Clustering the Unstructured Documents using Topic Modeling

Sravya Bhaskara*
MSIT Division, IIIT-Hyderabad,
Hyderabad, Telangana, India.
bhaskaravsravya@gmail.com

Arun Kumar Parayatham*
MSIT Division, IIIT-Hyderabad,
Hyderabad, Telangana, India
arunkp@gmail.com

Anisha Vulli*
MSIT Division, IIIT-Hyderabad,
Hyderabad, Telangana, India
anishavulli26061996@gmail.com

ABSTRACT

According to projections from IDC, 80 percent of the data generated will be unstructured by 2025. The generation of unstructured data geared up in recent years. Now, it is weighing greater importance while making crucial decisions in an organization when compared to structured data. There comes a saying "Unstructured data matters" .

The main focus of this project is to study the text documents which are unstructured and cluster them on topic similarity. The generation of text data is done through emails, articles, blogs, reviews, resumes, reports, etc. Topic Modeling is one of the emerging trends in the field of text analytics which is used to study unstructured data.

This report emphasizes the topic modelling algorithms – Word2Vec, LDA, LSA, NMF, Doc2Vec and the clustering analysis done on unstructured data. In this project we converted the text documents into word embeddings using the above algorithms. We also designed a method to tune automatically the hyperparameters appropriately to get optimum results. The obtained word embeddings are used for clustering the similar documents together which contain common "topics". The topic of a particular cluster is known by obtaining the words present in the documents which belong to the particular cluster. We have also compared the results produced by these algorithms using standard performance metrics. We have done experiments on text data from three different domains (Google News, Amazon Reviews and NIH research papers). We also developed a web interface called "Topic Modeling Toolbox" where the user can use these topic modelling methods to cluster unstructured data effectively.

ACM Reference format:
Sravya Bhaskara, Arun Kumar Parayatham, Anisha Vulli. 2020. Clustering the Unstructured Documents using Topic Modeling. In 8th ACM IKDD CoDS and 26th COMAD (CoDS, COMAD 2021), January 2-4, 2021, Bangalore, India. *ACM, New York, NY, USA, 5 pages.*

1 Introduction

The amount of text data being generated through websites and other social media is tremendous and a huge part of this data is mostly unstructured. Handling huge chunks of the data is a monotonous task. Topic modeling is an approach to cluster the text data based on its relevancy of the words that occurred in these texts, often called a "topic" [1][2]. Topic Modeling is an unsupervised machine learning and text mining technique which takes a set of untagged documents (text data) as input, observes the patterns within them and clusters these set of documents which have similar topics [3] [4]. The topic modeling extracts the main topics by observing the semantic relations among these text documents. Topic modeling is different from Topic Classification, which is a supervised technique and requires training. Topic modeling, on the other hand, requires no training and does not consider any tagged data [3].

The term topic modeling was first coined in the early 2000's. But it was limited only to some sets of applications where data was available [5]. Due to the exponential growth in data generation, topic modeling has expanded its approaches in various fields. Also, in recent times due to huge advancements in machine learning and text mining techniques, topic modeling became an emerging approach to study the untagged text data.

The goal of this project is to build a toolbox that shows the results of topic modeling algorithms for input text data. The contributions made in this paper are: (i) To obtain a detailed understanding of the topic models and, (ii) To build a web application using flask (python) where the user can use these models and obtain the clustering results for unstructured text data.

2 Approach

2.1 Data Gathering

For the purpose of the experimentation, we have taken unlabeled text from three different domains. We ran our toolbox for below three datasets and used the same as demo examples.

(1) **Google News**: Scraping the data from google news.
(2) **Amazon Reviews**: Amazon product reviews from Kaggle.
(3) **NIH data**: Research papers' abstract from the NIH website.

The user can either choose their own text data set or explore the topic modeling algorithms using the custom datasets provided in the toolbox as examples.

2.2 Data Analysis

We investigated the following algorithms for the analysis of topic modeling. The primary concern was to model all the below algorithms and to obtain the cluster results for each of them and also compare them.

- Word2Vector
- Latent Semantic Analysis (LSA)
- Doc2Vector
- Non-Negative Matrix Factorization (NMF)

* All authors contributed equally to this work.

