Vol.65: e22210830, 2022
https://doi.org/10.1590/1678-4324-2022210830
ISSN 1678-4324 Online Edition

Article - Engineering, Technology and Techniques

An Experimental Analysis of Optimal Hybrid Word Embedding Methods for Text Classification Using a Movie Review Dataset

Sandhya Alagarsamy[1]
https://orcid.org/0000-0001-9331-6597

Visumathi James[2]
http://orcid.org/0000-0002-6152-8189

Raja Soosaimarian Peter Raj[3]
https://orcid.org/0000-0002-7216-2207

[1]Sathyabama Institute of Science and Technology, Department of Computer Science and Engineering, Chennai, Tamil Nadu, India; [2]Veltech Rangarajan Dr Sagunthala R&D Institute of Science and Technology, Department of Computer Science and Engineering, Chennai, Tamil Nadu, India; [3]Vellore Institute of Technology, School of Computer Science and Engineering, Vellore, Tamil Nadu, India.

Editor-in-Chief: Alexandre Rasi Aoki
Associate Editor: Fabio Alessandro Guerra

Received: 22-Dec-2021; Accepted: 18-Feb-2022.

*Correspondence: sandhyalagar@gmail.com (S.A.).

HIGHLIGHTS

- Proposed Hybrid Word Embedding (HWE) models for Efficient Text classification.
- Data Sparsity issue is reduced using WordNet repository along with proposed model.
- Optimal model is derived based on the Performance evaluation on the model.

Abstract: Today, a wealth of data is being produced over the internet from multiple sources, giving rise to the term *big data.* Much big data is contributed largely in the form of text. This work focuses on text classification of movie reviews dataset using Hybrid Word Embedding (HWE) models and deriving the optimal text classification model. However, in text processing, efficient handling and processing of the words and sentences in a document plays a vital role. In traditional methods like Bag of words (BoW) semantic correlation among the words does not exist. Further, the words in a document are not always processed in order, which results in certain words not being processed at all and creating problems with data sparsity. To overcome the data sparsity problem, the proposed work applied hybrid word embedding using WordNet repository. The hybrid model is built with three word embedding methods, namely, an embedding layer, Word2Vec and GloVe, in combination with the deep learning Convolutional Neural Network (CNN). The results obtained for the movie review dataset set was compared and the optimal classification model is identified. Various metrics considered for evaluation includes Log loss, Area under Curve (AUC), Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG), Mean Absolute Error (MAE), Error Rate (ERR), Mathews Correlation Coefficient (MCC), Training Accuracy, Test Accuracy, Precision, Recall and F1 score. Finally, the experimental results proved that the word2vec is derived as the optimal hybrid word embedding model for classification of chosen movie review dataset.

Keywords: HybridWord Embedding; Natural Language Processing; Deep Neural Network; Text Classification; CNN.

INTRODUCTION

Text classification, which is a crucial element in NLP applications, handles word sequences. It assigns predefined single or multiple labels to a text sequence. Each sentence in a document is represented as individual words so as to ascertain their similarity. The Three parts of text classification are feature vector representation, feature extraction and classification algorithm. Text classification is undertaken using techniques like knowledge engineering, expert systems and machine learning models such as the Support Vector Machine (SVM), K-Nearest Neighbour (KNN) and Maximum Entropy (ME) [9, 11].

Various other traditional methods like unigrams, bigrams, Bag of words and the Term Frequency-Inverse Document Frequency (TF-IDF) is used to find the probability among the words. Building a model with traditional method is simple and it makes the process of training small datasets easy. In traditional methods, however, the semantic correlation between words is not handled. Also, the fact that words are not processed in order culminates in data sparsity issues.

Data grows rapidly with inflows from social networks over the internet, compounding the data sparsity problem. The traditional models show poor classification results for larger datasets. Text representation using machine learning faces data sparsity issues as well, resulting in weak feature extraction that requires manual feature engineering for large datasets. Therefore, to solve the data sparsity issue, Hybrid word embedding model is built using wordnet. Stemming and Linking are two main processes done using wordnet. Words with a similar meaning, irrespective of grammar, are grouped under a single node using stemming, which a tree-like structure is made up of similar words. The linking process mapshypernyms for the words in the stemming tree. Since, all the words are processed in an order using stemming, the data sparsity issue vanishes and efficient embedding model is built.

Multilayer neural networks, which transform low-level features into deeper and more advanced features, came into being to handle large datasets. Deep neural networks have magnificent learning capabilities thereby helps to achieve outstanding results in nature language processing. They extract relevant features without complex artificial feature engineering techniques, thus bypassing the problem of data sparseness. The neural network models like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) work better than the traditional machine learning models and overcome the issue of data sparsity.

In deep neural network models, the words are converted into vector sequence of fixed length before the model is trained. In the vector space [4,5], words with similar meanings i.e) semantically similar words are represented closer to each other. The embedding layer, Word2Vec and GloVe are among the better text classification approaches. In this paper, the experimental results obtained for a movie review dataset using the three approaches are compared using several performance metrics.

While handling the classification of text data, the traditional models can process the smaller datasets in an efficient way without discarding any words. Also the semantic relation and correlation is not calculated in traditional models. When the data grows abundantly, the traditional models cannot handle the text classification. So Deep Neural Networks, with its architecture can process millions of data at faster rate without discarding any of the data. The hybrid model achieves semantic mapping for all text in an order and does efficient classification. Hence DNN performs better than traditional models.

Hence the objective of this research work is to propose a Hybrid Word Embedding (HWE) models for Efficient Text classification. Data sparsity is reduced using the WordNet repository with the proposed model, and an optimal model is derived, based on a performance evaluation.

Related work

Weston & Collobert [18] suggested convolution neural network architectures for natural language-processing problems. Image processing-related work initially used the same approach. Pennington and coauthors [15] recommends a word embedding method for word representation called Global Vectors (Glove). Word2Vec [10, 13] is another popular word embedding method used in neural network based natural language processing. All these approaches did not implement the hybrid approach.

