Evolutionary Intelligence (2023) 16:1667–1677
https://doi.org/10.1007/s12065-023-00825-3

SPECIAL ISSUE

Text embedding techniques for efficient clustering of twitter data

Jayasree Ravi[1] · **Sushil Kulkarni**[1]

Received: 27 February 2022 / Revised: 20 November 2022 / Accepted: 14 January 2023 / Published online: 7 February 2023

Abstract
World wide web is abundant with various types of information such blogs, social media posts, news articles. With this type of magnitude of online content, there is a need to deeply understand the insights of it in order to make use of the information for practical applications such as event detection, polarity, sentiment analysis and so on. Natural Language Processing (NLP) is the study of such information which is used for text classification, sentiment analysis, clustering of similar text. NLP makes use of linguistic knowledge and build machine learning models to analyse textual information. NLP finds its way in various applications like classification of online review into positive and negative without actually reading the reviews and feedback. For text analysis, there should be a way to quantify the text based on its frequency of occurrence, correlation with neighbouring words, contextual similarity of words, etc. One such way is word embedding. This study applies various word embedding techniques on tweets of popular news channels and clusters the resultant vectors using K-means algorithm. From this study, it is found out that Bidirectional Encoder Representations from Transformers (BERT) has achieved highest accuracy rate when used with K-means clustering.

Keywords Text embedding · Tf-idf · Word2Vec · GloVe · BERT · Twitter analytics

1 Introduction

Word embedding is a technique of representing a particular word as a vector in a pre-defined vector space. Each word is embedded as one vector and these vector learning works similar to a neural network. That is the reason it is one of the applications of deep learning. The best approach of incorporating word embedding is to use a dense vector of low dimensional nature instead of sparse vector of high dimensional nature. Most of the neural network toolkits do not perform well with sparse vectors of high dimensions. The main advantages of using a dense vector are generalization and computational complexity. If some features provide similar hints in corpus, then it is better to provide word embedding mechanism that captures these similarities. Clustering algorithm is one of the important algorithms in discovering patterns in a text corpus. It helps in finding statistics in the corpus and in acquiring the underlying patterns that are present in the corpus. This study discusses various text embedding techniques, applies them for vectorization of corpus and clustering the corpus. To conduct this study, tweets of popular Indian news channels have been collected during a particular period.

2 Literature review

In the study proposed in Ref. [1], the authors have used Term Frequency-Inverse Document Frequency (TF-IDF) and cosine similarity measure to find similarity in patient support forums. In Ref. [2], the authors have classified the research articles using TF-IDF and Latent Dirichlet Allocation (LDA). Sentiment analysis of textual data using TF-IDF is handled by the authors of [3] and [4]. In the study conducted by the authors of [5], they have proposed an unsupervised algorithm-Edge-Weight model which can be adapted to the changes of WordNet and feature-based semantic similarity technique for Wikipedia links that focusses on improving the accuracy of TF-IDF method. One more

Sushil Kulkarni have contributed equally to this work.

✉ Jayasree Ravi
jayasree.ravi@mithibai.ac.in

Sushil Kulkarni
sushiltry@gmail.com

[1] Department of Computer Science, University of Mumbai, Kalina, Mumbai, Maharashtra 400098, India

application of TF-IDF is mentioned by the authors of Ref. [6]. In this, the authors discuss how chatbots interprets the user queries based on keywords and respond based on the keywords. Chatbots store the keyword and query responses in backend and retrieve it after comparing the text. N-gram, TF-ÏDF and Cosine similarity are used for this purpose. with reference to handling social media data also, TF-IDF is used. In the study Ref. [7], the authors aim to suggest tourist places to visit using user generated content that is available in the web. The main aim of this paper is to discover a set of keywords that are sematically meaningful and that can characterize a tourism place. The authors of Ref. [8] have used the Siamese Long Term Short Term Memory (LSTM) model to find the underlying semantics of short documents. Three distance measures are used to evaluate the performance of them —Manhattan Distance, Euclidean Distance and Cosine Distance. To vectorize the sentences, Doc2Vec technique is used by the authors. In Ref. [9], the authors have combined the techniques of cosine similarity measure with Latent Semantic Analysis to find out the reasons for variation in sentiments present in twitter data. To achieve this, the authors have two novel methods of Latent Dirichlet Allocation (LDA)-Foreground and Background LDA and Reason Candidate and Background LDA. In the method proposed in Ref. [10], the authors have used TF-IDF technique and Cosine Similarity to analyze and compare the contents of blog documents posted by bloggers on social media. Regarding BERT embeddings, there are many studies conducted on the applications of BERT—for identifying cyberbullying [11] and sentiment analysis of investors in stock market [12].

There have been some studies conducted to analyze the performances of different word embedding techniques. In Ref. [13], the authors have compared the performances of Global Vectors for word representation(GloVe), Bidirectional Encoder Representations from Transformers(BERT) and Robustly Optimized BERT(RoBERTa) in detecting plagiarism in articles and have concluded that BERT has outperformed the other two methods. The authors of Ref. [14] have used TF-IDF with other deep learning approaches of word embeddings for a clinical domain problem and have found out that TF-IDF combined with Support Vector Machine(SVM) has outperformed other word embedding models like GloVe and Word2Vec models. In Ref. [15], the authors have examined the performance of BERT model combined with different clustering algorithms. They have also done the comparison of TF-IDF and BERT and have concluded that BERT is better than TF-IDF with respect to many metrics.

