Multimedia Tools and Applications (2020) 79:32389–32413
https://doi.org/10.1007/s11042-020-09632-9

Improved method of word embedding for efficient analysis of human sentiments

Santwana Sagnika[1] · **Bhabani Shankar Prasad Mishra**[1] · **Saroj K. Meher**[2]

Received: 15 April 2020 / Revised: 10 August 2020 / Accepted: 13 August 2020 /
Published online: 27 August 2020

Abstract

User database of the internet is expanding at a swift rate with the dramatic growth of social media. These include information as well as personal opinions about products, ideas, news, politics, etc. These online opinions and reviews act as a word-to-mouth medium for enhancing or diminishing the popularity of a product, item or concept. Thus, automated analysis of the tone of online opinions helps customers and business personnel significantly to take decisions and develop strategies efficiently. This task, known as sentiment analysis, is an area of active research that relies heavily on the text processing methodology called word embedding. Word embedding is a process of representing text into numeric format, to enable mathematical operations on them. The present study proposes a method of enhancing the performance of word embedding approaches, by integrating sentiment-based information, to render them more suitable for sentiment analysis. Sentiment-based information is incorporated through self-organizing map, where similarity is calculated based on the scores of sentiment-based words. The similarity is further tuned using particle swarm optimization method. Experimentally, performance of the proposed method is justified for sentiment analysis task using various classifiers. Different performance measurement indexes are used to validate the superiority of the proposed method compared to existing approaches.

Keywords Sentiment analysis · Opinion mining · Natural language processing · Word embedding · Self-organizing map · Particle swarm optimization

✉ Santwana Sagnika
santwana.sagnika@gmail.com

Bhabani Shankar Prasad Mishra
mishra.bsp@gmail.com

Saroj K. Meher
saroj.meher@isibang.ac.in

[1] School of Computer Engineering, Kalinga Institute of Industrial Technology Deemed to be University, Bhubaneswar, Odisha, 751024, India

[2] Systems Science and Informatics Unit, Indian Statistical Institute, Bangalore Centre, 8th Mile, Mysore Road, RVCE Post, Bangalore, 560059, India

1 Introduction

Advancement of technologies leads to the generation of a huge amount of data. This includes a substantial amount of views, opinions and judgments about almost every topic and object. Such data become useful to a wide range of users and entrepreneurs to know about a product before dealing with them. For example, a movie-goer can read the review of a movie and decide whether to watch it or not. Similarly, a company can access the ratings of their products and address the problems for customer satisfaction. A political party can follow the public trend and work upon their campaign plan. In this scenario, it becomes useful to collect, analyze and quantify the mass of data. This process is known as sentiment analysis or opinion mining [17].

Efficient sentiment analysis can facilitate decision making for users as well as businesses, by identifying the general conception of people towards any topic. Since a majority of such data is textual, manual annotation and categorization is an accurate approach. But it is extremely time-consuming, since user-generated data grows exponentially on online platforms. More manual effort is needed, and this leads to increased cost and disagreement among different annotators. To reduce manpower, time and effort, it is necessary to automate the entire process. But, such kind of automatic analysis of data without direct intervention of humans is a challenging task, owing to the fact that the data is huge, varied and unstructured. This implies that an automation technique should be scalable, robust and flexible, in order to deal with increasing quantities and heterogeneous types of data. As a result, sentiment analysis has been an area of active research in recent times [14, 28].

Technically, automated sentiment analysis refers to identification and categorization of textual opinion into pre-defined sentiment categories. The categorization can be broad or fine, depending upon the requirement. The automation process generally involves text mining, i.e. natural language processing, which involves reading and analyzing the meaning from text. Other aspects of sentiment analysis include subjectivity classification, spam detection, aspect extraction, review usefulness measurement, etc. Sentiment analysis finds application in the field of data analytics and decision making in the personal level, as well as in business, news, political and marketing domains [5, 12].