Kim (2014) proposed [9] a deep neural network method for feature extraction and document classification that performs remarkably well in NLP tasks. Bozyigit and coauthors [3] collected data from multiple news sites and created a dataset comprising a few categories. Miscellaneous machine learning algorithms were applied to the dataset to carry out text classification. Ming and Xianchun [13] proposed the Doc2Vec model

that combines the Word2Vec and a clustering algorithm to extract information from documents. The TF-IDF algorithm is used alongside theWord2Vec to create document vectors. Andrei and Radu (2017) proposed a new text classification approach using clustering-based word embeddings and the k-means. This work outperforms the bag-of-words approach.

Hughes and coauthors [12] proposed an approach for sentence- level classification using a multilayer deep convolutional neural network that generates optimal features to represent word semantics. Kilimci et a [l8] suggested different word embeddings and ensemble learning for classifiers in text classification. The use of heterogeneous ensembles with word embeddings and deep learning enhances the text classification. Roger and coauthors [17] proposed the word embedding models along with machine learning models for hierarchical text classification. Word2vec, Glove and fast Text proved to be best classification models. Yao and coauthors [19] proposed Graph convolutional neural network for text classification. Single text graph is built for word corpus and then Text Graph convolutional network built for the corpus yields better results.

Albalawi and coauthors [1] implemented deep learning models like BiLSTM with word embeddings and compared with traditional machine learning models for health related tweets from social media. The deep learning model produces greater classification accuracy than ML models. Guilherme and coauthors [6] proposed a distance-based vector embedding technique based on Logistic Markov Embedding (LME). The scalability issue is addressed using the proposed model with a negative sampling approach. Moreo and coauthors [14] proposed word class embedding methods merged with pre-trained word embeddings for solving NLP tasks. The proposed work enhances the deep learning training and multiclass classification. C28 Pittaras and coauthors [16] suggested extracting the semantics of each word and applying aWord2Vec embedding model thereafter. Applying semantics yields better text classification performance. The summary of related works are listed in Table 1.

Table 1. Summary of Related Works

Reference	Proposed Methodology	Finding	Limitation
Weston and Collobert (2008)	Convolution neural network architectures were proposed for performing NLP tasks	Text classification process can be performed using CNN.	Word embedding techniques were not used.
Kim (2014)	Suggested to implement feature extraction techniques along with deep neural network methods.	Simple CNN and word2vec with slight modification in the hyper parameters, yields optimal classification of NLP tasks.	Single layer CNN is used. This will not perform efficiently for huge datasets.
Le Quoc and T. Mikolov (2014)	Proposed an unsupervised algorithm named paragraph vector to overcome the drawbacks of bag-of-words.	Paragraph vector works efficiently than bag of words in text classification	Implementation of paragraph vector is expensive
Pennington et al. (2014)	Proposed a global regression model that combines two models namely global matrix factorization and local context window method.	The vector space produced by the model is with meaningful sub structures.	The models performance varies based on the number of negative samples.
Bozyigit et al. (2015)	Five classifiers and two feature selection methods in the Text classifications were evaluated on news dataset.	Best classification accuracy is obtained using this combination on the dataset.	The classification accuracy is not achieved for large datasets.
Ming and Xianchun (2016)	Proposed the doc2vec model which is the combination of word2vec and clustering algorithm to express the information of document.	TF-IDF algorithm is used along with word2vec to form document vectors.	Single layer CNN is used which holds good for small documents. Multiple layers is missing for handling large documents.

Cont. Table 1

Reference	Proposed Methodology	Finding	Limitation
Andrei and Radu (2017)	The new approach proposed for text classification is clustering based word embeddings using k-means.	The proposed work provides better results than bag of words approach.	Alternate to k-means algorithm can be implemented to yield better results.
Hughes et al. (2017)	The proposed approach is to perform classification at sentence level using deep convolutional neural network.	Multilayer deep convolutional neural network generates optimal features to represent semantics.	This approach has scalability issue. It works only for small datasets.
Kilimci et al (2018)	Different word embeddings and ensemble learning for classifiers is proposed for text classification.	The use of heterogeneous ensembles with word embeddings and deep learning enhances the text classification.	Selecting the appropriate ensemble technique to yield optimal accuracy is challenging.
Roger et al (2019)	Word embedding models along with machine learning models is proposed for hierarchical text classification.	Word2vec, Glove and fastText proved to be best classification models.	Hierarchical text classification becomes complex while handling the real time continuous data.
Yao et al (2019)	Graph convolutional neural network is proposed for text classification.	Single text graph is built for word corpus and then Text Graph convolutional network built for the corpus yields better results.	This approach lowers the training percentage in the dataset.
Albalawi et al. (2021)	Deep learning models like BiLSTM with word embeddings are compared with traditional machine learning models for health related tweets from social media.	The classification accuracy is more with deep learning model when compared with ML models.	The use of advanced Deep learning techniques like auto encoders may impact better than the used approaches.
Guilherme et al. (2021)	An embedding technique (Distance based vector Embedding) based on Logistic Markov Embedding is proposed.	Scalability issue is addressed using the proposed model along with negative sampling approach.	The work limits with machine learning approaches. Deep learning techniques were not implemented.
Moreo et al. (2021)	Proposed word class embedding methods were merged with pre-trained word embeddings for solving NLP tasks.	The proposed work enhances the deep learning training and multiclass classification.	This approach is not suitable for binary classification.
Pittaras et al. (2021)	Semantics were extracted for each word and then word2vec embedding model is applied.	Applying semantics yields better performance on text classification.	The classification models computation complexity is increased.

Contributions

This research proposes the implementation of a hybrid word embedding model using WordNet and compares several word embedding models used for text classification so as to derive an optimal model. Though traditional methods execute text classification, the three different hybrid word embedding models implemented in this work outperform them by doing away with the data sparsity issue and yielding optimal results. The models are evaluated, based on specific metrics, and the best one is derived. The entire implementation is done for the movie review dataset obtained from IMDB repository.

