Multimedia Tools and Applications (2023) 82:32967–32990
https://doi.org/10.1007/s11042-023-14653-1

Modified term frequency-inverse document frequency based deep hybrid framework for sentiment analysis

Ranit Kumar Dey[1] · **Asit Kumar Das**[1]

Received: 31 January 2022 / Revised: 12 April 2022 / Accepted: 3 February 2023 /
Published online: 4 March 2023

Abstract
Sentiment Analysis is a highly crucial subfield in Natural Language Processing that attempts to extract the public sentiment from the accessible user opinions. This paper proposes a hybridized neural network based sentiment analysis framework using a modified term frequency-inverse document frequency approach. After preprocessing of data, the basic term frequency-inverse document frequency scheme is improved by introducing a non-linear global weighting factor. This improved scheme is combined with the k-best selection method to vectorize textual features. Next, the pre-trained embedding technique is employed for the mathematical representation of the textual features to process them efficiently by the Deep Learning methodologies. The embedded features are then passed to the deep neural network, consisting of Convolutional Neural Network and Long Short Term Memory. Convolutional Neural Networks can build hierarchical representations for capturing locally embedded features within the feature space, and Long Short Term Memory tries to recall useful historical information for sentiment polarization. This deep neural network finally provides the sentiment label. The proposed model is compared with different state-of-the-art baseline models in terms of various performance metrics using several datasets to demonstrate its efficacy.

Keywords Natural language processing · Sentiment analysis · Term frequency-inverse document frequency · Deep learning · Convolutional neural network · Long short term memory

1 Introduction

In recent times, the workings of society have changed in a great manner due to the application of different technologies related to the internet [18]. People place orders on

✉ Asit Kumar Das
akdas@cs.iiests.ac.in

Ranit Kumar Dey
ranit.dey00@gmail.com

[1] Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah 711103, West Bengal, India

E-Commerce websites, communicate on social media, and use other web utility services. Different social networking platforms like Facebook and Twitter have grown up as a place for expansion of business, political campaigning, innovative idea sharing, promotion of different goods and services [67]. Users share their perspectives on these platforms and appraise the services and the products by giving star ratings [7] or thumbs up or down [17, 64]. Consequently, these platforms are significant resources of opinions given by the consumers. Regarding this context, Sentiment Analysis (SA) is an important research area focused on analyzing user opinions and partitioning them into mutually exclusive sets. SA is considered to be a subfield of Natural Language Processing (NLP), an extensive area in the domain of Artificial Intelligence [5], that works on the intercommunication between computers and natural human languages. Another popular name that is tagged with SA is opinion mining which helps in the extraction of emotion within the text [9, 33]. It classifies the text data into positive, neutral, or negative class [11] and also quantifies the polarity level [66]. SA can be employed on varieties of languages to capture the sentiment [7, 39].

During the previous decades scientists have explored the SA extensively [4, 48, 49]. The most popular approaches in SA contain lexicon-based, rule-based and Machine Learning (ML) based techniques [13, 37]. In the case of lexicon-based techniques, the sentiment polarity within the token of text is decided based on the semantic orientations of the text constituents [55]. Semantic orientation is the measurement of subjectivity and opinion within textual information. The rule-based approach [70] is based on the algorithms of association rule mining that discover rules with higher confidence and support which are capable of deciding the underlying sentiments of certain interesting features effectively. The rule-based technique finds the opinion words in the text and subsequently classifies the text depending on the number of negatively polarized and positively polarized words. It considers the various classification rules like polarity in the dictionary, boosting words, negation words, etc. ML employs different varieties of learning algorithms that use labeled data sets to train the classifier for determining the inherent sentiments [7, 57] which exist in the data set. Along with the mainstream techniques, the popularity of the hybrid methods is also growing in the sentiment classification.

Neural networks have gained so much popularity these days among different ML techniques in different applications including text mining [16, 32, 38], image processing [44, 54], video processing [34, 59] and audio [71] processing. Now-a-days, with the steady evolution of ML, more complex deep neural network models can be handled on large data sets [23, 31, 36]. Convolution Neural Networks (CNN) are becoming prominent Deep Learning (DL) techniques to achieve very accurate results in the domain of computer vision [31]. Within CNN, stacking of multiple convolution layers and pooling layers help in the task of sequential extraction of the hierarchical representation of the input data [29, 35]. Along with its presence in different domains, CNN is also showing improvement in SA [27, 28]. CNN can capture the non-linearity within the data and tries to learn the local embedding. Besides CNN, Recurrent Neural Network (RNN) [10] has recently become prominent that gives memory element for capturing dependencies in long-term sequence. In various language processing based tasks, RNN has shown very fruitful results [23, 61]. The main issues in the simple RNN are the vanishing gradient problem, where the value of the gradient tends to zero, and the exploding gradient problem, where the gradient tends to have a very high value. Long Short Term Memory (LSTM) is a variant of simple RNN that tries to mitigate the issues of vanishing gradient problem and exploding gradient problem [10, 24]. Term Frequency-Inverse Document Frequency (TF-IDF) [53] is a technique in text processing that is used for quantifying the tokens in the textual data. TF-IDF is a numerical statistical measurement that indicates how much value a constituent holds to the document. It is

prevalent for scoring the words in the text categorization algorithms [62]. During the weighing of tokens, this technique considers its importance in a single document and considers its relevance in the whole corpus. The weighing factor gives an edge to the TF-IDF text vectorization technique over the Bag of Words (BoW) scheme [73]. TF-IDF based approaches have shown promising results for different automated text analysis tasks like information retrieval [72], text classification [21, 62] etc.

Due to the growing importance of designing an integrated sentiment classification approach, we have presented a deep hybrid sentiment analysis framework by incorporating state-of-the-art deep learning techniques with a modified TF-IDF based text vectorization approach. Summarization of the proposed framework is described in the following sequence of steps:

(i) The data pass through the cleaning (garbage removal, slang correction, stopwords removal) and preprocessing (tokenization of text, parts of speech tagging) phase.
(ii) A modified TF-IDF based approach combined with the k-best selection method is devised for feature vectorization.
(iii) The pre-trained Word2Vec model on Google News corpus [22] is used for the embedding of the feature vector in 300 dimensional space. The pre-trained Word2Vec models are capable of giving high end embedding vectors [41, 52].
(iv) At the final stage, we appoint DL techniques by integrating CNN and LSTM. The DL techniques help extract the locally encoded dominant features, and the LSTM is used to apprehend the historical information.

We have shown the efficacy of the proposed framework compared to different baseline models with the help of various performance metrics like accuracy, precision, recall and F-Measure. The remaining portion of this paper is segmented in the following way. Section 2 describes the correlated research in the domain of Sentiment Analysis and the various techniques which are amalgamated within the hybrid approach. Section 3 discusses the proposed method and framework designing strategy in detail. Section 4 highlights the setup for the experiment and empirical observation analysis, and finally, Section 5 talks about the conclusion briefly and the future prospect of this paper.

2 Related work

Within this segment, we will talk over the related explorations that have been accomplished in the field of SA and the different methods we have utilized. A thorough overview of different algorithms, their amplification, and real-life applications in SA are demonstrated in [40]. It also discusses how to build a resource for sentiment analysis tasks. As a whole, it tries to give an overall scenario of SA and the associated fields with a brief introduction. One of the most popular techniques for weighing the constituents of textual data is TF-IDF. This is a statistical measurement based strategy for scoring the elements of the text. The mathematical foundation of the simple TF-IDF scheme is well explained in [53]. This paper has provided theoretical arguments and shown some problems with the information theory based approaches. It also provides theoretical justifications that the TF-IDF scheme is a comparatively better probabilistic model. The impact of the TF-IDF scheme, as a feature extraction scheme in the context of sentiment analysis, is discussed in [3]. By experimental results, it has shown that the TF-IDF scheme can give 3% to 4% higher performance than using the N-gram feature. A comparative study of the Bag of Words model and TF-IDF scheme is done in [15]. Experimental observations show that the TF-IDF scheme

performs better than the Bag of Words model of feature selection in the context of sentiment analysis. When they employed the Bag of Words model for feature selection in their movie review dataset, they achieved the accuracy of near about 86.6%. But when they employ TF-IDF scheme, the accuracy improves to 89.0%. Similar experiments have been conducted on movie review data in [21] where TF-IDF achieves better performance with an accuracy of 73.8% , but its counterpart achieves an accuracy of 66.5%. Now, the performance of simple TF-IDF scheme can further be improved if some weighting factors are imposed on top of the basic scheme. In [62] linear weighting factors are imposed on the basic TF-IDF scheme, and experimentally it has been shown that the weighted TF-IDF gives better results with improvement in accuracy in the range of 2.9% to 7.4%. Similar kinds of experiments have been carried out in [50]. Here the authors have considered data from different categories and, by experimental observations, they have shown that the TF-IDF scheme works better when the weighting factor is imposed on it. For example, in programmer category f-measure value increases to 0.79 from 0.74, in data configuration category f-measure value increases to 0.80 from 0.67, in operating system category f-measure value increases to 0.83 from 0.81 etc. As a result, to achieve better performance, we have modified the basic TF-IDF scheme by introducing a local weighting factor and a non-linear global weighting factor to accomplish the task of feature extraction.