CoDS and 26th COMAD (CoDS, COMAD 2021), January 2-4, 2021

- Latent Dirichlet Allocation (LDA)

While modeling an algorithm, we provided an option for users to choose their own set of hyperparameters respectively (this is limited to only few algorithms). During clustering analysis, the users can choose their own cluster value. If the users do not provide these inputs, optimum values are considered for better results.

3 Proposed Architecture

Toolbox pre-processes the provided untagged text data and uses above mentioned algorithms to find similar clusters based on conceptual topics and assign cluster labels for each document in the dataset. The clustering results are evaluated using standard clustering performance metrics. Functionality involves following process. The flow chart representation is shown in **Figure 1.**

3.1 Preprocessing the text data

The text data contains a list of documents. This is considered as raw data. The necessity of pre-processing is, raw data cannot be sent to the model as it adds up noise for stop words or punctuation marks. Cleaning the text, splitting the text into words, and handling the case is required. Below steps are followed in pre-processing using the NLTK [6] package. (1) Removing stop words. (2) Removing punctuation and handling the case. (3) Stemming

3.2 Algorithm and Clustering

The pre-processed text is considered as the corpus and the different algorithms are used to covert these words in the documents into respective vectors. The topic modeling algorithms consider the following criteria while producing the results: word frequency, the distance between words, minimum occurrences of words, etc.

3.2.1 Word2Vec: Word2Vec generates the vector representations of a particular word and thus vectors are calculated for the whole document. The vectors are calculated considering the semantic relations between the words such as the relationship between words based on past occurrences. The hyperparameters considered while building word2vec vectors are **Dimensional size, Window size, Min count, and, Sg** [8]. The generation of vectors is done using python library – genism [7]. In the word2vec model, we equipped different weighting strategies in order to get better results.

(1) **Word2vec with bigrams**: Combining two most frequently occurring words is considered as bigram generation [11]. This is useful to develop the text efficiently and also to get better classification results.

(2) **Word2vec with PCA**: The word2vec vectors are further given to the Principal Component Analysis (PCA) algorithm [9] to reduce the dimensionality of the vectors still retaining the useful information. The reduced dimension of the vectors is taken as two, in order to visualize the vectors in a two-dimensional space.

(3) **Word2vec with TF-IDF**: As word2vec algorithms weigh each word equally, the problem arises while dealing with documents because words do not equally contribute to the meaning of the sentence. Term Frequency – Inverse Document Frequency (TF-IDF) [9] would help to quantify the word based on its occurrence. The TF-IDF score of each word is combined with the word2vec vector to generate the results.

(4) **Word2vec with TF-IDF and PCA**: To visualize the reduced vectors along with combined TF-IDF scores.

Clusters: The word2vec vectors are generated using the above methods and thus are fetched to clustering techniques [10] (unsupervised) in order to cluster similar documents together based on their vector representation. We used the following clustering methods in order to generate the labels. The number of clusters required to separate the documents is either specified by the user or determined by grid search cv [12] (which indicates the optimal cluster value).

- Kmeans Clustering and Hierarchical clustering

3.2.2 Latent Semantic Analysis (LSA): LSA generates the similarity measure among the documents. The pre-processed documents are used to produce the document-term matrix using SVD (Singular Value Decomposition) [9]. The similarity matrix between all the documents is calculated from the document term matrix. The hyperparameters considered for generating the LSA similarity matrix: **N components and N iterations** [13].

Clusters: We used below clustering techniques to cluster the documents which are in the form of a similarity matrix [10].

- Spectral Clustering
- DB scan: Determines the number of clusters by itself.

3.2.3 Non-negative Matrix Factorization (NMF): The NMF [14] converts the vectors from higher dimensions to lower dimensions. For obtaining word embeddings, we used TF_IDF as the weighing technique. The TF-IDF matrix is a high dimensional matrix with order m * n. The input for the NMF function is the TF-IDF matrix and the number of required topics (k).

m - number of documents
n - number of words
k - number of topics (input)

The NMF function decomposes the TF-IDF matrix into two non-negative matrices. One with order n * k and another with order k * m (low dimensional). The fit transform methods are used for the model to obtain this conversion.

Clusters:

- Once the NMF model is fit, the method called argmax is used to get the labels of each document.

3.2.4 Doc2vec: Doc2Vec model converts the group of words collectively taken as a single unit into vectorized representation. The genism [7] model is used to obtain the vectors. The Doc2Vec method has an option to preserve the word order. The same weighing strategies as word2vec are also applied to doc2vec.