Outline of the proposed work

The proposed work focuses on efficient text classification using hybrid word embedding models. The input considered for classification is standard movie review dataset. To begin with, the dataset is preprocessed to remove commas, punctuation and stop words, following which hybrid word embedding is applied to the cleaned dataset. Stemming and linking are two hybrid word embedding processes. Similar and

related words from the input document are grouped under same structure by referring the wordnet. Grammar is not considered in the grouping process. Also, since words are processed in order, the data sparsity issue is overcome. Linking maps the hypernym relation from WordNet for all the words in the stemming tree-like structure created. Now all the words in the tree structure were processed by various word embedding methods and the model is built. Then these models were compared and the best model is given as output. The overview of the proposed work flow is given in Figure. 1.

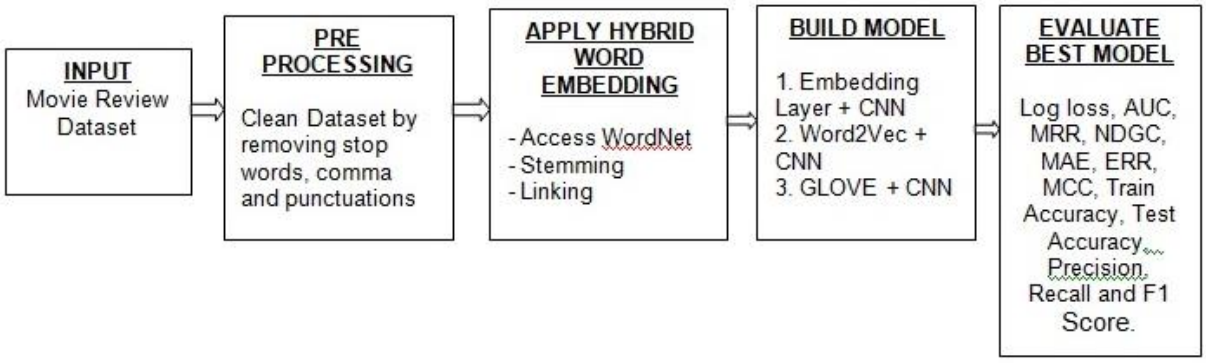

Figure 1. Overview of Proposed Workflow.

Methodology

The proposed work considers a movie review dataset obtained from the standard IMDB database for positive and negative text classification. The dataset is first cleaned and tokens are obtained. Then the dataset is split into train and test data. The testing vocabulary is built before processing the model. Now word tree is created using stemming and linking process. Then appropriate word embedding model is applied and passed to the CNN. Inspite of various neural networks that are available, convolution neural network handles the text data in an efficient way. To attain the most accurate text classification output, this research work implemented CNN along with various embedding algorithms. All the outputs received from the model are now concatenated to obtain the final classification results. Several word processing algorithms and methods are used in this work to create an embedding layer, and a deep learning model is built for prediction.

Section 2.1 explains the nature of the dataset used and how it is pre-processed before it is split into training and testing datasets for further predictions. Section 2.2 describes the three different word embedding algorithms used in this work and how the deep learning model is built using the CNN algorithm for predictions. The three models are compared using a slew of metrics to obtain the best model, which is used for further word processing applications with large datasets to obtain optimal predictions faster.

Proposed Models-

This section discusses the dataset used for the proposed hybrid model as well as the implementation of the three word embedding algorithms with the CNN.

Datasets

The Movie Review Dataset used in this work for evaluation of our model, is a collection of movie reviews fetched from the standard IMDB database (https://reviews.imdb.com/Reviews/review_polarity.tar.gz). Version 2 of the dataset used here is an updated and cleaned version, referred to as *v2.0*.

The movie review dataset consists of 1,000 positive and 1,000 negative movie reviews. This corpus dataset is known as polarity dataset. The dataset has positive and negative labels. The entire dataset is split into training (90%) and test data (10%). The last 100 positive reviews and the last 100 negative reviews are considered for test set (200 reviews) and the remaining 1,800 reviews for training dataset. Before the model is evaluated, the dataset is pre-processed. Commas, punctuation and stop words are removed. alongside words with a character length less than or equal to one. A list of cleaned tokens is obtained after pre-processing. Stemming and linking are then applied to form the word tree. The cleaned tokens that constitute the vocabulary are constructed, based on the word tree. Next, the word embedding methods are applied and the neural network model trained for the movie review dataset. The results obtained from the embedding methods are compared and evaluated.

Hybrid Word Embedding

The paper implements Hybrid word embedding. The corpus of semantically related words for any given word is available in wordnet. Various application programming interfaces helps to access the corpus. Wordnet lexical database is a part of NLTK corpus and the synonymous nodes are accessed using wordnet API. The wordnet processes and converts the pre-processed text data into vector representation. The obtained vector is fused with the output obtained from word embedding. The fused representation is passed to neural network and text classification is done.

Stemming and linking are indispensable to hybrid word embedding. Similar words appear under same root using the morphology function available in WordNet. For example, “eats”, “eat” and “eating” are all mapped under the root word “eat”. Here we do not consider the grammar. so, the word “eaten” is also mapped to root “eat”. The linking process identifies hypernyms in a process that maps terms with general and specific terms. The hypernym relationship is exploited from WordNet to build the word trees. For example, an orange is a hyponym of edible fruit and an edible fruit is a hypernym of an orange. To retrieve most semantically matched words during linking, hypernym function is used. In hypernym function the related words are retrieved based on the type of relation associated with the root word. In this work, the parameter that controls the hypernym mapping is the “level”. This work uses three levels of hypernym retrieval. In the hybrid word embedding phase, the end result obtained from hypernym is in the form of vector representation. The vector representation is based on semantic correlation among the words. Words in different context represent different concepts, which are represented as multiple word trees. This work implemented disambiguation property. The wordnet API retrieves all synchronous set of words from the corpus. To attain the most accurate match, not all the synsets are processed to next level. To perform the most efficient retrieval, parts of speech (POS) disambiguation property is implemented. Under POS disambiguation, the retrieval is restricted and more related semantic synsets are obtained.

The Fusion methodology is used to attain the hybrid word embedding. Using fusion methodology, the pre-processed text is passed in two ways. One to the wordnet and other to the embedding algorithm. The values obtained from both the ways are fused and passed to the neural network phase. This fusion methodology helps to attain more accurate classification. Thus, the tree representation completes the processing of all the words, leaving no room for the sparsity issue. Now various embedding algorithms are applied and results are compared.