For mathematical modeling, the textual data elements should be converted to floating values, and such a process is popularly termed Word to Vector embedding. Word2Vec is one of the widely used embedding approach [26] where each constituent of the text is converted to a floating-point vector. A comparative study of the bag of words embedding and Word2Vec embedding is presented in [20]. Here authors have considered cross-domain text classification results to show that Word2Vec embedding performs better than a bag of words embedding. For example, in the books domain, with a bag of words embedding, they achieve a classification accuracy of 66.90%, but with Word2Vec embedding, classification accuracy is improved to 68.74%. Similarly, in the music domain, the accuracy achieved with a bag of words embedding is 67.94%, but with Word2Vec embedding, accuracy improves to 68.53%. A comparative study between pre-trained embeddings, namely Glove embedding and pre-trained Word2Vec embedding, is depicted in [52]. The authors have shown that Word2Vec embedding works better than Glove embedding by experimental observations. When the authors use a static model, the Glove embedding gives the accuracy of 80.1%, 84.5%, 80.7% and 43.2% for MR, SST, TR, and SST-1 datasets, respectively. The Word2Vec embedding improves the accuracy to 80.6%, 84.9%, 81.0% and 46.1% respectively. When the authors have considered the non-static model, the Glove embedding shows the accuracy of 81.0%, 85.0%, 81.2% and 45.6% for respective datasets. The Word2Vec embedding gives improvement with accuracy of 81.2%, 85.3%, 81.5% and 46.1% respectively. Now how the performance varies with the underlying model of the embedding process is discussed in [41]. In this paper, by experimental results, the authors have shown that Skip-gram model based embedding works better than NNLM and CBOW model based embedding. The average accuracy that they have gotten for NNLM and CBOW based modelings are 50.8% and 63.7%, respectively. But for Skip-gram model based embedding, the average accuracy is 65.6%. A similar kind of study has been done in [42]. Here CBOW model based Word2Vec architecture achieves an accuracy of 0.772 and the Skip-gram model based Word2Vec architecture achieves an accuracy of 0.834. This study also has experimented on the effect of dimensionality of embedding. Experimentally it has shown that 100-dimensional and 200-dimensional embedding have the accuracy of 0.798 and 0.804 respectively. The 300-dimensional embedding outperforms them by giving the maximum accuracy of 0.808. Considering the previous research, for enhancement of performance,

we have considered the Skip-gram model based 300-dimensional pre-trained Word2Vec embedding, which is trained on Google News corpora.

Among different Machine Learning models, Neural Network is the one that is gaining popularity these days. It shows better performance in sentiment detection compared to other traditional approaches, which is experimentally observed in [11]. When authors employ a neural network for sentiment categorization in this study, the f-measure improves to 0.812 from 0.778 for the EC dataset. Similarly, for the MP3 dataset and Blog dataset, the f-measure improves to 0.735 from 0.683 and to 0.875 from 0.698, respectively. CNN is a kind of neural network that helps sequential extraction of features and hierarchical data processing. In the first stage, CNN showed great success in image processing [30]. Gradually, it has made its presence in language processing and text analysis. In [29] sentiment analysis has been performed on Twitter data using unigram and bigram features where SVM and MaxEnt have given the third-highest and second-highest performance with an accuracy of 81.6% and 83.0% respectively. The CNN outperforms them with an accuracy of 87.4%. Similarly, by experimental observation, the fruitfulness of CNN in sentiment analysis is shown in [28]. When authors have done sentiment analysis on the SED dataset, SVM and Logistic regression model have given f-score of 85.08% and 86.08% respectively. The CNN improves the performance with an f-score of 87.66%. For the SSTd dataset, the SVM and Logistic regression model have f-score of 77.90% and 76.18%, respectively. Here also, CNN gives the highest f-score of 80.72%. A similar scenario is observed in the STSGd dataset, where the SVM and logistic regression model have shown an f-score of 69.21% and 70.66%. Here CNN shows a remarkable improvement with an f-score of 82.65%. In [27], authors have collected reviews from different websites and created customized datasets. They have performed sentiment analysis on those customized datasets. In the case of the hotel dataset, the authors have found that Random Forest has given the third-highest performance with an accuracy of 83.0%, and SVM has shown the second highest accuracy of 92.3%. Here CNN gives the highest accuracy value of 94.3%. For their automobiles dataset, SVM and Random Forest have the accuracy of 84.8% and 85.6%. CNN outperforms them with an accuracy value of 93.4%. So previous studies have confirmed the success of CNN in a sentiment categorization task. RNN is a kind of neural network that can help in storing historical details and modeling the sequence data through its feedback connections. For simple vanilla RNN, the vanishing gradient (gradient becomes too small) and the exploding gradient (gradient becomes too high) problems complicate practical applications. LSTM is a variant of RNN that mitigates the obstacles of vanishing and exploding gradient. How the issues of vanishing and exploding gradient are resolved in LSTM are theoretically discussed in [24]. LSTM has successfully made its presence in sentiment analysis. Performance of LSTM in sentiment analysis is analyzed in [6]. Here authors have shown the superiority of LSTM in sentiment analysis over multi-layer perceptron and traditional machine learning approaches. They have performed sentiment analysis on a movie dataset where the logistic regression model, support vector machine, and multi-layer perceptron have achieved the accuracy of 85.50%, 82.89% and 87.70% respectively. The LSTM outperforms them with an accuracy of 88.46%. Similar kinds of experiments have been carried out in [74]. Here authors have performed sentiment analysis on micro-blogging data where Bayesian network, random forest, support vector machine achieve the f-measure of 63.97%, 67.18%, and 62.35% respectively. The LSTM improves the performance with an f-measure of 72.20%. In [63], covid-19 related tweets are analyzed using Naïve Bayes classifier, SVM, and LSTM. Here SVM and Naïve Bayes have shown the accuracy of 69% and 71%, respectively. The LSTM further improves the accuracy to 79%. So previous research has demonstrated a good foothold of LSTM in sentiment analysis tasks.

In this paper, different techniques are modified and combined for providing a better hybrid framework of sentiment polarity analysis. After preprocessing of data, the basic TF-IDF scheme is modified by introducing a local weighting factor and a non-linear global weighting factor, and k-best selection is made based on that modified scheme. Next, pre-trained Word2Vec embedding is employed. Finally, a deep neural network is designed by combining both the CNN and LSTM to take both advantages of sequential feature extraction and dependency preservation. The deep neural network architecture takes the embedded vectors as input to produce the desired sentiment label as output. It has been shown that the proposed hybrid framework performs better than the traditional machine learning approaches by testing the framework on various datasets by different performance metrics.

3 Proposed method and framework design

The broad view of the proposed framework is depicted in Fig. 1. From a broader perspective, the framework has four stages. The stages in sequence are Data Preprocessing, Text Vectorization, Embedding of Feature and ultimately, the application of deep network, which are described in detail in the following subsections. In Data Preprocessing phase, unnecessary details are removed, and the data is organized in a structured format. After that, the basic TF-IDF scheme is modified in the Text Vectorization phase by introducing a global weighting factor. This modified scheme is applied to data and the best features are selected from the data. After vectorization, textual features are converted into numerical structure by pre-trained Google News corpora-based embedding to be processed by neural networks more prominently. In the deep network phase, CNN and LSTM are combined to take advantage of both. CNN helps in the hierarchical extraction of predominant features in data, and LSTM assists in dependency preservation within the data. The embedded features are passed as input to the deep network, and it ultimately provides the sentiment label as the output.

3.1 Preprocessing of data

In the proposed framework preprocessing of data is the first footstep. Whenever data are collected, various kinds of noise, non-dictionary terms, expressions through emoji, acronyms, grammar irregularities, and poorly constructed sentences exist within the collected data. These types of deformities lead to the decaying of performance. Data preprocessing steps are essential to eliminate such irregularities. The whole procedure can be more efficient by representing the data in a structured shape. Data preprocessing consists of the following following sub-stages described below.

(i) **Garbage elimination:** The web links, URLs, and numeric values don't contain sentiment-related detailing. As a result, URLs, numeric values, non-ASCII characters and non-alphabetic phrases are eliminated from the data with the help of our designed custom regular expression

(ii) **Emoji substitution:** Emoji is basically used to express the writer's instant frame of mind with an icon. In this step, the emoji is substituted with its equivalent text by using the emoji package of python [19]. It assists in apprehending the sentiment information lying within the icon.