From the above literature review, it is clear that there has not been any research article that have done extensive study of text embedding methods and how they perform on social media data such as twitter. The studies that have been conducted so far related to text embedding techniques deal with structured data such as Wikipedia and research articles. The literature that have used twitter data have handled limited applications such as sentiment analysis, finding the polarity of tweets and topic modelling. This paper attempts to do an extensive study of various text embedding techniques and choose an appropriate one for vectorization of twitter data. This study further performs the clustering of the data using K-Means clustering algorithm.

3 Types of word embedding

To demonstrate the types of word embedding, the following documents are considered.

S1—'Raj like Mango'
S2—'Raj do not like cheese'
S3—'She does not like to go out'

3.1 One-hot vector

This is the simplest method of word embedding. If there are m unique words in a corpus, then those words can be converted to $1 \times m$ vectors. For instance, if we have four words—"Delhi", "Mango", "Blue", "Cheese", then these words can be represented as [0, 0, 0, 1], [0, 0, 1, 0], [0, 1, 0, 0], [1, 0, 0, 0] respectively. But there are few challenges in this model. When we have m unique words in the corpus and p number of words in the corpus, the resultant vector will have a dimension of $p \times m$. To illustrate this, we have assumed the above sentences. The unique words found in the above corpus is ['cheese', 'do', 'does', 'go', 'like', 'mango', 'not', 'out', 'raj', 'she', 'to'] and the number of unique words is 11. Total number of words present in the corpus is 15. So, the dimension of the resultant vector is 15×11 as given below.

[[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]]

3.1.1 Strengths

1. This technique is used where the corpus is less.
2. there is no need to find the contextual relationship between words.

3.1.2 Shortcomings

1. The memory usage leads to exponential increase in algorithm complexity which in turn leads to the curse of dimensionality.
2. Secondly, this model is specific to applications making transfer learning almost impossible. Word embedding of one application cannot be used on another application with a different text corpus.
3. Lastly, this model does not capture the contextual meaning of words present in the corpus. Ideally, similarity ("Mango", "Guava") should be greater than similarity ("Mango", "Delhi"). This aspect is missing in this model.

3.2 Bag of words (BoW)

This technique is mainly used in classification algorithm in Natural Language Processing. A Bag is synonymous to class where when there are specific set of words in a sentence, it belongs to a class. When two sentences have same set of words, both of them belong to the same class. Each sentence is represented by a row with n number of columns, where m is the number of unique words in the corpus. This model is illustrated using the sample dataset considered above.

Features—['cheese', 'do', 'does', 'go', 'like', 'mango', 'not', 'out', 'raj', 'she', 'to']

As we observe the above example, each sentence in the corpus is converted into a vector.

3.2.1 Strengths

1. This is easy to understand. Each word in the corpus acts as a feature.
2. Each document in the corpus is converted to a vector of size equal to number of words in the document.
3. It not only counts the number of words in the document and the number of unique words in the document but also identifies where each word appears in the corpus.

3.2.2 Shortcomings

1. This model also creates a sparse vector which occupies a considerable memory space.
2. Though this model is better than One-Hot Vector in terms of the size of the vector, the size of the vector increases as the size of the data set increases.

3.3 N-gram model

N-gram model is based on conditional probabilistic approach. For instance, if $N = 3$, this model calculates the probability of the occurrence of a term based on two preceding terms. This model breaks the sentences into chunks (grams) instead of splitting the sentences into individual words like BoW model. BoW model is n-gram model with gram = 1. The point to be noted is the proximity of the source word and target word is not important. The demonstration using the same corpus given below explains this model with gram = 2.

Resultant Matrix [[00001100100][11001010100][00111011011]]

Table 1 Tf-idf values of sentence 1

Word	Tf	idf	Tf-idf	Norm-tf-idf
Raj	1	1.1761	1.1761	0.55
Like	1	1	1	0.47
Mango	1	1.477	1.477	0.69

Features—['do not', 'does not', 'go out', 'like cheese', 'like mango', 'like to', 'not like', 'raj do', 'raj like', 'she does', 'to go']

$$idf(w) = 1 + \log_e \frac{n}{df(w)} \tag{3}$$

$$idf(w) = \log_e \frac{n}{df(w)} \tag{4}$$

where n is the total number of documents available, w is the word for which idf has to be calculated and $df(w)$ is the number of documents in which the word w appears. tf-idf is the product of $tf(w)$ and $idf(w)$ which were calculated in Eqs. (1) and (3) respectively.

Resultant Matrix [[00001000100][10010011000][01100110011]]

3.3.1 Strengths

1. This model is able to find the context of a term to some extent by considering the presence of neighbouring term.
2. This model can easily be generalized by modifying the value of N.

