**Topic Modeling on a million NEWS Headlines**

**Bhavish Kumar, Prankur Garg, Tejas Patil**

School of Information Studies, Syracuse University

**Dr. Bei Yu**

Syracuse University

**Abstract**

This project is based on a large corpus of NEWS headlines published by “Australian Broadcasting Corporation” between the years 2003 to 2019. The goal of the project is to use this corpus of news headlines to identify all the topics of discussion in the news produced by Australian Broadcasting Corp. (ABC). Moreover, to maintain a well-structured and organized corpus of News, it is essential to categorize every news headline under a set of topics to provide easy and quick access of news content for their viewers.

**1. Introduction**

Australian Broadcasting Corporation is a big & reputed news channel capable of producing a large number (millions) of news articles every month. Because of such large scale production, the news articles/headlines are quite often un labelled, making it unfit for storing them in an organized manner. For ABC to maintain an organized storage, all the news articles of the whole corpus must be labelled into a specific set of topics and grouped.

Short Texts such as the News Headlines are also an important source of information which can be extracted from efficient & effective topic models capable of discovering topics from a collection of documents. Topic Modeling algorithms such as Latent Dirichlet Allocation (LDA) [1] and Non-Negative Matrix Factorization (NMF) [2] are widely adopted for discovering topics from a text corpus, without requiring any prior annotations or labeling of the documents. In this paper we examine both the LDA & NMF algorithms and compare the performance of the topic models produced by both. The model performance is going to be examined by assessing the coherence of topics based on human judgement by observing the word intrusion for a topic (Ideally, the top 20 words of a given topic should only contain words from a single domain and should not have an intruding word from another domain). The LDA model performance can also be quantified using “Perplexity”[[1]](#footnote-1)[2] & “Log Likelihood”[[2]](#footnote-2)[3] scores.

**2. Data Description**

The data set used for this project is obtained from the Kaggle website. The link to the data set can be found below this paragraph. This data set is composed of two variables namely, ‘publish\_date’ and ‘headline\_text’. The first variable contains the information about the date on which the article was published, and the second variable contains the headline of the corresponding news articles. The given data set has a total of 1155838 observations. It is a collection of the headlines of news articles published over the period of 17 years (from the date 02/19/2003 to 12/31/2019). Most of the news articles have a focus on the local news about Australia but a significant amount of documents also represent the historical records of noteworthy events around the planet.

**3. Method**

We shall discuss the 3 unsupervised exploratory text mining approaches that have been used for the project. The approach involves data preprocessing through tokenization (what to count) and vectorization (how to count) to convert the unstructured text data to structured data with a vector of numbers representing each text document. Once the text data has been vectorized and then transformed into a Term-Document Matrix, the required machine learning algorithm can be applied on the matrix to build models. The sklearn library has the required Countvectorizer [4] and tfidf[[3]](#footnote-3) vectorizer [5] functions with all the required parameters suitable for converting a text corpus into a term document matrix. We shall discuss all the parameters and the vectorization techniques used for each of the 3 algorithms in the below sections. We will also compare and discuss the performance of the models. [6] [7]

**3.1 LDA Algorithm**

LDA is a probabilistic model that tries to estimate the probability distributions for topics in documents and words in topics. The LDA output provides us with a Topic level probability distribution of words which helps us to label the topics. The LDA output also consists of document level probability distribution of topics which helps us to assign a topic for each document. The number of topics can be tuned until we obtain coherent topics with minimum word intrusion [7]. Intruders are words which are out of place and don’t belong together with the other words.