Clusters: The vectors are clustered using the following techniques and labels are obtained [10].

- Kmeans and Hierarchical

3.2.5 Latent Dirichlet Allocation (LDA): LDA is a generative probabilistic model which assigns the probabilistic values to both words and documents. The LDA model is built in two ways [15]. (i): topic per document model - Tells about the probability distribution of the topics in a particular document. (ii): words per topic model – Probability distribution of words in a particular topic. These distributions are in the form of matrices.

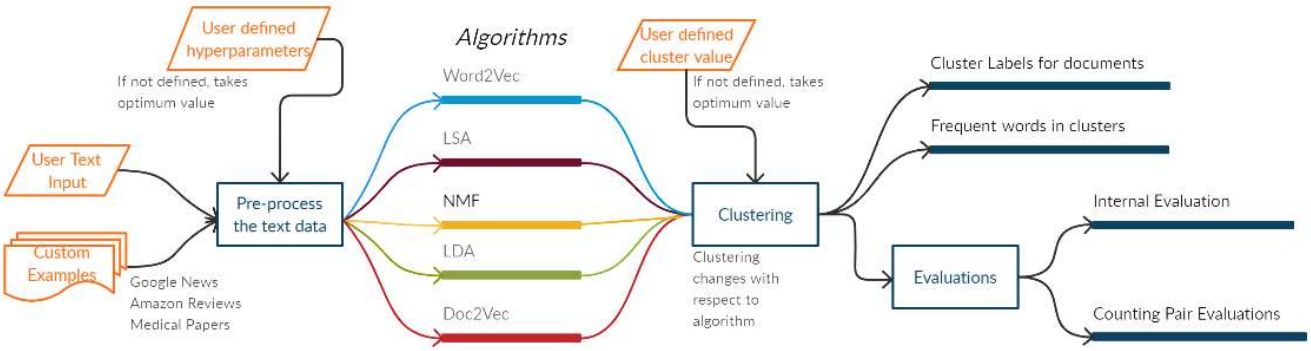

Figure 1: Proposed Architecture of Topic Modeling Application

Clusters: Based on the above two constructions, two types of clusters are obtained**.**

- The document cluster is formed using the probability values of the topics in a particular document. The topic which holds the highest probability for a particular document is considered to be the unique label of that particular document.
- The word cluster is obtained from the python library pyLDAvis [7], which gives the probability of the words occurring in each topic.

Note: The clustering techniques we used in this project differ from topic modeling algorithm as all of them do not produce similar representation of the word embeddings.

3.3 Evaluations of the clustering results

The evaluation of the clusters is done using two methods, internal and counting pair evaluations [16] [17]. Based on these evaluations, the algorithms which produce best results can be known.

3.3.1 Internal Evaluation: The cluster labels to each document in the data frame are evaluated independently using the internal cluster distance among the labels.

(1) **Silhouette Score**: The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters.
(2) **Calsinski Score**: Higher Calinski-Harabasz score relates to a model with better-defined clusters.
(3) **Davies Score**: Zero is the lowest possible score. Values closer to zero indicate a better partition.

3.3.2 Counting pair evaluation: This evaluation compares the labels of two clusters considering the pair of labels. Using these pair of labels from each cluster, the evaluation is made by assigning ones and zeros for the four different combinations.

- If a pair is assigned similar labels in both the clusters, n11 = 1.
- If a pair is not assigned as similar labels in both the clusters, n00 = 1.
- If a pair in one cluster is assigned similar labels but pair in another cluster is not assigned with similar, then n10 = 1. In contrast case, n01 = 1.

Using the above four values, the similarity measures such as Jacard Index, Rand Index, Fowlkes-Mallows Index, Mirkin Index, and Dice Index are calculated.

4 Experimentational Analysis

4.1 Implementation - Web Interface

We developed a web application where the user can upload files or explore demo examples and also set custom values for hyperparameters and cluster number. The web application was developed using flask and html (jinja). The algorithms are explained briefly and evaluation results are shown for all algorithms, so that user can pick up the best working algorithm. Each algorithm produces the following output:

- The pie chart to visualize the count of documents falling into each cluster in percentages.
- The frequently occurring words in each cluster.
- The internal evaluation table which computes the scores for all the clustering techniques used in that algorithm.
- The counting pair evaluation which compares all the clustering techniques used for the algorithm.
- The excel file which has original documents along with the cluster labels they fall into, for each technique, can be downloaded.