3.3.2 Shortcomings

1. This model also has the same problem as with BoW approach—it creates a sparse vector and this also suffers from dimensionality problem.

3.4 Term frequency—inverse document frequency (Tf-idf)

This is a statistical method used to calculate the importance of a particular word in a document. This method works similar to one-hot encoding but there is a value that corresponds to each word in the document instead of 1. That value is the product of term frequency and inverse document frequency measures. The example below shows the working of tf-idf technique. In all the above sentences, the word "like" appears in all sentences, "Raj" occurs in two sentences, "not" occurs in two sentences. Term Frequency (tf) can be defined as any one of the following. If w is the word for which tf has to be found, p is the number of times, the word w occurs in the sentence and s is the number of words in the sentence,

$$tf(w) = p \tag{1}$$

$$tf(w) = \frac{p}{s} \tag{2}$$

Inverse Document Frequency (idf) can be calculated using as any one of the following formulae mentioned in Eqs. (3) and (4).

$$tf - idf(w) = tf(w) \times idf(w) \tag{5}$$

To avoid longer sentences in the corpus dominating the shorter ones, normalization of each row in the sparse matrix is done using Euclidean norm as shown in Eq. (6).

$$norm - tf - idf(w) = \frac{tf - idf(w)}{\sqrt{\sum_{i=1}^{s} tf - idf(i)^2}} \tag{6}$$

For sentence 1, the demonstration of calculating tf-idf of each word is shown in the Table 1.

From the last column, it is evident that the value of tf-idf is inversely proportional to the number of times it appears in the corpus. After applying tf-idf to all the sentences, they are represented as a matrix as below.

Features—['cheese', 'do', 'does', 'go', 'like', 'mango', 'not', 'out', 'raj', 'she', 'to']

Resultant Matrix

[[0.0.0.0.0.4250.7200.0.0.5480.0.]]
[[0.5340.5340.0.0.3150.0.4060.0.4060.0.]]
[[0.0.0.4110.4110.2430.0.3120.4110.0.4120.411]]

The dimension of the resultant matrix of BoW, n-gram and tf-idf will be $n \times m$, where n is the number of sentences in the corpus and m is the number of unique words in the corpus.

3.4.1 Strengths

1. This model not only identifies the features but also the frequency of number of times each feature appears in the document. It considers both term frequency and inverse document frequency for the vectorization of each term.

3.4.2 Shortcomings

1. As the size of the corpus increases, the number of features also increases.
2. The second one is that the contextual relation of the word with other words in a sentence is not considered. They just count the frequency of occurrence of the words.
3. Like previous models, this also creates a sparse vector which escalates the space complexity of the algorithm.

3.5 Word2Vec

Word2Vec [16] is a statistical method of word embedding from a corpus of text. It is a shallow 2-layer neural network which takes words as input and converts them into vectors which are present in multidimensional vector space. The positioning of vectors are done in such a way that the words with the same context are positioned in close proximity to each other. So, this model is efficient in identifying the semantics of words. This model is a standard for modelling pre-trained embedding of words from text corpus. It involves analysis of word vectors and applying vector math on the representation of terms. For example, by subtracting the woman-ness from princess and adding man-ness to it results in the word prince. This captures the analogy "princess is to prince as woman is to man". These representations are efficient in capturing both syntactic and semantic similarities found in the corpus. The complexity of the model is represented by Eq. (7).

$$O = E \times T \times Q \tag{7}$$

where, E is the number of epochs, T is the word count in the corpus and the factor Q depends on the two variations of Word2Vec model—Continuous Bag of Words (CBOW) model and Skip-gram model. In the first method, given its context, this model predicts the word. The input of the model is the neighbouring words (both history words and target words) and the output is the word which suits the context of the neighbouring words. The number of neighbouring words depends on the window size set by the user. In the second model, the goal of the neural network is to learn the weights of the hidden layers which are actually the vector representation of words instead of focussing on the input and output of the network. The task for this model is to find the neighbouring words given a current word. The number of neighbouring words to be predicted is the window-size, which is a hyper-parameter. In CBOW model, Q is defined by the Eq. (8).

$$Q = N \times D + D \times \log_2 V \tag{8}$$

where N is the count of previous words, D is the word representations and V is the size of the vocabulary. In Continuous-Skip gram model, Q is represented by the Eq. (9).

$$Q = C \times D + D \times \log_2 V \tag{9}$$

where C is the window size. If $C = 4$, for each word that is trained, then a random value R where $1 < R < C$ is chosen and R words from history and future are used as labels. So, with the input word, we will predict $R + R$ words as output. The CBOW model does the word embedding by performing context-based prediction of current word. The continuous skip-gram model predicts the neighbouring words of the given current word.

3.5.1 Strengths

1. This model is one of the pioneer models which is built on a shallow neural network and uses pre-trained word embeddings.
2. It attempts to find the semantics of a word based on the neighboring words.
3. The size of the word vectors is much smaller than the previous models.

3.5.2 Shortcomings

1. The efficiently of the model largely depends on the parameters that need to be set in the neural network—size of the gram or the window, vector's target size, number of iterations and so on.
2. This model may not be suitable for classification models with multiple classes and uses softmax function.