(iii) **Slang substitution:** The slang and abbreviations are expanded to their complete form to interpret their inner meaning. Suppose if we come across the word "ttyl", it

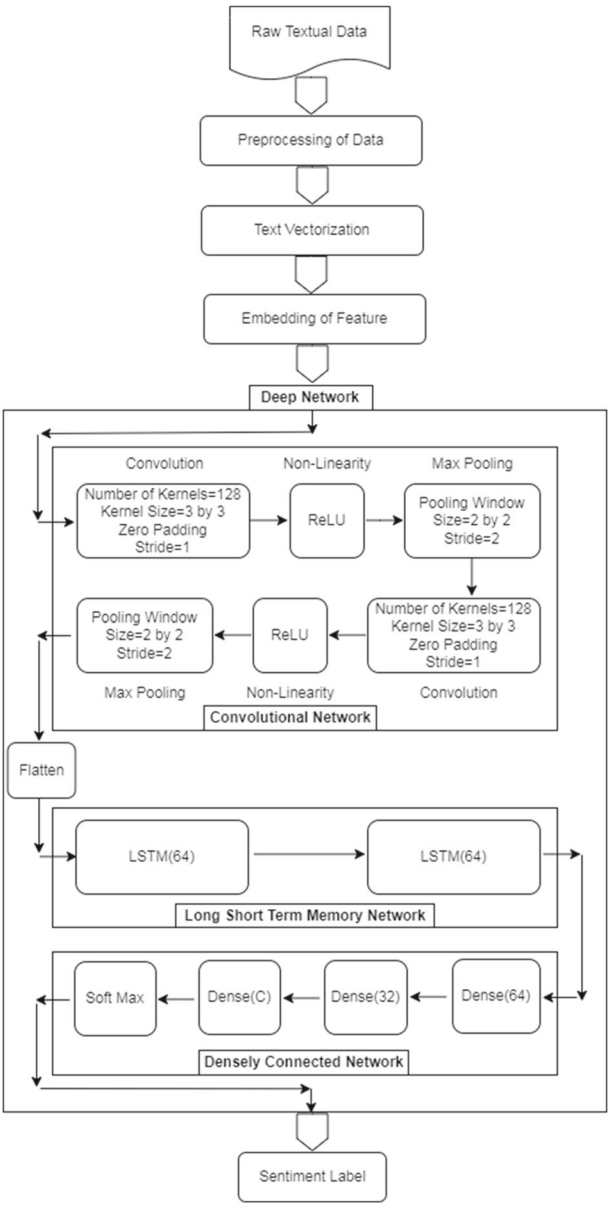

Fig. 1 Broad view of hybrid framework

is substituted by "talk to you later". We have created our custom dictionary by aggregating data from [25] and [14], and this dictionary is used for slang substitution. The primary working mechanism of replacing slang is described in Algorithm 1.

(iv) **Apostrophe reference substitution:** At this stage, the short apostrophe forms are expanded to their full expression to evaluate sentiment polarity better. Suppose if we come across "we've", it will be substituted by "we have" and the negative reference

such as "can't" will become "can not". Specifically, the negative references are vital in resolving the associated sentiment. For this task, we have designed a customized dictionary by taking data from [45]. The working mechanism behind this step is described in Algorithm 2.

(v) **Tokenization of text:** The words, phrases, and symbols are the meaning-bearing units of the text that are termed tokens. At this step of tokenization, the text is fragmented into tokens. Punkt Sentence Tokenizer [46] along with Penn Treebank Word Tokenizer [47] are imported from the python NLTK package for accomplishing this task. A customized data structure has been created to reserve the tokens for each sentence and sentence associated with each document.

(vi) **Stopword elimination:** Certain words such as "an", "the" etc. are quite common in discussion but do not hold too much significance while deciding the sentiment of the overall text. Such kinds of words are labeled as stopwords. To eliminate such kinds of stopwords, we import the NLTK package[43] of python. During the elimination of stopwords, the negation words like "not", "no" etc., are kept as they are valuable in sentiment polarity classification.

(vii) **Tagging parts of speech:** At this stage, each constituent within the text is tagged with associated part of speech (POS) depending upon its utilization within the text. To accomplish the task of POS tagging, POS tagger [8] has been employed from the NLTK package of python. The custom data structure has been maintained for the accumulation of the sentence tokens and the corresponding POS tags. After POS tagging, we only keep the adjectives, adverbs, nouns and adverbs as the sentiments can mainly be recognized from these POS.

Input: T_s(Text with slang)
Output: T_{ss}(Text after slang substitution)
begin
  /*lower case conversion of text*/
  $T_s \leftarrow T_s.lower()$
  /*tokenizing the text*/
  $F \leftarrow T_s.split()$
  /*slang loading from the dictionary*/
  $slang\_keys\_list \leftarrow slang\_dict\_load.keys()$
  /*initialization of final list*/
  $T_{ss} \leftarrow \phi$
  /*Updation of final list*/
  for *each* $f_i \in F$ **do**
    if $f_i \in slang\_keys\_list$ **then**
      /*Updation of token in the text*/
      $f_i \leftarrow slang\_dict\_load.value(f_i)$
    end
    $T_{ss} \leftarrow T_{ss}.append(f_i)$
  end
  return T_{ss}
end

Algorithm 1 SLANG_SUBSTITUTION.

Input: T_a(Text with apostrophe reference)
Output: T_{as}(Text after apostrophe reference substitution)
begin
  /*lower case conversion of text*/
  $T_a \leftarrow T_a.lower()$
  /*tokenizing the text*/
  $F \leftarrow T_a.split()$
  /*apostrophe references loading from the dictionary*/
  $apostrophe\_keys\_list \leftarrow apostrophe\_dict\_load.keys()$
  /*initialization of final list*/
  $T_{as} \leftarrow \phi$
  /*Updation of final list*/
  for *each* $f_i \in F$ **do**
    if $f_i \in apostrophe\_keys\_list$ **then**
      /*Updating the token within the text*/
      $f_i \leftarrow apostrophe\_dict\_load.value(f_i)$
    end
    $T_{as} \leftarrow T_{as}.append(f_i)$
  end
  return T_{as}
end

Algorithm 2 APOSTROPHE_SUBSTITUTION.

3.2 Text vectorization

To accomplish the task of sentiment analysis, the sentences in each review need to be represented as a feature vector. In NLP, one of the popular feature vector representation schemes is BoW [73] representation where the text is considered as a bag of words. In BoW, the ordering of the words and the grammar are disregarded. The basic principle of the TF-IDF technique is based upon the BoW scheme. In addition, the TF-IDF technique considers the weighing of the words in the text depending upon their occurrences. We have modified the basic TF-IDF scheme and then considered the k-best selection method. The first component of the TF-IDF scheme is TF, which is term frequency. Term frequency is the frequency of occurrence of a term in a document. Now the term frequency is proportional to document length, which means term frequency generally tends to be higher for a longer document. To mitigate the effect of document length, the term frequency of a term in a document is normalized by the sum of the term frequencies of all the terms in the document. This normalized term frequency is fundamentally the local weighing factor (LWF), as defined by the Eq. (1), where $lwf_{m,n}$ is the LWF of the m^{th} term of the vocabulary (t_m) corresponding to the n^{th} document (d_n) and $f_{m,n}$ is the frequency of occurrence of t_m in d_n. The second component of the TF-IDF scheme is IDF, which is inverse document frequency. The inverse document frequency indicates the importance of the term in the whole corpus, which is simply computed using subdivision of the total number of documents by the number of documents in which the term has occurred and taking the log function of it. The inverse document frequency is fundamentally the global weighing factor (GWF). In the modified new GWF (MNGWF), we have two modules, one is max GWF (MGWF) and another is smooth GWF (SGWF). The MGWF and the SGFW are specified in Eqs. (2) and (3) respectively, where $mgwf_m$ and $sgwf_m$ are the MGWF and SGWF of t_m, f_m indicates the number of documents in which t_m occurs and N is the total number of documents. The addition of MGWF and SGWF computes the MNGWF, and the final simplified form is given in Eq. (4). Finally,

the modified term frequency-inverse document frequency (MTFIDF) is given by the multiplication of LWF and MNGWF, which is shown in Eq. (5). In the case of the sentiment categorization task, each review is considered as a document. After preprocessing of data, MTFIDF is computed for each of the constituents of the individual review. Afterward, the K-best selection method is adopted for text vectorization. The whole mechanism is depicted in Algorithm (3).