Word Embedding Algorithms

The final output obtained from wordnet after disambiguation property and three levels of hypernomy are represented in vector form using semantic distance calculation algorithm. The word embedding algorithm also maps each word into vector representation. So the results from wordnet and word embedding are Fused and synchronized. Sigmoid activation function is used in the neural network. Since the research work performs the text classification, the linear transformation process is taken place. Irrespective of number of neural network is used; all linear functions are added to produce efficient function. Also, sigmoid activation function can efficiently deal with back propagation to yield more accuracy. The following three word embedding models are implemented while training the neural network model for the classification problem of movie review dataset.

Embedding Layer + CNN

In embedding layer, the entire vocabulary is converted into vector representation. Words with similar meanings are represented closer in the vector space. This representative is more expressive than traditional methods like bag-of-words. The embedding layer accepts the integer inputs, each of which is mapped to a unique token that has a specific real-value vector representation within the embedding. In the entire feed forward neural network, the words taken from the vocabulary represent the input. These word inputs are converted into vector representation. The vectors are fine tuned by back propagation. During this process, weights are assigned to the first layer known as Embedding layer.

The inputs are represented as w_{j-n+1} , w_{j-n+2} , …. w_{j-1}.The training text corpus has sequence of training words (w1, w2,….wt), belonging to vocabulary V. The size of the vocabulary is defined as|V|. Each word is associated with the input embedding v*w along with the dimensions d and the output embedding v’ * w. The output of the model is computed using the softmax function as in Equation (1).

$$P(w_t) = f(w_t, w_{t-1} \ldots . w_{t-n+1}) \quad (1)$$

Here, the number of words fed into the model is denoted by n.

Based on the above logic and representation, the embedding layer is implemented with CNN in the following manner. The first hidden layer used in the model is the embedding layer, which specifies the vocabulary size, the maximum length of the input documents and the real-value vector space size. The total number of words in the vocabulary plus one is the vocabulary size. The additional one is for the unknown words. The size of vector space used is 100 dimensions. A 32-filterConvolutional Neural Network (CNN) is used in this model with a kernel size of 8 and the ReLu activation function. The next layer is the pooling layer that reduces the output obtained from the convolutional layer by half. The features extracted by the CNN model is flattened and represented as one long 2D vector.

The CNN features are interpreted using standard Multilayer Perceptron layers. In the output layer, the sigmoid activation function is used to map a value between 0 and 1, with zero indicating a negative review and one a positive review. This neural network model is now fit for the training data. The training loss and accuracy are monitored and tracked using the binary cross entropy loss function and stochastic gradient descent, respectively. The model is trained for 30 epochs and evaluated for the reserved test dataset, with the loss and accuracy printed at the end of each epoch. The model achieves 96% accuracy on the training dataset and 82.5% accuracy on the test dataset with the embedding layer and CNN.

Word2Vec + CNN

The standalone word embedding model is developed using an algorithm called word2vec. In word2vec model, each word is represented in vector space as real value vectors. The entire text corpus is a sequence of words. For any word, say Wa, in the text corpus, the context of Wa is obtained from its neighbors on the left and on the right. While converting each word into vector, the probability of obtaining the output for a word in terms of the word input is defined by the softmax of the vector product. The softmax equation is represented in Equation (2).

$$\mathrm{P}\left(\mathrm{w_o}/\mathrm{w_i}\right) = \frac{\left(\mathrm{V_{w_i}} * \mathrm{V_{w_o}^T}\right)}{\sum_{\mathrm{j=1}}^{\mathrm{V}}\left(\mathrm{V_{w_i}} * \mathrm{V_{w_j}^T}\right)} \quad (2)$$

Here $\mathrm{w_o}$ = output word, $\mathrm{w_i}$ = Input word, $\mathrm{V_{w_i}}$= vector representation of input word.

The main objective of the model is to calculate the vector set that maximize the objective function. The objective function and loss function are computed using Eq.(3) and Eq.(4), respectively.

$$\text{Objective Function} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i=1}}^{\mathrm{N}}\sum_{\mathrm{j}\in\mathrm{i}}\log \mathrm{P}\left(\mathrm{w_j}/\mathrm{w_i}\right) \quad (3)$$

Based on the objective function, the loss function can be minimized using the below equation:

$$\text{LossFunction} = -\frac{1}{\mathrm{N}}\sum_{\mathrm{i=1}}^{\mathrm{N}}\sum_{\mathrm{j}\in\mathrm{i}}\log \mathrm{P}\left(\mathrm{w_j}/\mathrm{w_i}\right) \quad (4)$$

Now the document is prepared for embedding which involves data cleaning steps like removing white space, punctuation, and filtering the tokens. Using word2vec algorithm, documents are processed sentence by sentence. While cleaning, sentence based structure is created. The training data is loaded and converted into list of sentences to fit the word2vec model. The first layer is the hidden layer, where word2vec algorithm is used. List of cleaned sentences from the training data is passed to construct the class. The size of vector space is 100, the window size is 5. It represents the maximum distance between the target word and the words around the target word. The number of threads to use when fitting the model is 8. Once the model is fit, the size of the learned vocabulary should match the size of our vocabulary (tokens).

The learned embedding vectors are saved in ASCII format with one word and vector per line. The CNN model uses 32 filters, kernel size 8 with "relu" activation function. The next layer is the pooling layer that reduces the output. The model is trained for 10 epochs. Now the model is tested for the test set. The model achieves 99.5% accuracy on the training dataset and 98.5% accuracy on the test dataset with the Word2Vec and CNN.

Glove + CNN

GloVe stands for Global Vectors for Word Representation. This technique is based on factorizing a matrix of word co-occurrence statistics. GloVe works on co-occurrence value to map the words in vector representation. Co-occurrence value is defined as how frequently two words appear together. The GloVe model works on the logic of the Log Bilinear (LBL) regression model, and uses the simple Weighted Least Squares method.