**3.1.1 Data Preprocessing**

Started off with Tokenization to create a vocabulary (set of features) from the corpus. To reduce the size of the vocabulary the following filtering techniques were used:

i. *Removing Stop words*: Stopwords are unnecessary for a document, as they have no role in explaining the category of a document and are not at all helpful in text classification. Hence, they can be removed.

ii. *Lowercase Merging*: Since most of the uppercase and lowercase words have the same meaning, they can be treated as the same. Hence, we applied Lowercase Merging also, along with stopwords removal.

iii. *Minimum Document Frequency Filter:* Weremoved words that occur in very few documents by trying the min\_df = 2 & min\_df = 5 filter. This is done to improve the generalization performance and reduce overfitting.

iv. *Maximum Document Frequency Filter:* We removed words that occurred in more than 80% of the documents, as they are most likely to be stopwords or other unimportant words which are not specific to any document.   
v. We also applied the max\_features = 2000 filter, to pick the top 2000 features that are picked on the basis of term frequency across the corpus. Picking only the top 2000 features for Topic Modeling gave us better results, which we discuss in the below section.

After Tokenizing the next step was to vectorize the documents using either *Boolean* or *Term Frequency* Vectorizer only, since the *TFIDF* vectorizer is not suitable for LDA algorithm. Since LDA is a probabilistic model, the IDF weighting of TF is not suitable for LDA. Hence, we used the Term Frequency Vectorization method, as the documents are short news headlines for which capturing multiple occurrences of a word becomes important and boolean is not suitable.

The topic models were assessed using our human judgements to produce topics which are meaningful. The *coherence* of these topics were assessed using *word intrusion* and *topic intrusion* concepts. The assessment of these topic models will be discussed in the upcoming experimental results section.

**3.1.2 Experimental Results**

Model 1: Initially, we started off with 20 topics and without the max\_features filter, that resulted in very poor topic coherence, as all the 20 topics had a random mixture of words because of which the topics could not be labelled. This model had 38,475 features in the vocabulary and it resulted in Log Likelihood & Perplexity scores as shown below:



Model 2: To improve the above performance, we modified the vectorizer by limiting the vocabulary to top 4000 features. This resulted in a better model performance with a lower word intrusion for the topics. The Log Likelihood score increased, and the Perplexity score decreased as desired and the scores are as shown below:



Model 3: To try and further improve the above performance, we modified the vectorizer by adding bigrams [using the ngram\_range = (1,2) parameter of the count vectorizer] to the feature set and also maintaining the top 4000 features limit. This resulted in a similar performance as model 2, as the Log Likelihood score decreased, and the Perplexity score also decreased. The decrease in Perplexity is desirable, but the decrease in Log Likelihood score is not desirable. The scores are as shown below:



Model 4: Since adding Bigrams did not improve the performance of the topic models, we decided to get rid of bigrams and tweak the max\_features parameter to further improve the model performance. We reduced the number of max\_features to 2000, but did not try to reduce it further, as further reduction of feature size could result in a loss of important linguistic information. This model was the best performing model with the out of all the 4 models as it had the lowest perplexity and highest log likelihood scores as shown below:



However, even this best performing model could produce only a couple of coherent topics and all the other topics were non coherent with a poor mix of out of domain words. Hence, we tried the NMF algorithm to try and produce better results. The performance of the NMF models were indeed much better than the LDA models as they produced highly coherent topics, which can be easily labelled. we shall discuss the NMF models in the upcoming section.

**3.2 NMF Algorithm**

Non-Negative matrix factorization is an unsupervised learning algorithm. NMF breaks down the matrix of vectorized data into two smaller matrices. A vectorized data matrix (d\*w) is represented as the product of two smaller matrices with dimensions (d\*t) and (t\*w). Here, d is the number of documents, w is the number of features and t is the number of topics that the algorithm tries to fit on this data set. The Algorithm takes multiple iterations to eventually match the product with the original matrix. Both the lower dimensional matrices are non-negative, i.e. non-they will have non-negative coefficients.

**3.2.1 Data Preprocessing**

The pre-processing of the data is summarized mainly in two points: 1] What are we counting and 2] How are we counting? First, we’ll answer the question: how are we counting? We will be using the NMF algorithm provided by the sklearn package. It allows us to use the tfidf vectorizer. So, the raw frequency counts of the words will be multiplied by the idf (inverse document frequency) values. Multiplying by the idf value reduced the weightage of the words that occur in most of the documents. Now, to answer the question: What to count? Unlike supervised classification, here we can’t really evaluate the performance of the model by calculating certain parameters. Hence, we will try different combinations of vectorization function arguments and by manually inspecting, try to decide the best possible combinations of the function arguments.