4.2 Observations

By using the toolbox for three different datasets, the key takeaways while we observed the behavior of models are:

- The word2vec with PCA algorithm was showing better performance in all the three cases but with different clustering techniques as shown in the **Figure 2**.
- For a few documents, the LDA algorithm was assigning equal probabilities to all the topics.
- In LDA, clusters are intersected if they contain words that are common in all of them.
- In counting pair similarity measure, the word2vec with bigrams using kmeans and hierarchical are having a high Jaccard coefficient, indicating they are producing similar clusters.

Google News

Name	Silhoutte Score	Calinski Score	Davies Score
NMF	0.536777	152.957687	0.664509
LSA with spectral	0.612104	474.264730	0.446631
LSA with DBScan	0.529089	401.613531	1.117394
D2V	0.266196	125.182972	1.148572
LDA	nan	nan	nan
word2vec_Kmeans	0.191352	14.342364	2.428078
word2vec_pca_Kmeans	0.461887	181.430687	0.755492
word2vec_tfidf_Kmeans	0.055535	6.115254	3.425331
word2vec_tfidf_pca_Kmeans	0.379431	105.799500	0.830727
word2vec_Hierarchical	0.236375	17.855294	1.370246
word2vec_pca_Hierarchical	0.630252	213.367361	0.364926
word2vec_tfidf_Hierarchical	0.039323	6.514960	3.442678
word2vec_tfidf_pca_Hierarchical	0.376816	98.973840	0.807188

Amazon Reviews

Name	Silhoutte Score	Calinski Score	Davies Score
word2vec_pca_Hierarchical	0.572214	3752.421830	0.493938
word2vec_Kmeans	0.563306	4045.035000	0.544153
word2vec_pca_Kmeans	0.562564	3997.087314	0.499263
LSA with spectral	0.543716	2978.822895	0.504672
word2vec_Hierarchical	0.538544	3817.582831	0.542560
word2vec_tfidf_pca_Kmeans	0.533752	2087.028960	0.510778
word2vec_tfidf_Kmeans	0.527204	2057.936470	0.518328
word2vec_tfidf_pca_Hierarchical	0.520622	1863.460791	0.502149
word2vec_tfidf_Hierarchical	0.516349	1934.602404	0.512762
LSA with DBScan	0.456754	64.452408	0.367778
NMF	0.423942	257.524499	0.863903
D2V	0.073415	37.813100	2.299675
LDA	nan	nan	nan

Research Papers

Name	Silhoutte Score	Calinski Score	Davies Score
word2vec_pca_Kmeans	0.687440	1925.121612	0.380930
word2vec_pca_Hierarchical	0.675967	1787.305888	0.351633
LSA with spectral	0.583011	131.491454	0.331188
LSA with DBScan	0.561117	99.150978	1.144474
word2vec_Kmeans	0.536275	900.813816	0.629769
word2vec_Hierarchical	0.535168	890.125878	0.611690
word2vec_tfidf_pca_Kmeans	0.531389	531.690611	0.566234
word2vec_tfidf_pca_Hierarchical	0.523819	519.986828	0.570407
word2vec_tfidf_Kmeans	0.469259	487.928673	0.633399
word2vec_tfidf_Hierarchical	0.459026	459.511116	0.611768
NMF	0.414275	149.725392	0.831849
D2V	0.095772	27.275560	2.216925
LDA	nan	nan	nan

Highest to lowest - Yellow to Orange

Figure 2: Internal Evaluation Results of Three Different Text Data Sets

	2	0	3	1
0	(-, 77)	(technolog, 21)	(economi, 35)	(hollywood, 18)
1	(coronaviru, 39)	(latest, 5)	(news, 34)	(news, 4)
2	(ago, 27)	(news, 5)	(econom, 14)	(award, 3)
3	(hollywood, 27)	(scienc, 4)	(latest, 13)	(break, 2)
4	(technolog, 25)	(definit, 3)	(trade, 10)	(movi, 2)
5	(news, 24)	(use, 3)	(world, 6)	(industri, 2)
6	(covid19, 22)	(topic, 3)	(indian, 6)	(latest, 2)