Let wa,wb be the words in the corpus for word a and word b respectively. The word to word co-occurrence value is calculated based on the log probability of a and b. The co-occurrence of two words is represented in Equation (5).

$$w_a * w_b = \log P(a|b) \tag{5}$$

Also, in Glove, the word meanings are represented as the ratio of conditional probabilities. This model also derives a Target Function represented as F in the below Eq.(6).

$$F(w_a, w_b, \bar{w}_c) = \frac{P_{ac}}{P_{bc}} \tag{6}$$

Here,

w_a, w_b = words available in the corpus context
$\bar{w}_c$ = words from out of the context
P_{ac}, P_{bc} = words derived from the corpus.

While training the glove model, it represents the Target Function F by encoding the values of P_{ac} / P_{bc} present in the entire corpus.

The L.H.S of Equation (6) is the vector space. Since the vector spaces were in linear structures, the Eq. (6) can be rewritten in linear representation as in Equation (7).

$$F(w_a - w_b, \bar{w}_c) = \frac{P_{ac}}{P_{bc}} \tag{7}$$

From Equation (7), it is observed that L.H.S of Equation (7) is in vector form and R.H.S of Equation (7) is in scalar form. The dot product gives the scalar value and is applied to the L.H.S of Equation (7). As a result the L.H.S is matched with R.H.S. The dot product format of Equation (7) is given in Equation (8) below.

$$F\left((w_a - w_b)^T \bar{w}_c\right) = \frac{P_{ac}}{P_{bc}} \tag{8}$$

To achieve invariant symmetry, homomorphism property is applied where the algebraic structure of the two groups are preserved interchangeably. Using homomorphism property, Equation (8) is rewritten as below.

$$F\left((w_a - w_b)^T \bar{w}_c\right) = \frac{F(w_a^T \bar{w}_c)}{F(w_b^T \bar{w}_c)} \tag{9}$$

By solving Equation (8) and Equation (9), we get

$$F\left(w_a^T \bar{w}_c\right) = P_{ac} = \frac{X_{ac}}{X_a} \tag{10}$$

From Equation (10), it is inferred that function F is in exponential form. Now replace F with exponential form, we get Equation (11)

$$e^{(w_a^T \bar{w}_c)} = P_{ac} \tag{11}$$

Now apply log on both sides of Equation (11), we get Equation (12)

$$\left(w_a^T \bar{w}_c\right) = \log(P_{ac}) = \log(X_{ac}) - \log(X_a) \tag{12}$$

Now Equation (12) is simplified further by introducing bias term ba and another bias term for wc as follows.

$$\left(w_a^T \bar{w}_c\right) + b_a + \bar{b}_c = \log(X_{ac}) \tag{13}$$

Using the above equation all the word to word co-occurrences is calculated and the weights were assigned in the word corpus.

Initially, the dataset is cleaned and all text samples in the dataset are converted into sequences of word indices, which are integer IDs for the words. An embedding matrix is prepared, containing an index I, which is the embedding vector for theindexi word in our word index. This embedding matrix is loaded in the first layer and the weights, vector size are assigned. Thereafter, the CNN model is built till the softmax output is reached. Sequences of integers (2D input) are fed to the embedding layer. The input sequences must be padded so that they all have the same length in a batch of input data. The main task of Embedding layer is to map the integer inputs to the vectors found at the corresponding index in the embedding matrix. The output of the Embedding layer will be the shape of (samples, seq_len, embdg_dim). In this model training dataset

is easily learned and gives good accuracy of 96%. This model reaches 92% classification accuracy on the validation set after 28 epochs.

The core classification depends on the convolution operation between the input matrix and various convolution layers. The collected convolution result is used as the data feature for the classification operation. The CNN is composed of convolution layer, pooling layer and classification layer. In the built model, the Word vector matrix is given to the input layer. If there are n numbers of word and the dimension of word vector is d, then the size of input matrix is n x d. Convolution layer and pooling layer are present in the hidden layer. The volume of the text, which is deduced by using several convolution filter sizes, provides the weighted position of the input. n denotes the number of words available in convolution window and d denotes vector dimension of every word.

The product value of h and d extracts the local features. The CNN convolution works as below:

$$\mathrm{Conv}_{\mathrm{res}} = \mathrm{f}\left(\sum \mathrm{W}_1 * \mathrm{X}_{\mathrm{i:i+h-1}} + \mathrm{b}_1\right) \quad (14)$$

$\mathrm{Conv}_{\mathrm{res}}$ denotes the convolution operation result, which is the product of the output matrix and the convolution kernel along with the activation function output after the offset.

Here,

h = Window Size, $\mathrm{X}_{\mathrm{i:i+h-1}}$ = word vector matrix, W_1= Convolution Kernel, b_1= offset, f = Activation function.

Now, all the features were compressed at the pooling layer. Despite the existence of two types of pooling (average pooling and maximum pooling), text classification uses max pooling to the best advantage for optimal classification.

$$\mathrm{MaxPool}_{\mathrm{res}} = \max\{\mathrm{C}_1,\ \mathrm{C}_2, \ldots \mathrm{C}_{\mathrm{n-h+1}}\} \quad (15)$$

Here, the maxpool result value is obtained based on the result of the convolution operation.

The pooling layer acts as an input to this layer. The classification task is done through the softmax function. The classification formula is given below:

$$\mathrm{f(x)}_{\emptyset} = \frac{1}{1 + \exp\left(-\emptyset^{\mathrm{T}}\mathrm{x}\right)} \quad (16)$$

Where, $\exp$ denote the exponential function, $\emptyset$ = Evaluation parameter. The value is estimated by the minimum cost function $\mathrm{J}(\emptyset)$ as given below:

$$\mathrm{J}\left(\emptyset\right) = \sum_{\mathrm{i=1}}^{\mathrm{M}} \mathrm{y}\ (\mathrm{i}) \log \mathrm{f}_{\emptyset}\left(\mathrm{x}^{(\mathrm{i})}\right) \quad (17)$$

The function returns a value that is the probability of multiple components. Now, each component relates to the output category probability. Hence the information of the text category is classified appropriately.