$$lwf_{m,n} = \frac{f_{m,n}}{\sum_p f_{p,n}} \tag{1}$$

$$mgwf_m = \log(\frac{max\{m' \in d\} f_{m'}}{1 + f_m}) \tag{2}$$

$$sgwf_m = \log(\frac{N}{1 + f_m}) \tag{3}$$

$mngwf_m = mgwf_m + sgwf_m$
$= \log(\frac{max\{m' \in d\} f_{m'}}{1+f_m}) + \log(\frac{N}{1+f_m})$
$= \log(\frac{max\{m' \in d\} f_{m'} N}{(1+f_m)^2})$
$= \log(max\{m' \in d\} f_{m'}) + \log N - \log(1 + f_m)^2$
$= \log(max\{m' \in d\} f_{m'}) + \log N - 2\log(1 + f_m)$
$= \log(max\{m' \in d\} f_{m'}) - 2\log(1 + f_m)$
[Ignoring $\log N$ as for a given set of documents it is fixed]

$$mngwf_m = \log(max\{m' \in d\} f_{m'}) - 2\log(1 + f_m) \tag{4}$$

$$mtfidf_{m,n} = \frac{f_{m,n}}{\sum_p f_{p,n}} [\log(max\{m' \in d\} f_{m'}) - 2\log(1 + f_m)] \tag{5}$$

3.3 Embedding of feature

R_V that is computed by the Algorithm (3) is a set of feature vectors where each feature vector corresponds to an individual review, and each component of the feature vector represents a feature corresponding to that review. In this process of feature embedding, each component of the feature vector is converted to a floating vector so that further neural network based processing can be accomplished. Google's Word2Vec model is used [22] for the embedding of the feature, which is pre-trained on Google News dataset that contains nearly 100 billion words. The mechanism of the embedding process is depicted in Algorithm (4). Here, the vectorized review set R_V is passed as the input to the algorithm, and it produces embedded review set R_{eb} as output. Each component of individual review is embedded as a 300-dimensional vector that means R_{eb_p} is a matrix where $R_{eb_p} \in \mathbb{R}^{k \times 300}$. If a review vector is having n_c number of non-null components where $n_c < k$ then matrix of dimension $\mathbb{R}^{n_c \times 300}$ is concatenated with null matrix of dimension $\mathbb{R}^{(k-n_c) \times 300}$ to get the whole matrix of dimension $\mathbb{R}^{k \times 300}$. R_{eb} is the complete embedding set that contains each and every R_{eb_p}.

3.4 Deep network

The deep network stage consists of three phases, namely Convolutional Network, Long Short Term Memory Network, and Densely Connected Network, discussed subsequently. The architecture of the deep network is depicted in the box section of Fig. (1).

Input: R(Preprocessed and Cleaned Review Set)
Output: R_V(Vectorized Review Set)
begin
  /*initialization of final vector list*/
  $R_V \leftarrow \phi$
  /*updation of final vector list*/
  for *each* $r_i \in R$ **do**
    /*initialization of vector for an individual review*/
    $R_{V_i} \leftarrow \phi$
    /*initialization of MTFIDF list for an individual review*/
    $score_i \leftarrow \phi$
    /*computation of MTFIDF of each of the constituents of a review*/
    for *each* $r_{i,j} \in r_i$ **do**
      $score_{i,j} \leftarrow compute\_mtfidf(r_{i,j})$
    end
    $score_i \leftarrow score_i.sort()$
    /*updation of vector for an individual review*/
    $n_c = min(k, length(score_i)$
    for $iterator \leftarrow 1 to n_c$ **do**
      $R_{V_i}.append(score_{i,iterator})$
    end
    /*appending null for remaining $k - n_c$ components*/
    for $iterator \leftarrow 1 to (k - n_c)$ **do**
      $R_{V_i}.append(NULL)$
    end
    $R_V.append(R_{V_i})$
  end
  return R_V
end

Algorithm 3 MTIDF_K_BEST_SELECTION.

(i) **Convolutional network:** The Convolutional Network consists of a sequence of convolution operations, non-linear activation and max-pooling. The embedded review set R_{eb} is given as input to the convolutional network and it is passed through the series of sub-operations of the convolutional network. The convolution, non-linear activation and max-pooling operations are discussed in the following subsections. The block diagram of sequential processes in the convolutional network is depicted in the upper fragment of the deep network section in Fig. 1.

One of the most important components of a convolutional network is convolution operation. The convolution operation is helpful in the extraction of features from the local regions. Using a convolution kernel reduces the number of parameters that have to be learned in the network by sharing weights. The kernel slides over the input, and the portion of input within the window is point-wise multiplied with the $w_1 \times w_2$ kernel. The values are summed together to get a single representative value for the corresponding input window. Here, the convolution operation is performed with a stride value of 1, which indicates that the kernel should be shifted by one position over the input in each turn. For undefined values, zero padding is considered so that the convoluted output size becomes the same as the input size. The output feature map after convolution operation is expressed by the Eq. (6), where f is the output feature

Algorithm 4
REVIEW_EMBEDDING.

Input: R_V(Vectorized Review Set)
Output: R_{eb}(Embedded Review Set)
begin
  /*initialization of embedded vector list*/
  $R_{eb} \leftarrow \phi$
  /*updation of embedded vector list*/
  for *each* $r_p \in R_V$ **do**
    /*initialization of embedding for an individual review*/
    $R_{eb_p} \leftarrow \phi$
    /*updation of embedding for an individual review*/
    for *each* $r_{p,q} \in r_p$ **do**
      if $r_{p,q} \neq NULL$ **then**
        $R_{eb_p}.append(google_embedding[r_{p,q}])$
      end
      /*zero_vector(d) produces a d-dimensional zero vector*/
      else
        $R_{eb_p}.append(zero_vector(300))$
      end
    end
    $R_{eb}.append(R_{eb_p})$
  end
  return R_{eb}
end

map after convolution, I is the input and W is the kernel of size $w_1 \times w_2$.

$$f(x, y) = \sum_{p=0}^{w_1-1} \sum_{q=0}^{w_2-1} I(x - p, y - q)W(p, q) \tag{6}$$

The convoluted output is filtered through a non-linear function that gives the input as output if the input is greater than zero. Otherwise, it gives zero as output. This activation function is popularly known as Rectified Linear Unit (ReLU). This ReLU activation function reduces the vanishing gradient problem. The mathematical expression of ReLU is depicted in Eq. (7).

$$a(x) = max(0, x) \tag{7}$$

One of the popular sub-sampling methods in a convolutional network is max pooling which tries to select the most dominant feature from a region. It helps in dimensionality reduction while protecting the most valuable information. After max pooling, the number of parameters of the network gets reduced, which assist in mitigating the overfitting problem. In max pooling, a window of size $w_1 \times w_2$ slides over the input according to the stride size, and the maximum value of the input is selected from the corresponding window. The max-pooling operation is depicted in Eq. (8), where $m_w([x, x + w_1 - 1][y, y + w_2 - 1])$ is the output after max-pooling operation on the input window of position $x\ to\ (x+w_1-1)$ in vertical axis and $y\ to\ (y+w_2-1)$ in horizontal axis.

$$m_w([x, x\ +\ w_1 - 1][y, y + w_2 - 1]) = \max_{i \leftarrow 1\ to\ w_1} \max_{j \leftarrow 1\ to\ w_2} ([x, x + i - 1][y, y + j - 1]) \tag{8}$$

(ii) **Long short term memory network:** Vanilla RNN effectively preserves historical dependency, but vanishing and exploding gradients create problems. The Long Short Term Memory (LSTM) is an RNN variant that tries to mitigate these problems while memorizing the past dependencies. Convolutional Network output is flattened and then fed to the LSTM network as the input. A Block diagram of the LSTM network is shown in the middle fragment of the deep network section in Fig. 1 that has two layers, and each layer consists of 64 neurons. The LSTM computes the output $\hat{z_t}$ at any arbitrary time instant t with the help of current input to the network x_t and the previous internal state s_{t-1} at time instant $t-1$ which are shown in Eqs. (9) to (15). In those equations, $\odot$ indicates Hadamard product, $sigma()$ refers to sigmoid function and hyperbolic tangent function is indicated by $tanh()$ function. LSTM with a basic RNN structure contains different gating units for controlling the flow of information. First, there is the input gate i_t where we take the dot product of x_t that is the network input at time instant t and weight w_{x_i} and then the dot product of previous internal state s_{t-1} and weight w_{s_i} is taken. Finally, the sigmoid function is applied on the summation of them with the input gate bias b_i, as shown in Eq. (9). After the input gate, we have forget gate f_t that decides how much previous information needs to be ignored and how much historical information needs to be recalled for future processing. The expression of forget gate output is almost similar to that of input gate output; the only dissimilarity lies in the weighing factor. Here, x_t is multiplied by weight W_{x_f} and weighing factor for s_{t-1} is W_{s_f}. After the dot products, they are summed up with forget gate bias b_f, and the sigmoid function is applied to it to obtain forget gate output that is depicted in Eq. (10). Next, we have the modulation gate g_t. The mathematical expression of modulation gate output is also similar to that of input and forget gate; the distinction only lies in the weights, which is shown in Eq. (11). Then current memory content c_t at time instant t is computed with the help of memory content at the previous time instant, input gate, forget gate and modulation gate outputs using the Eq. (12). Subsequently, we have the output gate whose mathematical form is expressed in Eq. (13). Next, current internal state s_t is calculated with the help of output gate o_t and current memory content c_t using Eq. (14) and at last the output of the network is computed from the current internal state s_t following the Eq. (15).