1] Baseline Model: After trying multiple combinations of different function arguments, we fix a baseline model and if any changes are to be made, the results will be compared to the results of this model. The arguments that we set for the baseline model are as follows:

i) *Maximum document frequency = 0.7*

ii) *Minimum document frequency = 5*

iii) *Removing the stop words = ‘english’*

All these parameters are explained in the LDA section. Adding to these 3, we defined two more arguments in the baseline model:

iv) *Token Pattern for features = ‘[a-z][a-z]+’*

This argument of the vectorization function is used to remove numbers, punctuation marks and special characters from the vocabulary. Also, a one letter token will also be removed from the vocabulary as they will not make sense for any of the topics in topic modelling.

v) Maximum features = 4000

It was observed that initially the vocabulary size for this dataset was more than 60000. Even after trying the different vectorization arguments, the vocabulary size was more than 30000. Since every document of our data is a news headline, which will have a maximum of 8 to 10 words in it, the vectorized data matrix will be very sparse. If the vectorized data is very sparse, one word feature can have a severe effect on topic modelling. This condition is like the overfitting condition in supervised learning techniques. Hence, after trying the different values for this argument, we decided to keep the 4000 important words as our features.

Note: All other arguments of tfidf vectorizer were not changed, so they will take the default values they are assigned to.

2] Baseline Model + Lemmatization

In any of the sklearn vectorizers we cannot use ‘tokenizer’ and ‘token\_pattern’ simultaneously. Hence, we need to perform lemmatization separately before passing the data to the vectorizer. So, we tokenized the news headlines using the nltk word tokenizer, then we used the ‘WordNetLemmatizer’ provided by the nltk package and finally detokenized the documents again using the ‘TreebankWordDetokenizer’ provided by the nltk package.

The reason for using lemmatizer instead of the stemmer is that lemmatizer replaces the trimmed word with its root (or a smaller) word. This step also helps in reducing the size of vocabulary. From a study[8], both stemming and lemmatization provide better performance in eliminating semantic duplicates. This allows more words relevant to the subject to be returned to the user, so that he can understand it better. Stemming, however, adds confusion to the tests, as it contains stems that are not actual words. Therefore, we use a lemmatizer before the vectorizer is applied on the data.

3] Baseline Model + Lemmatization + Additional Stop words

After studying the results of the previous model, it was observed that some words which are regularly used in english language and cannot be associated with a specific topic can be removed. So along with the English language stop words provided by sklearn’s ‘feature\_extraction’, we added these stop words:

[*'group', 'need', 'say', 'blog', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'needed', 'news'*]

4] Baseline Model + Lemmatization + Additional Stop words + Bigrams

In this model, along with the unigram features, we will be adding the bigram features. This will increase the size of vocabulary even more, and if setting max\_features to 4000 might make us lose some important feature words. Hence, we increase the max\_features limit to 6000.

Note: In all of these models, the vectorizer used is constant, i.e. sklearn tfidf vectorizer.

**3.2.2 Experimental Results**

The arguments given for the NMF algorithm were as follows:

i) *Number of Topics (n\_components) = 20*

These are the number of topics that we provided for all the models. Later it was observed that the actual number of topics is less than 20 as after the 13th or 14th topic, words from different domains were grouped into a single topic. So, this argument was changed for every model based on the results.

ii) *Set Seed (random\_state) = 1*

This argument was set so that we will be able to get the similar results when needed.

Next two parameters are the regularization parameters.

iii) *Regularization intensity (alpha) = 0.1*

The parameter controls the intensity of the regularization. Sinze, we have already limited the maximum number of features, we decided to keep this value a little low.

iv) *Regularization Type (l1\_ratio) = 0.5*

This parameter controls the balance between ‘l1’ and ‘l2’ type of regularization. Since, we were not really sure which regularization to use, we decided to keep this value 0.5 and maintain a balance between both the types.