Experimental Design

In this work, a complete analysis and prediction is performed using various word embedding models along with CNN model on the movie review dataset. The embedding layer model carries out processing by considering each word, while theWord2Vec model does so sentence by sentence. The GloVe model uses matrix-based processing with the sigmoid and ReLu activation functions. The vector space size considered is 100, filter size 32 and kernel size 8. As shown in Figure 1, three different models are built with these specifications and the best model is evaluated. Different models achieve various levels of Training Accuracy and Test Accuracy at different Epochs. In earlier studies, only machine learning algorithms along with traditional methods were used for text classification. This study, and its analysis of word embedding models, will help researchers solve word classification problems much more efficiently. This work proved that word embedding models provides better results on text classification problems for larger datasets also.

Performance Evaluation

The performance of the word embedding models is critically analysed with performance metrics to obtain the optimal model.

Performance Analysis

The deep learning model, built with the CNN and the three word embedding models–embedding layer, Word2Vec and GloVe–is evaluated with various performance metrics to derivean optimal model. The metrics

used for evaluation are Train Accuracy, Test Accuracy, Epoch, Precision, Recall, and F1 Score. Various formulas used to calculate the performance metrics are listed from Equation 18 to Equation 32.

Accuracy is calculated as the percentage of correct predictions to the total instances made by the model. The higher the accuracy value, the better the model. The Accuracy formula is given in Equation (18) and Equation (19).

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Instances}} * 100 \tag{18}$$

The accuracy is also represented using the terms TP, TN, FP and FN as,

$$\text{Accuracy} = \frac{(\text{TP}+\text{TN})}{(\text{TP}+\text{TN}+\text{FP}+\text{FN})} \tag{19}$$

Where, TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

Epoch is a major metric parameter used in deep learning models and is defined as the number of times the entire training dataset is processed completely.

$$\text{Epoch} = \text{Forward Pass} + \text{Backward Pass} \; (\forall \; \text{training samples}) \tag{20}$$

Recall, also known as sensitivity, is the ratio of the relevant retrievedinstances.

$$\text{Recall} = \frac{\text{TP}}{(\text{TP}+\text{FN})} \tag{21}$$

Precision is a value that denotes how accurately measurements are made, and represents the ratio of the true positive to the predicted positive.

$$\text{Precision} = \frac{\text{TP}}{(\text{TP}+\text{FP})} \tag{22}$$

The F1 score or F score or F measurecalculatesaccuracy, based on precision and recall.

$$\text{F} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{23}$$

Log Loss is the metric used for comparing models based on the probabilities. A minimal log loss value means betterprediction. The log loss equation is given below:

$$\log\text{loss} = -\frac{1}{\text{N}}\sum_{\text{i}=1}^{\text{N}} \text{Y}_\text{i} * \log\big(\text{P}(\text{Y}_\text{i})\big) + (1-\text{Y}_\text{i}) * \log\big(1-\text{P}(\text{Y}_\text{i})\big) \tag{24}$$

Where, Y_i= Actual class, $\log\big(\text{P}(\text{Y}_\text{i})\big)$ = probability of actual class, $\text{P}(\text{Y}_\text{i})$ = denotes probability of 1, $1-\text{P}(\text{Y}_\text{i})$ = denotes probability of 0.

ROC (receiver operating characteristic curve) is a graph representing the classification model's performance at all the classification thresholds. The curve plotting is based on two parameters namely True positive rate (TPR) and False positive rate (FPR).

$$\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{25}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}} \tag{26}$$

TPR vs FPR is plotted in ROC curve for various classification thresholds. The area under the ROC curve is termed the AUC, which ranges from 0 to 1. A model that predicts 100% accuracy has an AUC value of 1.0

MRR is the Mean Reciprocal Rank. Any system that returns a ranked list of responses to queriesis evaluated using the MRR measure, calculated using Equation 27 below.

$$\text{MRR} = \frac{1}{\text{Q}}\sum_{\text{I}=1}^{\text{Q}} \frac{1}{\text{rank}_\text{i}} \tag{27}$$

Where Q = Multiple Query, rank = highest ranked response position.

The Discounted Cumulative Gain (DCG) is a metric that measures ranking quality from the results retrieved. It is calculated according to Equation 28 below:

$$\text{DCG} = \sum_{\text{I}=1}^{|\text{REL}|} \frac{\text{rel}_\text{i}}{\log_2(\text{i}+1)} \tag{28}$$

Here $|\mathrm{REL}|$ = list of documents retrieved based on relevance, $\mathrm{rel_i}$ = graded relevance.

The performance of each algorithm cannot be comparedefficiently, from one query to the other, by the DCG. So Normalized DCG is calculated as the ratio of sorting all the relevant documents from the corpus to their relative relevance.

$$\mathrm{nDCG_P} = \frac{\mathrm{DCG_P}}{\mathrm{IDCG_P}}$$

Where,

$$\mathrm{IDCG_P} = \sum_{I=1}^{|\mathrm{REL}|} \frac{2^{\mathrm{rel_i}-1}}{\log_2(i+1)} \quad (29)$$

In mean absolute error(MAE), errors are measured between the predicted value and the observed value for the same phenomenon.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n} \quad (30)$$

Where, y_i = predicted value and x_i = observed value.

The Error Rate (ERR) is measured as the ratio of all the incorrect predictions to the total number of predictions in the dataset. For the model to be best the error rate must be 0.0 and if the error rate is 1, it denotes the model as worst.

$$\mathrm{ERR} = \frac{\mathrm{FP}+\mathrm{FN}}{\mathrm{P}+\mathrm{N}} \quad (31)$$

Mathews Correlation Coefficient (MCC) value is calculated based on the TN, TP, FP, FN values present in the confusion matrix. The range varies from -1 to +1. The best model has an MCC score of +1 and the worst a score of -1

$$\mathrm{MCC} = \frac{(\mathrm{TP}*\mathrm{TN})-(\mathrm{FP}*\mathrm{FN})}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TP}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \quad (32)$$

Based on the above mentioned metrics, the three word embedding model is evaluated for various parameters like batch sizes, learning rate and Dropout rate. The performance metric values are listed in Table 2 to Table 10.

Table 2. Performance of various Word Embedding Models based on Log Loss.

Word Embedding Model Name		Batch Size		Learning Rate		Dropout Rate	
		5	**10**	**0.3**	**0.5**	**0.2**	**0.5**
CNN	Embedding Layer +	1.36	1.65	1.27	1.25	2.87	2.41
	Word2Vec + CNN	0.33	0.35	0.30	0.27	0.44	0.35
	Glove + CNN	0.45	0.50	0.41	0.36	1.06	0.72

Table 3. Performance of various Word Embedding Models based on AUC.