3.6 Global vectors for word embedding (GloVe)

GloVe [17] is an unsupervised learning model also represents words into vectors. Training is done on word-to-word cooccurrence statistical information found in corpus and the vector representation of words reveal interesting linear substructures in the vector space. Glove makes use of statistical information present in the corpus and applies weighted least squares method that trains on word counts with global co-occurrence. The model showcases how co-occurrence probability can be used to extract certain underlying meanings found in the corpus. To explain this concept, consider two words a and b from the corpus. The relationship of these two words can be verified with the help of their ratio of co-occurrence probabilities with probe words, c. For c related to a but not b, the ratio $\frac{P_{a|c}}{P_{b|c}} > 1$. Similarly, for c related to b but not a the ratio $\frac{P_{a|c}}{P_{b|c}} < 1$. If c is either related to both a and b

or not related to both a and b, the ratio $\frac{P_{a|c}}{P_{b|c}}$ is close to 1. So, this suggests that the starting point of learning word vectors is to find the ratio of probabilities of co-occurrence of words. The general form of the model is represented in the Eq. (10).

$$F(w_a, w_b, w_c^{\#}) = \frac{P_{a|c}}{P_{b|c}} \tag{10}$$

where $w \in \mathbb{R}^d$ are word vectors and $w^{\#} \in \mathbb{R}^d$are probe word vectors in the corpus.

3.6.1 Strengths

1. GloVe method is based on calculating the co-occurrence counts of each term based on the entire corpus. It considers global statistical information to create word vectors.
2. It creates word vector space which preserves the semanticity of the corpus.

3.6.2 Shortcomings

1. The memory needed to store the co-occurrence matrix is large.
2. As in Word2Vec, setting the parameters for the model is critical. When the parameters are updated, the co-occurrence matrix will be re-created.
3. This model does not identify antonyms as the term and its corresponding antonym are located in close proximity in the vector space. So, this model may not be used for certain applications like polarity analysis of textual data.

3.7 Doc2Vec for document embedding

Doc2Vec [18] is an extension of Word2Vec model—Word-2Vec model intends to learn word vectors of high quality whereas Doc2Vec model is an unsupervised learning which aims to learn vectors for sentences of variable length. The word vectors are shared among sentences but each paragraph will have a unique paragraph vector. This model is capable of constructing uniform-sized vectors for paragraphs or sentences of any length. This distributed memory model uses the indirect result of capturing semantics while performing word vectors prediction operation. In this framework, each sentence and each word in the corpus are represented by a column of a matrix D and matrix W respectively. The word matrix W is common for all paragraphs. The model is trained using Backpropagation and Stochastic Gradient Descent is used to tune the parameters. Softmax activation function is used in this model.

If there are N tweets in the corpus, M total words in the corpus and if each tweet is mapped to p dimensions and each word is represented with q dimensions, then the model will have $N \times p + M \times q$ parameters (softmax function parameters not included). The similarity between word vector model and paragraph vector is shown below in the Eqs. (11) and (12). Given a set of words $w_1, w_2, w_T \in W$, the aim of word vector model is to increase the log probability.

$$\frac{1}{T}\sum_{t=k}^{T=k} logp(w_t \mid w_{t-k}, ..., w_{t+k}) \tag{11}$$

$$p(w_t \mid w_{t-k}, ..., w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{12}$$

The difference between word vector and paragraph vector lies in finding the value of h in Eq. (13). In the former, h is calculated by summing up or averaging the values of word vectors whereas in the latter, h is calculated from matrices D and W.

$$y = b + U \times h(w_{t-k}, ..., w_{t+k}; W) \tag{13}$$

3.7.1 Strengths

1. This is a generalization of Word2Vec model. It represents each sentence into a vector. This model creates equal-sized vectors irrespective of the number of terms in that sentence.

3.7.2 Shortcomings

1. Setting the vector size, which is one of the parameters is crucial in feature extraction. If the vector size is set to a bigger number, then there needs to be a good amount of training data. If the vector size is too small, then the representation of the corpus is not efficiently done.

3.8 Bidirectional encoder representations from transformers (BERT)

BERT [19] model is based on a deep neural network called transforms where the conditioning is performed on both sides of the context between layers, due to which tuning of weights can be easily done in accordance with the type of data and problem without changing the underlying architectural model. BERT is built on two algorithms—Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM predicts the masked words using the left context and right context of the word and NSP is used in word representations. The pretrained model used in this study is all-miniLM-L6-v2. This has the optimum quality of document representation with 5 times faster than other pretrained models [20].