$$i_t = \sigma(W_{x_i} x_t + W_{s_i} s_{t-1} + b_i) \tag{9}$$

$$f_t = \sigma(W_{x_f} x_t + W_{s_f} s_{t-1} + b_f) \tag{10}$$

$$g_t = \tanh(W_{x_g} x_t + W_{s_g} s_{t-1} + b_g) \tag{11}$$

$$c_t = g_t \odot i_t + c_{t-1} \odot f_t \tag{12}$$

$$g_t = \sigma(W_{x_o} x_t + W_{s_o} s_{t-1} + b_o) \tag{13}$$

$$s_t = o_t \odot \tanh(c_t) \tag{14}$$

$$z_t = \sigma(W_{s_z} s_t + b_z) \tag{15}$$

(iii) **Densely connected network:** The Densely Connected Network (DCN) is practically a reflection of the multilayer perceptron. The output of the LSTM network is passed to the DCN as input. The first layer of DCN has 64 neurons, the next layer consists of 32 neurons, and then there is a layer of c neurons where c is the number of output classes. After that softmax function makes a decision on the sentiment label. The

block of DCN is shown in the lower fragment of the deep network section in Fig. 1. The output of individual units within a layer is computed using Eq. (16). The softmax operation at the final layer is shown in Eq. (17) where O_q^l represents the output of the q^{th} node of the current layer and O^{l-1} is the output of the p^{th} node in the previous layer. W_{pq} is the weight of the link between the p^{th} node of the previous layer and the q^{th} node in the current layer. b_q^l indicates the bias associated with the q^{th} node of the current layer. O_i^f indicates the output of the i^{th} node of the final decision-making layer.

$$O_q^l = \sigma(\sum_p O_p^{l-1} W_{pq} + b_q^l) \tag{16}$$

$$softmax(O_i^f) = \frac{e^{O_i^f}}{\sum_j e^{O_j^f}} \tag{17}$$

4 Experimental observation and analysis

In the initial phase, we have discussed the experimental setup to perform the experiments. Next, we have introduced different datasets that have been utilized for comparative performance analysis. Then, the different baseline models that have been employed for comparative performance analysis are described. After that, various metrics of performance comparison that are measured for analysis of results with baseline models are discussed briefly. Finally, we have compared the observations shown by the proposed hybridized framework using different metrics of performance comparison with various baseline models.

4.1 Collection of dataset and setup of experiment

To compare the observations resulting from the proposed deep hybrid framework of sentiment analysis with the baselines, eight different datasets have been gathered from various sources. The various sources include Amazon alexa dataset (AN) [56], ETSY dataset (EY) [1], "Big Basket" app reviews dataset (BB) [65], Facebook dataset (FB) [69], Financial News dataset (FN) [58], Twitter dataset (TT) [2], Wine dataset (WN) [51]. The experiments have been carried out on an HP PC that has a core i7 Pentium processor, 32 GB of RAM and NVIDIA GPU (GTX Geforce 1080). Python has been primarily used for development with a main focus on two packages, namely NLTK for text processing and Keras and the Tensorflow in the backend for implementation of the deep network. The deep network has been trained using the backpropagation algorithm, and dropout technique [60] is employed with a drop out rate of 0.1. By the dropout technique, during training, randomly chosen neurons in the network are temporarily dropped following some probability where the activated values of corresponding neurons are not accumulated to the downstream neurons in case of the forward pass, and weight updations are not applied for those neurons in case of the backward pass. Dropout is one of the important regularization techniques that helps the network to prevent overfitting.

4.2 Baselines

To analyze performance, several baseline models of classification have been used that including IB-k, J48, JRip, NB, PART, RF, Logistic, SMO, CNN and LSTM, which are implemented using Weka [68] toolkit. A very brief introduction of the baseline models are given below.

(i) IB-k is an Instance-Based classification model based on k-nearest neighbors. When very minimal knowledge of the data distribution is available, then IB-k is an acceptable choice to perform classification.
(ii) J48 is a very popular decision tree-based classifier where entropy-based measure information gain is used to select attributes. The attribute giving the highest information gain is selected for further splitting.
(iii) JRip is based on learning the propositional rules, and the error here is brought down by incremental fashion pruning.
(iv) NB classifier works following the Bayes Theorem along with the assumption that all the features of the instances are independent in nature.
(v) PART follows the divide and conquer strategy for constructing a rule and further accomplishes the task of classification based on that rule base.
(vi) RF is an ensemble classifier where the decisions of a large number of decision trees are combined to give the final label.
(vii) The logistic classifier follows the logistic regression method and finally computes the posterior probability with the help of the sigmoid function.
(viii) SMO is the acronym for Sequential Minimal Optimization technique, which is used for finding solutions to quadratic programming problems that arise during the training of Support Vector Machines (SVM).
(ix) Convolutional Neural Network is popularly termed CNN. It is a neural network based classifier capable of hierarchical extraction of dominant features from the data.
(x) LSTM is the acronym for Long Short Term Memory. This neural network based model contains memory cells that help preserve historical dependency in data.

4.3 Metrics of performance

For analysis of performance, six performance metrics have been considered. The first performance metric is accuracy. First accuracy for each class is computed using Eq. (18) and then weighted average is calculated by Eq. (19) where ac_i is the set of instances that belong to class i and pc_i is the set of instances that are predicted as class i. Precision is the next performance metric. Like accuracy, first precision is computed for each class with the help of Eq. (20), and then the weighted average is taken using Eq. (21). Then recall for each class is measured using Eq. (22) and the overall recall is computed by weighted average with the help of Eq. (23). After that, f-measure for each class is the harmonic mean of precision and recall for that class which is depicted in Eq. (24), and the weighted mean is computed by the Eq. (25). The next performance metric is Area Under the Curve (AUC), where the area under the Receiver Operating Characteristics (ROC) curve is computed. For each class, the ROC curve is drawn by plotting the True Positive Rate (TPR) on the y-axis and False Positive Rate (FPR) on the x-axis and the area is calculated under that curve. After computation of AUC for each class weighted average is taken for the overall AUC. The Higher value

of AUC indicates a better model. After that, a statistical measure named Cohen's kappa co-efficient [12] is used for the evaluation of the model.

$$accuracy_i = \frac{|\{e : e \in ac_i \wedge e \in pc_i\}|}{\sum_{j=1}^{c} |ac_j|} \tag{18}$$

$$accuracy = \sum_{i=1}^{c} \frac{|ac_i| * accuracy_i}{\sum_{j=1}^{c} |ac_j|} \tag{19}$$

$$precision_i = \frac{|\{e : e \in ac_i \wedge e \in pc_i\}|}{|\{e : e \in ac_i \wedge e \in pc_i\}| + |\{e : e \notin ac_i \wedge e \in pc_i\}|} \tag{20}$$

$$precision = \sum_{i=1}^{c} \frac{|ac_i| * precision_i}{\sum_{j=1}^{c} |ac_j|} \tag{21}$$

$$recall_i = \frac{|\{e : e \in ac_i \wedge e \in pc_i\}|}{|\{e : e \in ac_i \wedge e \in pc_i\}| + |\{e : e \in ac_i \wedge e \notin pc_i\}|} \tag{22}$$

$$recall = \sum_{i=1}^{c} \frac{|ac_i| * recall_i}{\sum_{j=1}^{c} |ac_j|} \tag{23}$$

$$f - measure_i = \frac{2 * precision_i * recall_i}{precision_i + recall_i} \tag{24}$$

$$f - measure = \sum_{i=1}^{c} \frac{|ac_i| * f - measure_i}{\sum_{j=1}^{c} |ac_j|} \tag{25}$$

4.4 Performance analysis

The proposed hybrid framework is compared with the baseline models concerning previously depicted metrics of performance comparison.