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	**10**	**0.3**	**0.5**	**0.2**	**0.5**
Embedding Layer + CNN	0.75	0.71	0.72	0.66	0.73	0.68
Word2Vec + CNN	0.98	0.97	0.95	0.91	0.97	0.95
Glove + CNN	0.81	0.89	0.92	0.85	0.87	0.81

Table 4. Performance of various Word Embedding Models based on MRR.

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	**10**	**0.3**	**0.5**	**0.2**	**0.5**
Embedding Layer + CNN	0.79	0.72	0.79	0.76	0.68	0.66
Word2Vec + CNN	0.88	0.87	0.97	0.95	0.98	0.97
Glove + CNN	0.81	0.84	0.87	0.84	0.89	0.88

Table 5. Performance of various Word Embedding Models based on NDCG.

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	10	0.3	0.5	0.2	0.5
Embedding Layer + CNN	0.658	0.618	0.667	0.651	0.622	0.617
Word2Vec + CNN	0.985	0.961	0.983	0.979	0.969	0.968
Glove + CNN	0.861	0.838	0.841	0.837	0.830	0.827

Table 6. Performance of various Word Embedding Models based on MAE.

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	10	0.3	0.5	0.2	0.5
Embedding Layer + CNN	0.691	0.687	0.670	0.666	0.751	0.750
Word2Vec + CNN	0.935	0.930	0.899	0.897	0.962	0.950
Glove + CNN	0.818	0.815	0.873	0.838	0.891	0.817

Table 7. Performance of various Word Embedding Models based on Train Accuracy.

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	10	0.3	0.5	0.2	0.5
Embedding Layer + CNN	96.5%	96%	94.5%	94.2%	89.9%	89.7%
Word2Vec + CNN	99%	98.5%	98.9%	98.5%	98.5%	98.1%
Glove + CNN	95.5%	94%	96.1%	95.9%	96.2%	96%

Table 8. Performance of various Word Embedding Models based on Test Accuracy.

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	10	0.3	0.5	0.2	0.5
Embedding Layer + CNN	84%	82.5%	85.7%	85.2%	87.2%	86.8%
Word2Vec + CNN	98.7%	98.5%	97.9%	97.4%	98.1%	97.6%
Glove + CNN	92.5%	92%	91.7%	91.1%	89.9%	89.8%

Table 9. Performance of various Word Embedding Models based on Error Rate.

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	10	0.3	0.5	0.2	0.5
Embedding Layer + CNN	0.69	0.72	0.65	0.66	0.59	0.51
Word2Vec + CNN	0.18	0.21	0.22	0.29	0.28	0.19
Glove + CNN	0.46	0.49	0.51	0.53	0.39	0.31

Table 10. Performance of various Word Embedding Models based on MCC

Word Embedding Model Name	Batch Size		Learning Rate		Dropout Rate	
	5	10	0.3	0.5	0.2	0.5
Embedding Layer + CNN	0.68	0.61	0.66	0.61	0.62	0.67
Word2Vec + CNN	0.95	0.91	0.93	0.99	0.96	0.98
Glove + CNN	0.81	0.83	0.81	0.83	0.80	0.82

From Table 2 to Table 10, it is inferred that word2vec embedding model along with CNN yields a value that best suits within the range of the corresponding metrics.

The movie review dataset used in this work consists of 1800 training samples. These samples are trained using three hybrid models to obtain the training accuracy value using various hyper parameters. One of the hyperparameters used is batch size, which varies from 1 to 10. The training accuracy percentage value is obtained for each batch size against three models and listed in Table 11. From the average training accuracy value calculated for all the three models, we infer that the Word2Vec with the CNN achieved the maximum

training accuracy. Hence we conclude that the Word2Vec embedding model with the CNN is the best text classification model.

Table 11. The three embedding models compared in terms of batch size.

Batch Size	Training Accuracy in Percentage		
	Embedding Layer + CNN	Word2Vec + CNN	Glove + CNN
1	94.845	97.631	90.928
2	97.196	98.721	98.193
3	95.941	99.156	89.154
4	89.158	98.943	93.721
5	96.522	99.066	95.599
6	97.490	98.721	98.963
7	88.921	97.911	93.113
8	95.810	99.167	92.992
9	89.733	98.922	90.851
10	96.231	98.515	94.532
Average	94.12%	98.64%	93.7%

The performance evaluation of various embedding models is listed below. Table 12 depicts the Training and test Accuracy values for each of the embedding models used in building the deep learning model. Accuracy values are measured in terms of percentage. Table 12 also lists the number of epochs the model obtained for testing accuracy.

Table 12. Training and testing accuracy compared with the number of epochs.

Word Embedding Models	Training Accuracy	Test Accuracy	Epoch
Embedding Layer + CNN	96%	82.5%	30
Word2Vec + CNN	99.5%	98.5%	10
Glove + CNN	96%	92%	28

Table 13. Precision, Recall and F score of Word Embedding Models

Word Embedding Models	Precision	Recall	F score
Embedding Layer + CNN	0.868	0.887	0.875
Word2Vec + CNN	0.901	0.905	0.901
Glove + CNN	0.874	0.886	0.877

In Table 13, precision, recall and F score values were listed for each of the word embedding models. From these values it is proved that the word2vec model performs better when compared to remaining models in terms of the metrics precision, recall and F score.

DISCUSSION AND CONCLUSION

The various results obtained using the performance metrics and model summary of three different models with CNN is discussed in section 4.1. Table 2 to Table 12 shows the results of three different models when evaluated with various metric parameters like Log loss, AUC, MRR, NDCG, MAE, Test Accuracy, Train Accuracy, Precision, Recall and F score. The Summary of various word embedding models are given in Fig 2 to Figure 4. It is inferred from the table that Word2Vec with CNN model takes less number of epochs and the train and test accuracy are almost same with minor differences. Also the Word2Vec with CNN is optimal model that produces maximum Accuracy in minimum number of epochs. Hence it is the best model among all other models. The results demonstrate that word embedding models outperform traditional classification algorithms. Also, among the three word embedding models, the word2vec model yields higher Train and Test Accuracy at minimum number of epochs. The implementation of CNN yields good results for word classification problems. The unique approach for word classification is implementing hybrid word embedding models along with deep learning model. This approach will hold good for even huge datasets and eradicates

data sparsity issue. The synsets retrieved from wordnet is a tree structure. Based on this structure, the POS disambiguation property filters the most appropriate word at faster rate for all the words obtained after pre-processing. So, there does not exist any time complexity issue while implementing hybrid word embedding model.

