2 = lda_predict(z3, lda_model2, words2)

        stability = 0
        for i in range(n-2*m -1):
            j = i+1
            agree1 = labels1[i] == labels1[j]
            agree2 = labels2[i] == labels2[j]
            if agree1 == agree2:
```

```
            stability += 1

        stabilities.append(stability)

    return stabilities

def check_coherence(doc_list, n=len(doc_list), m=int(len(doc_list)/3), K=30,
    coherences = []
    for k in range(2,K):
        words, corpus, lda_model = build_model(doc_list, k)
        coherence_model_lda = CoherenceModel(model=lda_model, texts=doc_list, co
        coherence = coherence_model_lda.get_coherence()
        if print_value:
            print('\nCoherence Score for k = {}: {} '.format(k, coherence))
        coherences.append(coherence)
    return coherences

K = 30
stabilities_C = np.empty((0, K-2))
voting_k = []
for c in range(10):
    print(c)
    stabilities = check_stability(doc_list, K=K)
    stabilities_C = np.vstack([stabilities_C, stabilities])
    voting_k.append(np.argmax(stabilities))

print(np.argmax(np.mean(stabilities_C, axis=0))+2)

coherences = check_coherence(doc_list, K=K)
```

Simulation Study: Data Generation

To reproduce the simulation in (Tibshirani et al., 2001), the following function generates the data for a given distance.

```
# code for simulating example (e) in Tibshirani et al. (2001b)
import numpy as np

def simulate_tibshirani(distance=10):
    np.random.seed(0)
    t = np.linspace(-0.5,0.5,100)
    X = None
    y = None

    x1_0 = t + np.random.normal(0, 0.1)
    x2_0 = t + np.random.normal(0, 0.1)
    x3_0 = t + np.random.normal(0, 0.1)

    X = np.transpose(np.array([x1_0, x2_0, x3_0]))

    x1_1 = t + np.random.normal(0, 0.1) + distance
    x2_1 = t + np.random.normal(0, 0.1) + distance
    x3_1 = t + np.random.normal(0, 0.1) + distance

    X = np.vstack([X,np.transpose(np.array([x1_1, x2_1, x3_1]))])
    y = np.concatenate([np.zeros(100), np.ones(100)])

    assert X.shape == (200, 3)
    assert y.shape == (200, )
    return X, y
```

Simulation Study: Clustering Method

In this reproduction study, the simulated data points are clustered via the popular k-means model (Kanungo et al., 2002).

```
# code for measuring cluster stability with kmeans
from sklearn.cluster import KMeans
import numpy as np

def split(X, m, n, perm=True):
    np.random.seed(0)
    ''' Cross Validation: Split Data '''
    if perm:
        X = np.random.permutation(X)
    z1 = X[:m]
    z2 = X[m:2*m]
    z3 = X[2*m:]
    return z1, z2, z3

def instabilities_kmeans(X, K=10, m=67, n=200, inits=20):
    instabilities = []
    inertias = []
    z1, z2, z3 = split(X, m, n)

    for k in range(2, K):
        inertias.append(0)
        model = KMeans(n_clusters=k, n_init=inits, random_state=0)
        model.fit(X)
        inertias.append(model.inertia_)

        model1 = KMeans(n_clusters=k, n_init=inits, random_state=0)
```

```
        model2 = KMeans(n_clusters=k, n_init=inits, random_state=0)
        model1.fit(z1)
        model2.fit(z2)

        labels1 = model1.predict(z3)
        labels2 = model2.predict(z3)
        instability = 0
        for i in range(len(z3)-1):
            j = i+1
            agree1 = labels1[i] == labels1[j]
            agree2 = labels2[i] == labels2[j]
            if (agree1 != agree2):
                instability += 1
        instabilities.append(instability)

    assert len(instabilities) == K-2
    return instabilities, inertias
```

Simulation Study: Varying Distance

In the original example the clusters are highly separated. This simulation is extended by varying the distance between the two simulated clusters between 0 and 10.

```
# select number of topics for varying distances
from measures import split, instabilities_kmeans, inertias_kmeans
from tibshirani import simulate_tibshirani
import numpy as np
import pandas as pd
import stats

CVa = []
```

```
CVv = []

for d in range(10):
  X, y = simulate_tibshirani(d)
  instabilities_voting, instabilities_averaging = instabilities_kmeans(X, K=

  CVa.append(np.argmin(instabilities_averaging)+2)
  CVv.append(stats.mode(instabilities_voting)+2)

df = pd.DataFrame(list(zip(CVa, CVv)))
df.T.to_csv('results-simulation-varying.csv')
```