(i) **Accuracy analysis:** Accuracy is a performance measure representing what percentage of test data is perfectly categorized. A comparison of the accuracy of various models on different datasets is expressed in Table 1. LSTM and RF show the 3^{rd} highest and 2^{nd} highest accuracy of 82.3% and 83.2% respectively for the Amazon dataset. The hybridized framework improves the accuracy to 84.4%. When the ETSY dataset is considered, CNN and LSTM have the accuracy of 74.2% and 74.7%. The hybridized method shows improvement with the accuracy percentage of 75.0%. In the case of the Big Basket dataset, CNN sets out the accuracy of 62.6%, and RF shows betterment with the accuracy of 62.8%. The hybridized approach delivers almost similar performance to RF with the accuracy of 62.9%. When the Facebook dataset is considered, RF and CNN give the accuracy of 57.8% and 59.3% respectively, whereas the hybridized model shows the highest accuracy of 62.1%. In the case of the Finance dataset, the 3^{rd} highest and 2^{nd} highest performance are given by the Logistic Model and LSTM with accuracy percentages of 78.6% and 79.4% respectively. The hybridized framework shows further improvement with an accuracy of 80.3%. For the Twitter dataset, CNN and LSTM have classification accuracy of 73.8% and 74.4%, respectively. The hybridized approach shows better accuracy of 76.9%. When

Table 1 Accuracy analysis (in %)

	AN	EY	BB	FB	FN	TT	WN
IB-k	81.1	69.2	55.5	41.3	66.8	52.7	82.9
J48	75.2	69.9	53.1	49.4	72.7	59.3	82.1
JRip	73.1	69.2	45.5	49.0	70.5	62.6	84.0
NB	70.5	68.1	49.2	48.1	67.8	61.8	79.2
PART	75.0	68.2	54.9	49.2	69.1	61.6	78.8
RF	83.2	69.9	62.8	57.8	72.9	69.3	84.9
Logistic	81.7	74.1	54.1	55.6	78.6	73.1	84.7
SMO	81.1	72.8	59.2	53.1	75.5	71.5	82.0
CNN	81.9	74.2	62.6	59.3	78.2	73.8	84.5
LSTM	82.3	74.7	62.2	57.5	79.4	74.4	86.8
Hybridized	**84.4**	**75.0**	**62.9**	**62.1**	**80.3**	**76.9**	**89.2**

the Wine dataset is considered, RF and LSTM have 84.9% and 86.8% accuracy. The hybridized framework further improves the performance with the accuracy of 89.2%. The proposed hybridized approach achieves the highest accuracy for all the datasets, as shown in Table 1 with boldface.

(ii) **Precision analysis:** Precision is a performance measure that indicates what proportion of a class identification is flawless. The comparison of precision values of various models on different datasets is shown in Table 2. For the Amazon dataset, CNN and LSTM have the same precision value of .810. Then RF and the hybridized approach achieve the same higher precision value of .835. In the case of the ETSY dataset, LSTM shows the 3^{rd} highest, and RF gives the highest performance with precision values of .712 and .722, respectively. Compared to RF, the proposed hybridized framework sets out slightly less precision of .715. When the Big Basket data set is considered, CNN and RF have the precision of .622 and .629, respectively. The hybrid model achieves further improvement with the precision of .637. For the Facebook dataset, CNN and SMO set out the 3^{rd} highest and 2^{nd} highest precision with values of .502 and .504, respectively. The hybridized approach shows betterment with the precision value of .508. While considering the Finance dataset, the Logistic Model and LSTM give the precision of .774 and .788, respectively. The proposed hybridized framework achieves further improvement with the precision value of .806. In the case of the Twitter dataset, the Logistic Model and LSTM have the precision of .734 and .788, respectively. Compared to them, the hybridized model shows good improvement with higher precision of .819. For the Wine dataset, LSTM gives the 3^{rd} highest, and RF shows the highest precision values of .808 and .850, respectively. The proposed hybridized method shows less precision than RF but slightly better performance than the LSTM with a precision value of .809. The highest precisions for different datasets are marked with bold faces in Table 2.

(iii) **Recall analysis:** The recall is one of the important performance measures that represents what proportion of the base truth of a class is identified flawlessly. A comparison of recall values of various models on different datasets is represented in Table 3. For the Amazon dataset, CNN and LSTM give recall values of .814 and .823, respectively. The hybridized framework shows improvement with a recall value of .841. In the case of the ETSY dataset, the Logistic model and LSTM have recall values of .741 and

Table 2 Precision analysis

	AN	EY	BB	FB	FN	TT	WN
IB-k	.785	.627	.560	.404	.651	.555	.790
J48	.712	.648	.530	.458	.720	.610	.762
JRip	.641	.642	.459	.399	.714	.670	.792
NB	.676	.640	.492	.462	.678	.656	.776
PART	.727	.657	.550	.468	.683	.619	.770
RF	**.835**	**.722**	.629	.468	.749	.701	**.850**
Logistic	.797	.696	.546	.478	.774	.734	.807
SMO	.797	.705	.594	.504	.754	.716	.791
CNN	.810	.708	.622	.502	.762	.732	.798
LSTM	.810	.712	.617	.498	.788	.788	.808
Hybridized	**.835**	.715	**.637**	**.508**	**.806**	**.819**	.809

.743, respectively. The proposed hybridized approach achieves a further higher recall value of .750. While considering the Big Basket dataset, CNN gives the 3^{rd} highest, and RF shows the highest performance with recall values of .598 and .626, respectively. The hybridized model achieves a slightly lower recall value of .624 than RF. For the Facebook dataset, RF and CNN give recall of .579 and .602, respectively. The proposed hybridized framework improves performance with a recall value of .620. In the case of the Finance dataset, the Logistic model gives the 3^{rd} highest performance, and the LSTM shows the highest performance with recall values of .786 and .791, respectively. The hybridized model achieves the highest recall value of .803. While the Twitter dataset is considered, the CNN and LSTM have recall of .737 and .744, respectively. The hybridized approach further improves the performance with a recall value of .769. For Wine dataset, the Logistic model shows the 3^{rd} highest recall value of .848 and LSTM gives the 2^{nd} highest recall value of .868. The proposed hybridized framework achieves the highest recall value of .892. The highest recall values for different datasets are marked with bold faces in Table 3.

Table 3 Recall analysis

	AN	EY	BB	FB	FN	TT	WN
IB-k	.806	.692	.555	.413	.669	.528	.830
J48	.751	.701	.530	.495	.727	.594	.819
JRip	.686	.698	.458	.490	.705	.627	.840
NB	.606	.623	.487	.481	.669	.632	.792
PART	.750	.682	.550	.493	.691	.617	.788
RF	.822	.698	**.626**	.579	.729	.693	.837
Logistic	.809	.741	.542	.556	.786	.731	.848
SMO	.811	.729	.593	.531	.756	.715	.823
CNN	.814	.738	.598	.602	.782	.737	.842
LSTM	.823	.743	.582	.573	.791	.744	.868
Hybridized	**.841**	**.750**	.624	**.620**	**.803**	**.769**	**.892**

Table 4 F-measure analysis

	AN	EY	BB	FB	FN	TT	WN
IB-k	.789	.631	.555	.406	.654	.533	.798
J48	.723	.664	.530	.473	.722	.596	.780
JRip	.646	.658	.458	.419	.702	.638	.807
NB	.635	.628	.488	.469	.678	.635	.778
PART	.737	.669	.550	.479	.686	.618	.778
RF	.825	.710	.627	.494	.735	.695	.841
Logistic	.799	.712	.543	.503	.778	.732	.812
SMO	.801	.715	.593	.515	.754	.715	.804
CNN	.811	.718	.614	.524	.768	.734	.814
LSTM	.819	.723	.602	.502	.789	.760	.821
Hybridized	**.836**	**.728**	**.629**	**.526**	**.804**	**.772**	**.842**

(iv) **F-measure analysis:** F-measure of a class is assessed by the harmonic mean of the precision and recall of the corresponding class. A comparison of the f-measure of various models on different datasets is depicted in the Table 4. For the Amazon dataset, LSTM and RF give an f-measure of .819 and .825, respectively. The hybridized framework shows better performance with the f-measure value of .836. In the case of the ETSY dataset, CNN and LSTM have f-measure values of .718 and .723, respectively. The proposed hybridized method achieves an f-measure value of .728. While considering the Big Basket dataset, CNN and RF give the 3^{rd} highest and 2^{nd} highest f-measure values of .614 and .627 respectively. The hybridized approach shows the highest performance with a slight improvement of f-measure to .629. For the Facebook dataset, SMO and CNN set out the f-measure of .515 and .524, respectively. The proposed hybridized framework further improves the performance with an f-measure value of .526. In the case of the Finance dataset, the Logistic model and LSTM have f-measure of .778 and .789, respectively. The hybridized model improves performance with an f-measure value of .804. While considering the Twitter dataset, CNN and LSTM give the f-measure values of .734 and .760, respectively. The hybridized method sets out a further higher f-measure value of .772. For Wine dataset, LSTM and RF show the 3^{rd} highest and 2^{nd} highest f-measure values of .821 and .841 respectively. The hybridized approach achieves greater performance with the f-measure value of .842. The proposed hybridized framework sets out the highest f-measure values for all the datasets, which are bold-faced in the Table 4.

(v) **AUC analysis:** The higher the AUC under the ROC curve better the model is. A comparison of AUC of various models on different datasets is depicted in Table 5. For the Amazon dataset, LSTM gives the 3^{rd} highest, and RF sets out the highest AUC values of .862 and .904, respectively. The hybridized model gives slightly less AUC of .899 compared to RF. In the case of the ETSY dataset, RF and Logistic model give 3^{rd} highest and highest AUC of .875 and .881, respectively. The hybridized approach shows slightly less performance than the Logistic model but very similar performance compared to RF with an AUC of .876. While considering the Big Basket dataset, CNN and RF have AUC of .849 and .857, respectively. The proposed hybridized framework sets out a further higher AUC of .861. For the Facebook dataset, RF gives the highest AUC of .771, and the Logistic model shows 2^{nd} highest AUC of .754.