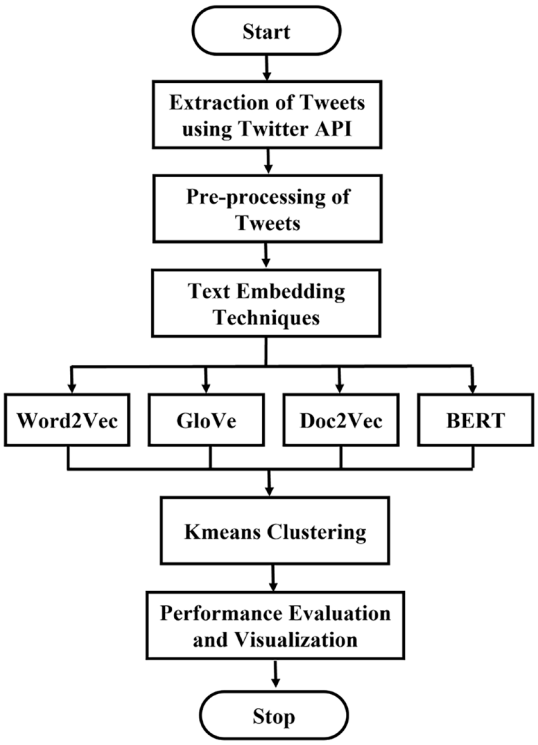

Fig. 1 Proposed model

3.8.1 Strengths

1. This is a classic example of using transfer learning in deep learning network. It has the capacity of managing big data.
2. As this model is able to tap the contextual information of the corpus, this can be used in various applications other than sentiment analysis such as finding semantic similarity of documents, topic modelling, text summarization and so on.

3.8.2 Shortcomings

1. This model is slow and expensive due to its computational complexity.
2. More weights are involved in this deep neural network model and they all have to be tuned efficiently.

4 Methodology

We extract tweets from Twitter API and pre-process the tweets. The cleaned tweets are transformed into vectors by adapting different text embedding models. The vectors are clustered using K-means clustering and the resultant clusters are compared using different performance metrics.

The methodology applied in this study is represented in the Fig. 1.

4.1 Collection of tweets and pre-processing of tweets

We have considered the twitter handles of 5 popular news channels of India and collected tweets from 17th August 2021 to 24th August 2021 relevant to regime change in Afghanistan and subsequent takeover by Taliban. Tweets were collected from 10th January 2022 to 20th January 2022 relevant to the spread of new variant of COVID-19 virus - Omicron. Tweet id, Timestamp of the Tweet and Tweet text were collected. The upper limit for downloading tweets from each user id through Twitter API is 200. After collecting the tweets, they were preprocessed to make it suitable for the study. The following preprocessing steps were carried out on the tweets.

1. Converting tweets to lowercase
2. Removal of Punctuation marks
3. Removal of Hyperlinks
4. Removal of non-English words
5. Removal of Retweets

After performing the above preprocessing steps, there were 2013 tweets that were considered for our study.

4.2 Document embedding and vectorization of cleaned tweets

Tweet corpus has to be converted into vectors before they are used in clustering. In this study, we have compared the performance of four document embedding techniques-CBOW Word2Vec, GloVe method, Doc2Vec model and BERT model in terms of their clustering quality. To perform the procedure in Word2Vec[1], we have used a pre-trained embedding developed by google. The googlenews word vectors were trained on a corpus available with google. They were trained using CBOW Word2Vec model. The corpus on which the embedding is developed is around 100 billion words. Three million words and phrases have been taken into account, each represented as a 300-dimensional vector. As we all know, each document in the corpus is of different word length. To bring in uniformity in the length of the document vector, padding of each document to a specific size m is performed, so that each document is uniformly represented as $m \times 300$ dimensional vector. The words in the corpus are vectorized using the pre-trained embedding. As we have done the padding of each document to a size of m and the number of tweets is n, the entire corpus will be of

[1] https://github.com/jaima77/Text-Embedding.git.

size $n \times m \times 300$. Next step is to represent each document in the corpus into vectors. We can either find the sum or average of every word vector corresponding to that document. This cannot handle variable-length documents. One efficient way to perform this transformation is to take the weighted average of each word vectors using the weights of tf-idf. After incorporating this method, the corpus is converted into $n \times 300$ dimensional vector, which is used in clustering. The next method considered in this study is GloVe method. The steps for this method is same as Word2Vec method. But, instead of using pre-trained embedding provided by google, pre-trained Glove embeddings provided by Stanford GloVe embedding of 100 dimensions is considered for this study. Similarly in Doc2Vec and BERT methods, the corpus is represented as vectors of $n \times 100$ and $n \times 768$ dimensions respectively. The resultant document vectors created through different embedding techniques are clustered using K-means clustering technique.

5 Clustering metrics used

We have considered both External clustering metrics—Accuracy, F1-Score and Adjusted Rand Score and internal clustering metrics— Silhouette Score and Davies Bouldin Index for the evaluation of document embedding techniques and subsequent KMeans clustering. External clustering metrics need ground truth to which the clustering results are compared. To facilitate the performance evaluation, we labelled every tweet based on which topic it belonged to - whether it is about Afghanistan issue or about Omicron virus. Internal clustering metrics is concerned with the inter-cluster distance and intra-cluster distance. This is the metric used when there is no pre-existing class labels in the dataset. Accuracy and weighted F1-Score are represented by the Eqs. (14) and (17) respectively. Accuracy is a metric which considers True Positives and True Negatives from the predicted results whereas in F1-score, all aspects of confusion matrix are taken into account. Both the metrics range from 0 to 1—1 being the value for a good clustering quality.