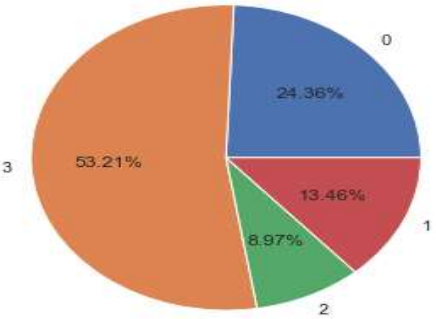

Figure 3: Document Distribution and Frequent Words (top 6) in Cluster for Word2vec with PCA using Hierarchical Clustering

- Most of the frequent words are confined to a particular cluster but few words which are generic, are falling into all clusters. Example of words: latest in Google news, product in Amazon reviews.
- The words with frequency 1 are also present in all the clusters and they were also contributing to cluster separation.
- The topic contained in each cluster can be analyzed from frequent words table as shown in **Figure 3**. Example: Cluster 3 has economy, trade. Cluster 0 has technology and science, Cluster 1 has Hollywood and awards. and Cluster 2 has corona, covid-19.

4.3 Future Scope

Advancements in this project can be made by considering following factors:

- The input data can not only be text files but also unlabelled audio, video, and image files.
- In word2vec, the words we have generated are bi-grams, we can also make use of tri-grams and n-grams.
- In NMF, we confined to standard set of hyperparameters, but they can be changed based on the applications.
- In LDA, the same document can be put into different clusters if they belong to two or more clusters.
- In all methods, once the cluster centres are fixed, each document can be given a score that is relevant to belonging to a cluster. (based on the document distance to cluster centres)
- The frequently occurring words in each cluster can be useful to generate a summary of the topic or analysis of the documents for the respective cluster.
- The cluster labels obtained from all the algorithms can be ensembled into unique result. (using enssembling methods).

5 Conclusion

In this paper, we investigated and explained how the topic modelling algorithms can be used for clustering unstructured text data. We built a web application (Topic Modeling Toolbox) for clustering unstructured text data based on topic similarity. The application provides the cluster labels and performance evaluation results as output through web interface. It also provides an option for user to choose the data set of their own and experiment the tool box. This toolbox can find its application in various domains like analyzing product/service reviews, extracting key features of a product, clustering similar resumes, etc. We are interested in continuing this research by exploring all the topic models available in text analytics.

References

[1]https://solutionsreview.com/data-management/80-percent-of-your-data-will-be-unstructured-in-five-years/
[2]https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/
[3] https://monkeylearn.com/blog/introduction-to-topic-modeling/
[4]https://www.analyticsvidhya.com/blog/2018/10/stepwise-guide-topic-modeling-latent-semantic-analysis/
[5]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5028368/#:~:text=The%20origin%20of%20a%20topic,LSI)%20(Deerwester%20et%20al.&text=Based%20on%20LSI%2C%20probabilistic%20latent,proposed%20by%20Blei%20et%20al.
[6] https://pythonspot.com/nltk-stop-words/
[7] https://radimrehurek.com/gensim/auto_examples/index.html
[8] https://radimrehurek.com/gensim/models/word2vec.html
https://medium.com/@ranasinghiitkgp/featurization-of-text-data-bow-tf-idf-avgw2v-tfidf-weighted-w2v-7a6c62e8b097
[9]https://shichaoji.com/2017/02/18/unsupervised-learning-2-dimension-reduction-pca-tf-idf-sparse-matrix-twitter-posts-clustering/
[10] https://scikit-learn.org/stable/modules/clustering.html
[11] https://www.tidytextmining.com/ngrams.html
https://medium.com/@manjunathhiremath.mh/identifying-bigrams-trigrams-and-four-grams-using-word2vec-dea346130eb
[12] https://scikit-learn.org/stable/modules/grid_search.html
[13]https://thesai.org/Downloads/Volume6No1/Paper_21-A_Survey_of_Topic_Modeling_in_Text_Mining.pdf
[14] https://mlexplained.com/2017/12/28/a-practical-introduction-to-nmf-nonnegative-matrix-factorization/
[15]https://ai.stanford.edu/~ang/papers/nips01-lda.pdf
[16]https://en.m.wikipedia.org/wiki/Cluster_analysis#Evaluation_and_assessment
[17]https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.cluster