Table 5 AUC analysis

	AN	EY	BB	FB	FN	TT	WN
IB-k	.742	.628	.754	.620	.667	.689	.578
J48	.723	.662	.774	.647	.735	.697	.614
JRip	.651	.621	.613	.575	.684	.737	.602
NB	.714	.725	.715	.718	.704	.729	.738
PART	.762	.728	.777	.653	.728	.714	.643
RF	**.904**	.875	.857	**.771**	.795	.856	.796
Logistic	.850	**.881**	.831	.754	.843	.888	.739
SMO	.791	.776	.828	.718	.812	.867	.722
CNN	.858	.868	.849	.730	.831	.891	.749
LSTM	.862	.872	.841	.722	.852	.897	.772
Hybridized	.899	.876	**.861**	.734	**.867**	**.901**	**.802**

The hybridized model sets out 3^{rd} highest AUC of .734. In the case of the Finance dataset, the Logistic model and LSTM have AUC of .843 and .852, respectively. The hybridized approach shows further betterment with an AUC of .867. While considering the Twitter dataset, CNN and LSTM show AUC of .891 and .897, respectively. The hybridized model gives further betterment with an AUC of .901. For the Wine dataset, LSTM and RF have the 3^{rd} highest and 2^{nd} highest AUC of .772 and .796 respectively. The proposed hybridized framework achieves the highest performance with an AUC of .802. The highest AUC values for different datasets are marked with bold-face in Table 5.

(vi) **Cohen's Kappa co-efficient analysis:** One of the popular statistical measurements used for assessing classification performance is Cohen's Kappa coefficient(ckce). This statistic is used to measure inter-rater reliability for categorical data. The higher the value of ckce, the better the model is. A comparison of ckce values of various models on different datasets is depicted in Table 6. For the Amazon dataset, CNN and LSTM show ckce values of .578 and .582. The hybridized framework improves the performance with ckce value of .686. In case of the ETSY dataset, the 3^{rd} highest ckce of .513 is given by CNN and the 2^{nd} highest ckce of .536 is shown by LSTM. The hybridized model achieves the highest ckce of .618. While considering the Big Basket dataset, CNN and RF set out the ckce of .532 and .534, respectively. The proposed hybridized approach shows betterment with ckce of .576. For the Facebook dataset, LSTM and CNN give ckce of .416 and .438, respectively. The hybridized framework further improves the performance with ckce of .526. In the case of the Finance dataset, the Logistic model and LSTM set out ckce of .587 and .602, respectively. The hybridized model shows a higher ckce of .637. While considering Twitter dataset, Logistic model and LSTM have 3^{rd} highest and 2^{nd} highest ckce of .589 and .598 respectively. The proposed hybridized approach achieves the highest performance with ckce of .625. For the Wine dataset, CNN and LSTM give the ckce of .268 and .277, respectively. The hybridized framework further improves the ckce to .293. For all the datasets, the proposed hybridized framework shows higher ckce values compared to the baseline models and the highest ckce values for all the datasets are marked with boldface in Table 6.

Table 6 Cohen's Kappa coefficient analysis

	AN	EY	BB	FB	FN	TT	WN
IB-k	.486	.272	.442	.165	.317	.298	.144
J48	.332	.364	.412	.263	.471	.384	.123
JRip	.310	.259	.353	.174	.384	.416	.112
NB	.309	.371	.383	.262	.401	.410	.215
PART	.386	.379	.436	.267	.421	.402	.201
RF	.495	.386	.534	.325	.492	.523	.227
Logistic	.517	.460	.424	.321	.587	.589	.225
SMO	.537	.467	.489	.317	.553	.564	.266
CNN	.578	.513	.532	.438	.581	.587	.268
LSTM	.582	.536	.514	.416	.602	.598	.277
Hybridized	**.686**	**.618**	**.576**	**.526**	**.637**	**.625**	**.293**

5 Conclusion and future prospect

Inside this paper, a modified TF-IDF based integrated deep hybrid framework has been designed, deployed and evaluated. It comprises modified TF-IDF based text vectorization, google news corpus-based pre-trained embedding followed by a deep network. When the analysis is done with respect to the accuracy, the proposed hybridized framework shows better performance than the baseline models for all datasets. When precision is considered as the performance metric, for ETSY and Wine datasets, the hybridized model shows 2^{nd} highest performance. For all other datasets, it sets out the highest precision value. While considering the Recall value, the hybridized framework gives 2^{nd} highest value in the case of the Big Basket dataset. For all other datasets, it achieves the highest recall value. In the case of the F-Measure as a performance metric, the hybridized approach produces the highest value for all the datasets. While considering the AUC, the hybridized model shows the 2^{nd} highest performance compared to baseline models in the case of Amazon and ETSY datasets and the 3^{rd} highest value for the Facebook dataset. For all other datasets, the hybridized framework shows better AUC values. From the statistical perspective, the proposed hybridized approach outperforms all the baselines for all the datasets in the case of Cohen's Kappa coefficient. So what has been observed is that the hybridized frame shows the best accuracy for all the datasets. Though it does not provide the best possible precision and recall for some portions, but for every dataset, it gives the best F-measure value which is harmonic combination of precision and recall. From the statistical viewpoint, the proposed approach also performs better than the rest of the baseline models for all the datasets. As a consequence, from the overall performance summary, it is concluded that the proposed hybrid deep framework adds value in the area of Sentiment Analysis.

As a future prospect, it can be investigated how the framework performance varies with the deepness of the network. In the case of a multiclass sentiment classification problem, there is a likelihood that the dataset may accommodate vague and uncertain data with overlapping classes, and there may be an imbalance among instances of different classes. Such uncertainty, vagueness, overlapping of classes and imbalance may be resolved with the help of various machine learning and soft computing techniques, which are the further scopes of the paper.

Acknowledgment In the accomplishment of this work, the University Grants Commission(UGC) of India is acknowledged for assisting by providing fellowship.

Declarations

Conflict of Interests The authors declare that the manuscript here has no conflict of interest concerning any other already published source and any resource that has not been previously published partly or in whole. No fabrication or manipulation of the data has been carried out to assist our conclusion.

References

1. Abdulelah Etsy reviews — kaggle. https://www.kaggle.com/csabdulelah/etsy-seller-reviews. Accessed 24 Nov 2021
2. Agrawal D Tweetsentimentanalysis/twitter.csv at master · dakshitagrawal/tweetsentimentanalysis · github. https://github.com/dakshitagrawal/TweetSentimentAnalysis/blob/master/Twitter.csv. Accessed 24 Nov 2021
3. Ahuja R, Chug A, Kohli S, Gupta S, Ahuja P (2019) The impact of features extraction on the sentiment analysis. Procedia Comput Sci 152:341–48
4. Ansari H, Vijayvergia A, Kumar K (2018) Dcr-hmm: Depression detection based on content rating using hidden markov model. In: 2018 Conference on Information and Communication Technology (CICT), IEEE, pp 1–6
5. Baclic O, Tunis M, Young K, Doan C, Swerdfeger H, Schonfeld J, Data P, Hub I (2020) Natural language processing (NLP) a subfield of artificial intelligence. CCDR 46(6):1–10
6. Bodapati JD, Veeranjaneyulu N, Shareef SN (2019) Sentiment analysis from movie reviews using LSTMs. Ingénierie des Systèmes d Inf 24(1):125–129
7. Boiy E, Moens MF (2009) A machine learning approach to sentiment analysis in multilingual web texts. Inf Retr 12(5):526–58
8. Categorizing and tagging words. http://www.nltk.org/book/ch05.html. Accessed 24 Nov 2021
9. Cambria E, Poria S, Gelbukh A, Thelwall M (2017) Sentiment analysis is a big suitcase. IEEE Intell Syst 32(6):74–80
10. Chen G (2016) A gentle tutorial of recurrent neural network with error backpropagation. arXiv:161002583
11. Chen LS, Liu CH, Chiu HJ (2011) A neural network based approach for sentiment classification in the blogosphere. J Informetrics 5(2):313–22
12. Cohen's kappa - wikipedia. https://en.wikipedia.org/wiki/Cohen_kappa. Accessed 24 Nov 2021
13. Collomb A, Costea C, Joyeux D, Hasan O, Brunie L (2014) A study and comparison of sentiment analysis methods for reputation evaluation. Rapport de recherche RR-LIRIS-2014-002
14. Complete list of text abbreviations & acronyms — webopedia. https://www.webopedia.com/reference/text-message-abbreviations/. Accessed 24 Nov 2021
15. Das B, Chakraborty S (2018) An improved text sentiment classification model using TF-IDF and next word negation. arXiv:180606407
16. Das P, Das AK, Nayak J, Pelusi D, Ding W (2021) Group incremental adaptive clustering based on neural network and rough set theory for crime report categorization. Neurocomputing 459:465–80
17. Deng ZH, Luo KH, Yu HL (2014) A study of supervised term weighting scheme for sentiment analysis. Expert Syst Appl 41(7):3506–13
18. DiMaggio P, Hargittai E, Neuman WR, Robinson JP (2001)
19. Emoji · pypi. https://pypi.org/project/emoji/. Accessed 24 Nov 2021
20. Enríquez F, Troyano JA, López-Solaz T (2016) An approach to the use of word embeddings in an opinion classification task. Expert Syst Appl 66:1–6
21. Ghag K, Shah K (2014) SentiTFIDF–sentiment classification using relative term frequency inverse document frequency. Int J Adv Comput Sci Appl 5(2). Citeseer
22. Github - mmihaltz/word2vec-googlenews-vectors: word2vec google news model. https://github.com/mmihaltz/word2vec-GoogleNews-vectors. Accessed 24 Nov 2021
23. Graves A, Mohamed Ar, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, pp 6645–6649
24. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–80
25. Internet slang dictionary & text slang translator. https://www.noslang.com/. Accessed 24 Nov 2021