$$Accuracy = \frac{No.\ of\ Correct\ Predictions}{Total\ No.\ of\ Predictions} \tag{14}$$

Let TP = True Positive, FP = False Positive and FN = False Negative

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

Table 2 External clustering metrics

Text embedding	Accuracy	F1	Rand score
Word2Vec	0.65	0.66	0.08
GloVe	0.57	0.59	0.00
Doc2Vec	0.89	0.89	0.59
BERT	0.98	0.98	0.92

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{17}$$

Rand Index is used to indicate how similar two cluster results are. It considers the pairs of samples and check if the pairs are assigned in same cluster or in different cluster in the ground truth and predicted labels. The value ranges from 0 to 1—1 being the value indicating a good clustering result. To explain Rand Index, let us assume the following—let n be the size of the dataset, let x and y be the two clustering solutions to be compared, let a be the If there are two clustering solutions x and y for a particular dataset of size n, Let a be the number of pairs of elements in the dataset that are in the same cluster of x and y, Let b be the number of pairs of elements in the dataset that are in the different clusters of x and y. Rand index and Adjusted Rand Index are represented by Eqs. (18) and (19) respectively.

$$RI = \frac{(a + b)}{\frac{(n(n-1)}{2}} \tag{18}$$

$$Adjusted - RI = \frac{RI - Expected\ RI}{max(RI) - Expected\ RI} \tag{19}$$

Silhouette score [21] is used to measure the goodness of the clustering solution and its value ranges from − 1 to + 1. + 1 indicates that the clusters are separated well apart and the points in each cluster are closely clustered. Silhouette score is represented in the Eq. (20).

$$SS = \frac{(u - v)}{max(u - v)} \tag{20}$$

where u is the average of the Euclidean distances between each data point and every other data point and v is the average of the distances of all clusters with every other cluster. This measure suffers from the curse of dimensionality. In this study, as we are dealing with textual data clustering and each document in the corpus is represented by a number of dimensions, we have reduce the number of dimensions of the vectors before using this metric. We have used Principal Component Analysis (PCA) for dimensionality reduction.

Table 3 Internal clustering metrics

Text embedding	Silhouette score	DB-Score
Word2Vec	0.38	1.01
GloVe	0.39	1.07
Doc2Vec	0.36	1.15
BERT	0.65	0.48

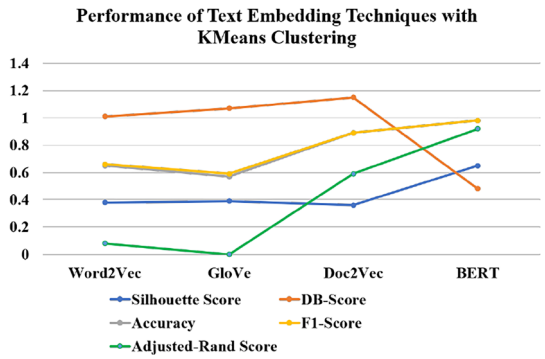

Fig. 2 Performance of text embedding techniques with K-means clustering

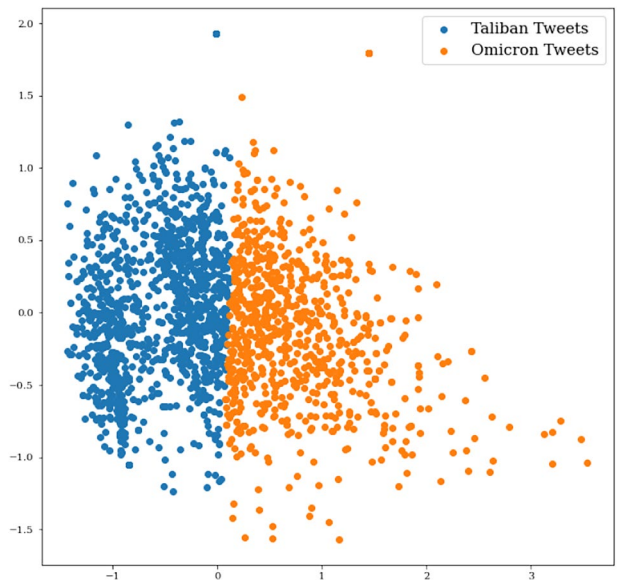

Fig. 3 Clusters with Word2Vec Embedding

Davies–Bouldin index [22] is also an internal evaluation metric for measuring the quality of clusters. Given a set of data points of n dimensions, let i be the number of clusters and C_i be the cluster of points. Let X_j be a vector of n dimensions which is assigned to a cluster C_i.