The implementation of Hybrid word embedding using wordnet helps to attain the semantic nature and performs the text classification. The synsets retrieved based on functions like disambiguation and hypernym.This work concludes that deep neural networks produce optimal results with hybrid word embedding algorithms, surpassing those produced by machine learning algorithms for classification problems. The transformer models will handle with text data. But the data is not processed in any order. This research work implements hybrid word embedding, where all the text are processed in an order using wordnet. Hence the word embedding model is efficient than the transformer model. Future enhancements include the use of two possible approaches to obtain the best results in the fastest possible time. In the first approach, hybrid neural networks can be implemented to yield more optimal solution. The current research work implemented text classification using CNN and attained accurate classification results. The accuracy can still be improved by implementing hybrid neural network, where two neural networks are used and the output of first neural network is passed as input to the second network. By implementing this hybrid nature, more accurate classification can be achieved for huge dataset also. The second approach includes applying a single-layer, rather than multiple-layer, multi-sized filters in the neural network model. These two approaches are considered for future enhancements in classification problems.

REFERENCES

1. Albalawi Y, Buckley J,Nikolov S. Investigating the impact of pre-processing techniques and pre-trained word embeddings in detecting Arabic health information on social media. J Big Data. 2021 Jul; 8(95):488-21.
2. Andrei M Butnaru, Radu Tudor Ionescu. From Image to Text Classification: A Novel Approach based on Clustering Word Embeddings.21st International Conference on Knowledge Based and Intelligent Information and Engineering Systems; 2017 Sep 6-8; Marseille, France. Elsevier B V; c2017. p.1783-1792.
3. Bozyigit, Kılınc D, Ozcift A, Yıldırım P, Yücalar F, Borandag E. TTC-3600: A new benchmark dataset for Turkish text categorization. J Inf Sci. 2017 Dec; 43(2): 174-85. Doi:10.1177/0165551515620551.
4. Conneau A, Schwenk H, Barrault L, Lecun Y. Very deep convolutional networks for text classification.15th Conference of the European Chapter of the Association for Computational Linguistics; 2017 Apr 3-7; Valencia, Spain. Association for Computational Linguistics; c2017. p.1107-1116.
5. Ge N, lu J, Wang Y, Howard N, Chen P, Tao X, et al. Visualization of big data. 14th International Conference on Cognitive Informatics & Cognitive Computing; 2015 Jul 6-8; Beijing, China.IEEE Computer Society Proceedings; c2015. 447p.
6. Guilherme Gomes B, Murai F, Goussevskaia O, Couto da Silva AP. Sequence-Based Word Embeddings for Effective Text Classification. Natural Language Processing and Information Systems.26th International Conference on Applications of Natural Language to Information Systems; 2021Jun 23-25; Saarbrücken, Germany.Springer, Cham; c2021. p.135-146.
7. Kim Y. Convolutional neural networks for sentence classification. In Proceedings of the Conference on Empirical Methods in Natual Language Processing (EMNLP); 2014 Oct; Doha, Qatar. Association for Computational Linguistics; c2014. p.1746-1751.
8. Kilimci Zeynep H,Akyokus S. Deep Learning- and Word Embedding-Based Heterogeneous Classifier Ensembles for Text Classification. Hindawi Complexity. 2018 Oct; 2018(7): 1-10.
9. Kim HK, Cho S. Bag-of-concepts: comprehending document representation through clustering words in distributed representation. Neurocomputing. 2017 Nov 29; 266: 336–52.
10. Le Quoc, Mikolov T. Distributed representations of sentences and documents. 31st International Conference on Machine Learning; 2014 Jun; Beijing, China. Proceedings of Machine Learning Research; c2014. p.1188-1196.
11. Li T, Gao M, Huang P. Text Classification Research Based on Improved Word2vec and CNN.The 16thInternational conference on service oriented computing; 2018 Nov 12-15; Hangzhou, Zhejiang, China. Springer; c2019. p. 126–135.
12. Mark Hughes, Irene LI, Kotoulas, Suzumura. Medical Text Classification Using Convolutional Neural Networks. Studies in Health Technology and Informatics. 2017 Apr; 235:246-250.
13. Ming T, Lei Z, Xianchun Z. Document vector representation based on Word2Vec. Comput. Sci., 2018 Dec; 43(6): 214-7.
14. Moreo A, Esuli A,Sebastiani F. Word-class embeddings for multiclass text classification. Data Mining and Knowledge Discovery. 2021 Feb; 35(3):911–63.

15. Pennington, Socher R, Manning C, Glove.Global vectors for word representation. In the Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014 Oct; Doha, Qatar. Association for Computational Linguistics; c2015. p.1532–1543.
16. Pittaras N, Giannakopoulos G, PapadakisG, Karkaletsis V. Text classification with semantically enriched word embeddings. Nat. Lang. Eng. 2020 Apr; 27(4): 391-425.
17. Roger Alan Stein, Patricia Jaques A, João Francisco Valiati. An analysis of hierarchical text classification using word embeddings. Inf. Sci. 2019 Jan; 471:216-32.
18. Weston J, Collobert R. A unified architecture for natural language processing: Deep neural networks with multitask learning. 25th International conference on Machine learning; 2008 Jul 5-9; Helsinki, Finland. ACM Digital Library; c2008. p.160-167.
19. Yao L, Mao C, Luo Y.Graph Convolutional Networks for Text Classification. The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19); 2019 Jan 27 – Feb 1; Hilton Hawaiian Village, Honolulu, Hawaii, USA. ACM Digital Library; c2022.p.7370-7377.