26. Introduction to word embedding and word2vec — by dhruvil karani — towards data science. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. Accessed 24 Nov 2021
27. Ishaq A, Asghar S, Gillani SA (2020) Aspect-based sentiment analysis using a hybridized approach based on CNN and GA. IEEE Access 8:135499–512
28. Jianqiang Z, Xiaolin G, Xuejun Z (2018) Deep convolution neural networks for twitter sentiment analysis. IEEE Access 6:23253–60
29. Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. arXiv:14042188
30. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. AcM New York, NY, USA
31. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
32. Kumar K (2021) Text query based summarized event searching interface system using deep learning over cloud. Multimed Tools Appl 80(7):11079–94
33. Kumar K, Kurhekar M (2017) Sentimentalizer: Docker container utility over cloud 2017 Ninth international conference on advances in pattern recognition. IEEE, ICAPR, pp 1–6
34. Kumar K, Shrimankar DD (2017) F-des: Fast and deep event summarization. IEEE Trans Multimedia 20(2):323–34
35. Kumar K, Shrimankar DD (2018) Deep event learning boost-up approach: Delta. Multimed Tools Appl 77(20):26635–55
36. Kumar K, Kumar A, Bahuguna A (2017) D-CAD: Deep and crowded anomaly detection. In: Proceedings of the 7th international conference on computer and communication technology, pp 100–105
37. Kumar K, Bamrara R, Gupta P, Singh N (2020) M2P2: movie's trailer reviews based movie popularity prediction system. In: Soft computing: theories and applications, Springer, pp 671–681
38. Kumar S, Kumar K (2018) Irsc: integrated automated review mining system using virtual machines in cloud environment. In: 2018 Conference on Information and Communication Technology (CICT), IEEE, pp 1–6
39. MartíN-Valdivia MT, MartíNez-CáMara E, Perea-Ortega JM, UreñA-LóPez LA (2013) Sentiment polarity detection in spanish reviews combining supervised and unsupervised approaches. Expert Syst Appl 40(10):3934–42
40. Medhat W, Hassan A, Korashy H (2014) Sentiment analysis algorithms and applications: a survey. Ain Shams Eng J 5(4):1093–113
41. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv:13013781
42. Muhammad PF, Kusumaningrum R, Wibowo A (2021) Sentiment analysis using word2vec and long short-term memory (LSTM) for indonesian hotel reviews. Procedia Computer Science 179:728–35
43. Natural language toolkit — nltk 3.5 documentation. https://www.nltk.org/. Accessed 24 Nov 2021
44. Negi A, Kumar K, Chauhan P (2021) Deep neural network-based multi-class image classification for plant diseases. Agric Inform: Autom IoT Mach Learn 117–129. Wiley Online Library
45. NLP - replace apostrophe/short words in python - stack overflow. https://stackoverflow.com/questions/43018030/replace-apostrophe-short-words-in-python. Accessed 24 Nov 2021
46. nltk.tokenize.punkt — nltk 3.5 documentation. https://www.nltk.org/_modules/nltk/tokenize/punkt.html. Accessed 24 Nov 2021
47. nltk.tokenize.treebank — nltk 3.5 documentation. http://www.nltk.org/_modules/nltk/tokenize/treebank.html#TreebankWordTokenizer. Accessed 24 Nov 2021
48. Pang B, Lee L (2009) Opinion mining and sentiment analysis. Comput Linguist 35(2):311–2
49. Prabowo R, Thelwall M (2009) Sentiment analysis: a combined approach. J Informetrics 3(2):143–57
50. Qu S, Wang S, Zou Y (2008) Improvement of text feature selection method based on TFIDF. In: 2008 International Seminar on Future Information Technology and Management Engineering, IEEE, pp 79–81
51. Rai R Wine reviews — kaggle. https://www.kaggle.com/krrai77/wine-reviews. Accessed 24 Nov 2021
52. Rezaeinia SM, Rahmani R, Ghodsi A, Veisi H (2019) Sentiment analysis based on improved pre-trained word embeddings. Expert Syst Appl 117:139–47
53. Robertson S (2004) Understanding inverse document frequency: on theoretical arguments for IDF. J Doc. Emerald Group Publishing Limited
54. Sharma S, Kumar K, Singh N (2017a) D-fes: Deep facial expression recognition system. In: 2017 Conference on Information and Communication Technology (CICT), IEEE, pp 1–6
55. Sharma S, Kumar P, Kumar K (2017b) Lexer: Lexicon based emotion analyzer. In: International Conference on Pattern Recognition and Machine Intelligence, Springer, pp 373–379

56. Siddhartha M Amazon alexa reviews — kaggle. https://www.kaggle.com/sid321axn/amazon-alexa-reviews. Accessed 24 Nov 2021
57. Singh H, Dhanak N, Ansari H, Kumar K (2017) HDML: Habit detection with machine learning. In: Proceedings of the 7th International Conference on Computer and Communication Technology, pp 29–33
58. Sinha A Sentiment analysis for financial news — kaggle. https://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-news. Accessed 24 Nov 2021
59. Solanki A, Bamrara R, Kumar K, Singh N (2020) Vedl: a novel video event searching technique using deep learning. In: Soft Computing: Theories and Applications, Springer, pp 905–914
60. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–58
61. Sundermeyer M, Ney H, Schlüter R (2015) From feedforward to recurrent LSTM neural networks for language modeling. IEEE/ACM Trans Audio Speech Lang Process 23(3):517–29
62. Tokunaga T, Makoto I (1994) Text categorization based on weighted inverse document frequency. In: Special Interest Groups and Information Process Society of Japan SIG-IPSJ, Citeseer
63. Tripathi M (2021) Sentiment analysis of nepali covid19 tweets using nb SVM and LSTM. J Artif Intell 3(03):151–68
64. Turney PD (2002) Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. arXiv preprint cs/0212032
65. Varshney A "big basket" google play app reviews for basic NLP — Kaggle. https://www.kaggle.com/apurvavarshney/big-basket-google-play-app-reviews-for-basic-NLP. Accessed 24 Nov 2021
66. Vijayvergia A, Kumar K (2018) STAR: rating of reviews by exploiting variation in emotions using transfer learning framework. In: 2018 Conference on information and communication technology (CICT), IEEE, pp 1–6
67. Wang C, Zhang P (2012) The evolution of social commerce: the people, management, technology, and information dimensions. Commun Assoc Inf Syst 31(1):5
68. Weka 3 - data mining with open source machine learning software in java. https://www.cs.waikato.ac.nz/ml/weka/. Accessed 24 Nov 2021
69. Wolber L Facebook_reviews_trustpilot — kaggle. https://www.kaggle.com/leonwolber/facebook-reviews-trustpilot. Accessed 24 Nov 2021
70. Yang CS, Shih HP (2012) A rule-based approach for effective sentiment analysis
71. Yasmin G, Das AK, Nayak J, Vimal s, Dutta S (2022) A rough set theory and deep learning based predictive system for gender recognition using audio speech. Soft Computing, 1–24. Springer
72. Zhang H, Wang D, Wu W, Hu H (2012) Term frequency–function of document frequency: a new term weighting scheme for enterprise information retrieval. Enterp Inf Syst 6(4):433–44
73. Zhang Y, Jin R, Zhou ZH (2010) Understanding bag-of-words model: a statistical framework. Int J Mach Learn Cybern 1(1-4):43–52
74. Zhao J, Zeng D, Xiao Y, Che L, Wang M (2020) User personality prediction based on topic preference and sentiment analysis using LSTM model. Pattern Recogn Lett 138:397–402