*v) Initialization (init) = ‘nndsvd’*

This is the initialization parameter. ‘nndsvd’ stands for ‘Nonnegative Double Singular Value Decomposition’. Sklearn recommends to use this value if the number of topics is less than the number of features, which is true in our case.

Note: All other arguments of NMF were not changed, so they will take the default values they are assigned to.

1] Baseline Model:

As discussed before, the number of topics were reduced to 14 from 20. After making the changes, it was observed that the performance of NMF was much better than that of LDA already. It was then possible to assign the name to the topics. Also, one more thing was observed that word intrusion for every topic was almost negligible i.e. almost all the words in a topic were relevant to each other.

2] Baseline Model + Lemmatization

and

3] Baseline Model + Lemmatization + Additional Stop words

In the results for both these models, it was observed that results are getting better and better. The word intrusion was for the topics getting almost zero. Since, the results were pretty similar to the Baseline model, they are discussed in detail.

4] Baseline Model + Lemmatization + Additional Stop words + Bigrams

Adding the bigrams did not increase the performance of NMF by much, but it improves the sense of the top 20 words of every topic.

For e.g. ‘Face’ (unigram) occurred with words like ‘court’, ‘high’, ‘murder’, etc. This word here does not make any sense, but if we add bigrams, we get ‘face court’, ‘high court’, etc. as our important features which definitely makes sense and makes it easier to assign a name to the topics.

**3.2.3 Nomenclature of Topics**

We extracted the top 20 words for every topic and tried to name the topic by assigning a domain category to the topic. Below you can see few of the topics with assigned name and the top 20 words for that topic:

Topic 1] Personal Interview of People:

*(interview) (extended interview) (extended) (interview michael) (michael) (nrl interview) (interview david) (smith) (nrl) (david) (interview john) (scott) (john) (interview james) (james) (interview ben) (ben) (shane) (adam) (josh)*

Topic 3] Local Miscellaneous News:

*(govt) (wa) (water) (nsw) (report) (health) (change) (qld) (hospital) (cut) (job) (school) (sa) (government) (urged) (farmer) (boost) (service) (claim) (funding)*

Topic 4] Police Reports:

*(police) (probe) (investigate) (police investigate) (police probe) (officer) (search) (death) (missing) (police officer) (hunt) (police hunt) (arrest) (drug) (police search) (shooting) (police seek) (seek) (assault) (body)*

Topic 8] Judicial System News:

*(court) (face) (face court) (charge) (accused) (murder) (man face) (told) (case) (drug) (court told) (trial) (high) (sex) (child) (man court) (hears) (appeal) (court hears) (high court)*

Topic 9] Cricket:

*(australia) (day) (world) (south) (test) (cup) (australia day) (world cup) (south australia) (india) (england) (china) (ash) (pakistan) (cricket) (highlight) (live) (africa) (south africa) (test day)*

Topic 12] Accidents:

*(crash) (woman) (car) (killed) (dy) (car crash) (fatal) (road) (driver) (plane) (dead) (plane crash) (injured) (hit) (fatal crash) (road crash) (truck) (bus) (hospital) (highway)*

Topic 13] Business/Share Market:

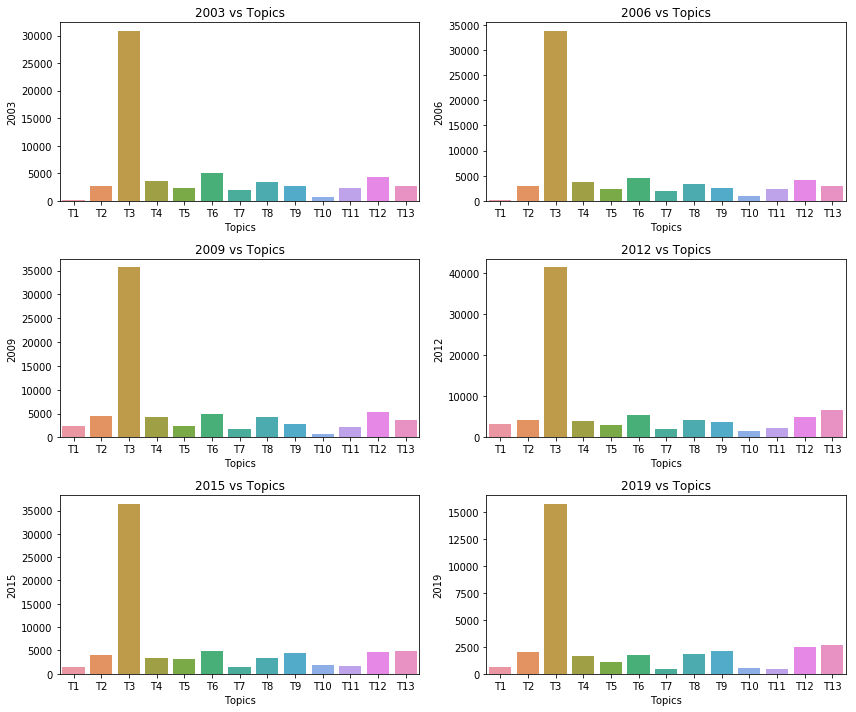
*(market) (australian) (abc) (share) (business) (share market) (open) (abc business) (analysis) (business market) (market analysis) (close) (local) (fall) (weather) (lead) (australian share) (australian open) (sport) (rise)*

The other topics were, Topic 2- Crime; Topic 5- International News; Topic 6 – Sports; Topic 7 – City planning; Topic 10 – Rural News; Topic 11 – City News.

**3.2.4 Topic Trend Analysis**

It is always interesting to find out the topics that were trending over the time. Given that the dataset that we have is a news headline data set along the publishing date of that article, we tried exploring the hot topics of every year.

The year was extracted from every date and was stored in a separate column ‘Year’. The vectorized data was transformed by the NMF algorithm, which returns one of smaller matrices of NMF algorithm (Documents vs Topics). The values in the matrix will be the non-negative values. Since these are not the



probability values, they do not add up to 1 for every document. Hence, we divide every value in the matrix with the sum of the values in every document.

We group all the values by the ‘Year’ column so the values are added for every year and then the bar graphs were plotted for the years 2003, 2006, 2009, 2012, 2015, 2019 to see what topics were trending over the years.

Topic 3 is the most dominant topic over all the years. Topic 3 is the local news of Australia; hence, this result is not a surprise at all. Although, an interesting observation is that Topic 9 (Cricket) and Topic 13 (Business/Share Market) were trending more in recent years as compared to the previous years.

**3.3 K-Means Document Clustering**

*k*-means clustering is a method that groups the data into clusters in which each observation belongs to the cluster with the nearest mean.

# Elbow method to find number of clusters

This method looks at the percentage of variance explained as a function of the number of clusters. If one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion". Although, this "elbow" cannot always be unambiguously identified. Percentage of variance explained is the ratio of the between-group variance to the total variance, also known as an F-test. A slight variation of this method plots the curvature of the within group variance.

**3.3.1 Data Preprocessing**

Before applying the k-means model, there was a need to prepare the data. The preparation of data is summarized below:

*Vectorization of data* with *Stemming:*

Started off with Tokenization and stemming of data to create a vocabulary (set of features) from the corpus and remove unwanted words. In order to reduce the size of the vocabulary the following filtering techniques were used:

*Removing Stop words*

*Minimum document frequency = 5*

*Regex tokenizer:* A RegexpTokenizer splits a string into substrings using a regular expression.

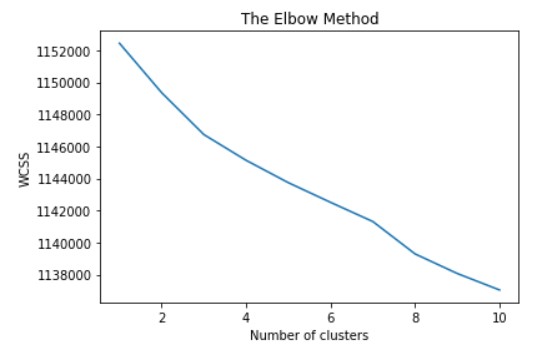
*Encoding:* Encoding was set to latin to define the accent of the document.