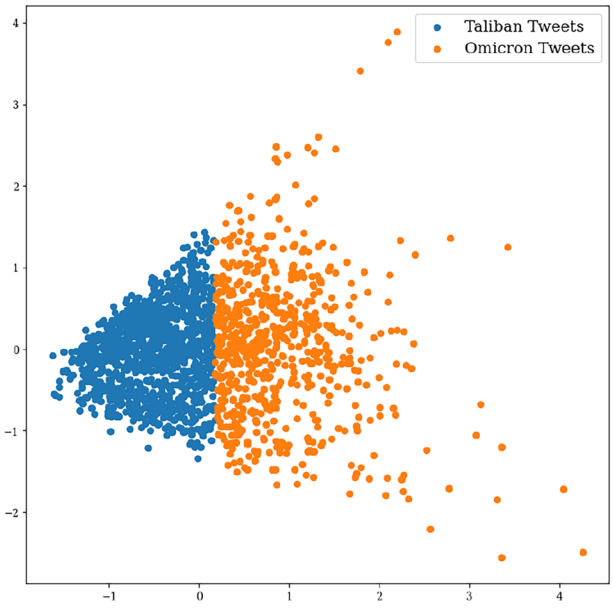

Fig. 4 Clusters with GloVe Embedding

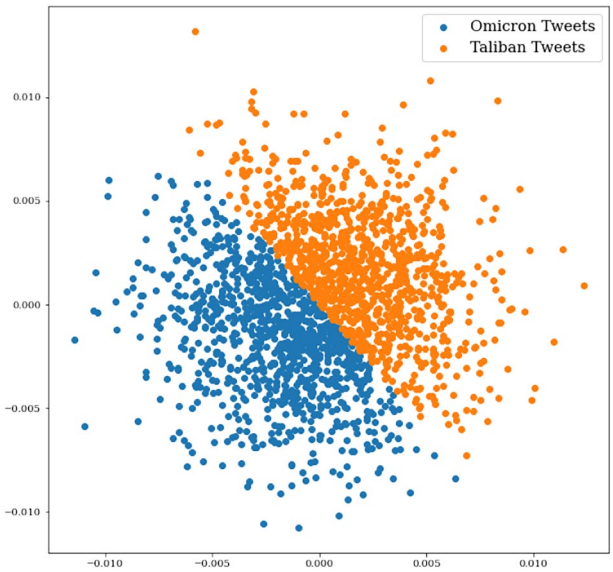

Fig. 5 Clusters with Doc2Vec Embedding

$$S_i = \left(\frac{1}{T_i} \sum_{1}^{T_i} \|x_j - a_i\|_p^q \right)^{\frac{1}{q}} \tag{21}$$

where A_i and T_i are the centroid and size of the cluster respectively. S_i indicates the qth root of the qth moment of the data points belonging to cluster. The value of p is set to 2 which sets the distance measure as Euclidean distance. The measure of separation M_{ij} of two clusters C_i and C_j is given by Eq. (22).

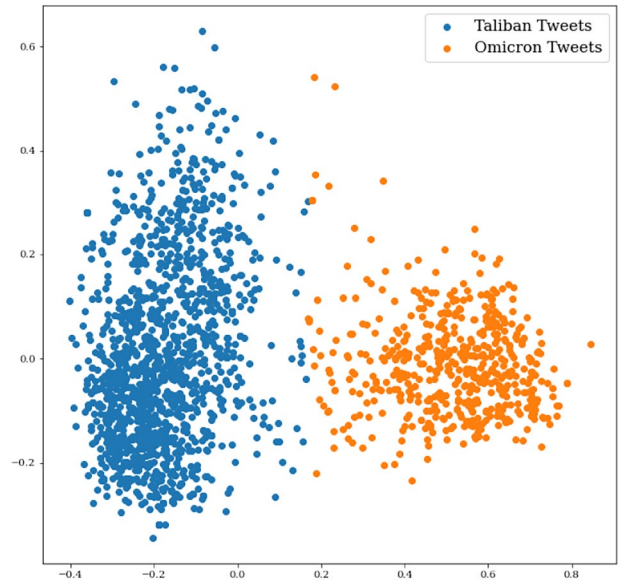

Fig. 6 Clusters with BERT Embedding

$$M_{ij} = \|A_i - A_j\|_p \tag{22}$$

where A_i and A_j denote two elements belonging to the clusters. Davies-Bouldin Index is represented by the Eq.(25).

$$R_{ij} = \frac{S_i + S_j}{M_{ij}} \tag{23}$$

$$D_i = max(R_{ij}) \| j \neq i \tag{24}$$

$$DB = \frac{1}{N} \sum_{1}^{N} D_i \tag{25}$$

where N is the number of clusters, R_{ij} is the measure to evaluate the goodness of clustering and DB is the Davies-Bouldin index. Lower the value of DB, better is the tightness of intra-cluster points and separation of clusters.

6 Results and analysis

We have collected tweets based on two core topics, we have kept the number of clusters in K-means algorithm as 2. Tables 2 and 3 list the performance scores of various text embedding methods adapted in this study.

The scores of various evaluation metrics are depicted graphically as shown in the Fig. 2. From the table and the figure, it is clearly evident that BERT models has outperformed Word2Vec, Doc2Vec and GloVe models in terms of all clustering evaluation metrics.

The Clusters formed using various techniques are shown in the Figs. 3, 4, 5, and 6.