*Note*: We only used unigram for the model since bigrams and trigrams were giving poorer results.

For this dataset, we used term frequency–inverse document frequency (tfidf) because tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

**3.3.2 Experimental results**

K-means: For the optimal results and to get the best possible number of clusters, we used the elbow curve method. Below is the elbow curve we got for different clusters (ranging from 1 to 11)



When plotting the curve, there is a slight deviation in the line at k = 3 and again when k=7 and lastly at k= 8. Thus, we ran different models with different values for k and compared the results to find out which one of them made sense.

While applying the model, some of the functions were kept constant like:

*Max\_iter = 300:* One iteration is one pass over the entire data set. Thus 300 means 300 passes

*N\_init = 10:* Number of times the k-means algorithm will be run with different centroid seeds.

*Random\_state = 0:* used for initializing the internal random number generator. There is no need to set seed.

*Verbose = False:* Verbose means that it will output messages which could be useful for debugging and for understanding how the training is doing. We set it up as false.

*N\_jobs = 1:* It’s an integer, specifying the maximum number of concurrently running workers. N\_jobs = 1 means that no joblib parallelism is used at all, which is useful for debugging

*Model 1:* At k=3, when we ran the model, the topics we got make less sense. Words from various categories were grouped in a single cluster which decreased the efficiency of the model.

Result: Below are the words which were included in one of the 3 clusters formed

*[new, plan, say, council, govt, australia, win, fund, kill, report, australian, water, nsw, urg, court, warn, chang, year, crash, wa, health, qld, open, death, elect]*

*Model 2:* Model with k=8 gave even better results than the last two models. Text in each cluster made more sense.

Result: Below are the words included in one of the 8 clusters formed

*[polic, investig, probe, offic, search, car, hunt, death, arrest, shoot, drug, miss, crash, charg, driver, seek, assault, fatal, attack, murder, suspect, raid, woman, station, warn]*

Looking at the results, it can be said that the result became better with increase in the number of clusters. For ex. At k = 100, the categorization of the clusters will be more precise and accurate. But it will take more time to compute and generate results.

Thus, since our aim was to find the optimum number of clusters (lowest number) that will give us the best results without compromising the efficiency and time complexity of the model, we decided to choose k = 8 based on the graph and the results.

**4. Conclusion**

Upon comparing the performance of LDA & NMF models on this corpus, we could observe that the NMF models were performing much better than the LDA models in terms of Topic Coherence. Hence, we decided to proceed with the best performing NMF Topic model to produce 13 Topics and every document was assigned one of these 13 Topics.

**References**

[1] Blei, D. et al., (2003) Latent dirichlet allocation. *Journal of machine learning research* 3(4-5):993-1022  
[Online]<https://jmlr.csail.mit.edu/papers/volume3/blei03a/blei03a.pdf>

[2] [Online] <http://epubs.surrey.ac.uk/841381/1/Using%20LDA%20to%20Uncover%20the%20Underlying%20Structures%20and%20Relations%20in%20Smart%20City%20Data%20Streams.pdf>

[3] [Online] <https://asandeepc.bitbucket.io/courses/inls613_summer2019/lectures/08-lda_topic_modeling.pdf>

[4] [Online] <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>

[5] [Online] <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>

[6] [Online] <https://medium.com/ml2vec/topic-modeling-is-an-unsupervised-learning-approach-to-clustering-documents-to-discover-topics-fdfbf30e27df>

[7] [Online] <http://users.umiacs.umd.edu/~jbg/docs/nips2009-rtl.pdf>

[8] [Online]

<https://blog.bitext.com/lemmatization-to-enhance-topic-modeling-results#:~:text=The%20conclusion%20of%20our%20experiment,see%20how%20its%20results%20are>

1. Lower Perplexity score is preferred [↑](#footnote-ref-1)
2. Higher Log Likelihood score is preferred [↑](#footnote-ref-2)
3. Term Frequency-Inverse Document Frequency [↑](#footnote-ref-3)