The above clusters are generated using KMeans Clutering algorithm after applying text embedding techniques. From visual inspection of the above plots, we understand that text embedding techniques impact the clustering quality. For instance, the inter-cluster distance found in clusters generated after BERT is more clearly evident than other text-embedding techniques. One important aspect of BERT which makes it more effective than other embedding is that it is bi-directional—it takes both left word and right word of the target word. BERT takes into account both statistical information and contextual information of the corpus to create better embedding whereas other embedding techniques considers only the statistical information such as frequency of words occurring in the corpus, number of times words occurs in a sentence, etc. Embedding techniques like tf-idf and Word2Vec produce similar vectors irrespective of the meaning of the word in the context. For example, bank is treated in a similar way in both the sentences—"I went to the bank to deposit money" and "I went to the banks of Ganga". Even in GloVe method, there is only one vector representation for a word. This aspect makes GloVe also context independent. But in BERT, the context of the word is taken into account by taking the left word and right word of the target word and multiple representations of the target word is deduced based on the context.

7 Conclusion and future scope

Clustering Social media data has various application areas in effective information retrieval, event detection and topic modelling. Transforming textual information into vector is a crucial step in extracting the information. Using an appropriate text embedding technique on unstructured data like twitter data improves the quality of clustering. In this study, we have explored different document embedding techniques—Word2Vec, GloVe, Doc2Vec and BERT models on Tweets of news channels and we have found out that BERT model achieved 98% accuracy and gives the best performance of clustering result when the document vectors are clustered using K-means algorithm. We have used Principal Component Analysis for reducing the dimensionality of the dataset before applying the clustering algorithm. Evaluation of clustering result is done using both external and internal metric. Future work will be in the direction of applying textual clustering along with spatial clustering to find location-specific patterns of textual information.

Funding No funding was received for conducting this study.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alodadi M, Janeja VP (2015) Similarity in patient support forums using Tf-idf and cosine similarity metrics. In: 2015 International Conference on Healthcare Informatics, pp 521–522
2. Kim SW (2019) Research paper classification systems based on tf-idf and lda schemes. Human-centric Computing and Information Sciences. https://doi.org/10.1186/s13673-019-0192-7
3. Das B, Chakraborty S (2018) An improved text sentiment classification model using Tf-idf and next word negation
4. Bania RK (2020) Covid-19 public tweets sentiment analysis using Tf-idf and inductive learning models. INFOCOMP J Comput Sci 19(2):23–41
5. Li F, Liao L, Zhang L, Zhu X, Zhang B, Wang Z (2020) An efficient approach for measuring semantic similarity combining wordnet and wikipedia. IEEE Access 8:184318–184338. https://doi.org/10.1109/ACCESS.2020.3025611
6. Athota L, Shukla VK, Pandey N, Rana A (2020) Chatbot for healthcare system using artificial intelligence. In: 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp 619–622
7. Devkota B, Miyazaki H, Pahari N (2019) Utilizing user generated contents to describe tourism areas of interest. In: 2019 First International Conference on Smart Technology Urban Development (STUD), pp 1–6
8. Verma D, Muralikrishna SN (2020)Semantic similarity between short paragraphs using deep learning. In: 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp 1–5
9. Mathapati S, Anil D, Tanuja R, Manjula SH, Venugopal KR (2018) Cosint: mining reasons for sentiment variation on twitter using cosine similarity measurement. In: 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), pp 140–145
10. Vasanthakumar GU, Priyanka R, Vanitha Raj KC, Bhavani S, Rani BRA, Shenoy PD, Venugopal KR (2016) Ptmib: profiling top most influential blogger using content based data mining approach. In: 2016 International Conference on Data Science and Engineering (ICDSE), pp. 1–6
11. Paul S, Saha S (2020) Cyberbert: Bert for cyberbullying identification. Multimedia Systems. https://doi.org/10.1007/s00530-020-00710-4
12. Li M, Li W, Wang F, Jia X, Rui G (2020) Applying bert to analyze investor sentiment in stock market. Neural Computing and Applications. https://doi.org/10.1007/s00521-020-05411-7
13. Rosu R, Stoica A, Popescu P, Mihaescu C (2020) Nlp based deep learning approach for plagiarism detection. Int J User-System Interact 13, 48–60 https://doi.org/10.37789/ijusi.2020.13.1.4
14. Dessí D, Helaoui R, Kumar V, Recupero DR, Riboni D (2020) Tf-idf vs word embeddings for morbidity identification in clinical notes. An initial study. https://doi.org/10.5281/ZENODO.4777594
15. Subakti A, Murfi H, Hariadi N (2022) The performance of Bert as data representation of text clustering. J Big Data. https://doi.org/10.1186/s40537-022-00564-9
16. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space
17. Pennington J, Socher R, Manning C (2014) GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Doha, Qatar pp 1532–1543. https://doi.org/10.3115/v1/D14-1162. https://aclanthology.org/D14-1162
18. Le QV, Mikolov T (2014) Distributed representations of sentences and documents
19. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding
20. Reimers N, Gurevych I (2019) Sentence-BERT: sentence embeddings using Siamese Bert-networks
21. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65. https://doi.org/10.1016/0377-0427(87)90125-7
22. Davies, D.L., Bouldin, D.W (1979) A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1(2), 224–227 https://doi.org/10.1109/TPAMI.1979.4766909