Sentiment analysis and its related tasks face multiple challenges. Language specific issues, such as context information, multiple meanings of the same word, hybridization of different languages, figures of speech like irony and sarcasm, shortcut words and emoticons, etc. make it difficult to process the text. Lack of lexicons and corpora, especially in non-English languages, make it difficult to perform accurate analysis. [28] Even though manual effort addresses some of these issues, it leads to more problems like discrepancies between annotators, difference in quality of processing, etc. To address these problems and generate uniform results through automated techniques, statistical and lexicon based approaches have been developed. A substantial amount of work using statistical and lexicon based methods exists in the literature of sentiment analysis. Recent popularity of neural networks has led to a shift from lexicon based methods to neural network based approaches, which are faster, more accurate, and require negligible manual intervention. Neural networks can also provide remarkable results in situations where there is a lack of sufficient lexicons or annotated corpora. An explicit description of the state-of-the-art solutions to sentiment analysis is provided in the Related Work section of this paper. In this work, we have leveraged the power of neural networks to generate features from words, and enhanced those features using information from existing lexicons. These features are then used to perform efficient sentiment analysis. The methods used are detailed in subsequent sections.

The rest of the paper in organized as follows. Section 2 contains some of the relevant related literature in the area. Section 3 explains the motivation and need behind the work carried out in this paper. Section 4 elaborates the proposed word embedding method and details the various steps and concepts used in each step. Section 5 gives details on the experiments performed. Section 6 mentions the indices used for performance evaluation of the work. Section 7 discusses the obtained results and their implications. Section 8 contains the conclusion and future scope of this work.

2 Related work

For the sentiment analysis task, several attempts have been made to solve problems in a multitude of domains using machine learning methods [13, 20–22, 32], which was otherwise difficult owing to the intractability of processing large-scale data by traditional approaches. For example, polarity detection, i.e. categorizing an opinion into positive or negative, is essentially a classification problem, and hence, classifiers like support vector machine (SVM), naïve bayes (NB), random forest (RF), etc. have proved to be extremely useful and efficient [1]. Many research works are carried out using supervised [22], unsupervised [32] and semi-supervised [13, 20] techniques on all kinds of domains, like product reviews, movie reviews, news, tweets, etc. Pang et al. [22] have demonstrated the application of naive bayes, maximum entropy and support vector machine classifiers on sentiment analysis, which are the initial works in this domain. This work became the basis of many further applications of supervised learning methods. Again, Pang and Lee [21] have also presented a movie review dataset that has been used as a standard dataset for sentiment analysis tasks. Caschera et al. [4] have provided a technique to extract sentiments from multimodal data by combining language-based formalization along with a machine learning approach, and used hidden Markov model to represent extracted emotions for the purpose of classification. Unsupervised learning methods have also been applied by Turney [32], who employed part-of-speech tagging to identify recommendable reviews. Recently, Ju et al. [13] have described semi-supervised learning by under-utilization techniques for sentiment analysis in case of inadequate availability of labelled data. Ortigosa-Hernandez et al. [20] demonstrated the solution of a multi-dimensional classification problem using a semi-supervised Bayesian classifier to identify subjectivity, polarity and will-to-influence aspects.

These research works motivated us to use machine learning approach for automatic and minimum computation burdened sentiment analysis. However, efficiency of these methods is mostly controlled by the reliable training data that represent all possible information of the system in hand. Owing to the fact that user-generated data is largely unstructured and dissimilar, it is necessary to represent the data in a standard and comprehensible manner, which can be well-understood by a system for further analysis. Data pre-processing thus becomes an essential step in order to analyze this information [11]. There exists number of techniques to address this, such as bag-of-words, tf-idf, co-occurrence matrix, etc. These methods generate numeric vectors that represent specific features of the text, and these numeric vectors enable mathematical operations on them. Recently, the Word2Vec and GloVe techniques have become popular as word-vector generating algorithms. They create word embeddings, i.e. numerical vectors for individual words using neural networks. These embedding techniques can be applied to any text mining-based method, including sentiment analysis.

Researchers have attempted enhancements on word embeddings by adding sentiment information to them. Cano and Maurisio [3] performed an empirical study on the impact

of factors like training technique, size and thematic content of training corpus on word embeddings, and tested them on sentiment analysis on multiple domains. Their research suggests that the impact varies across domains, and by integrating extra information through post-processing, the sentiment analysis results is enhanced. Rudkowsky et al. [27] prove the strength of using language dependent word embeddings for sentiment analysis in the field of social sciences by applying them on estimating negativity levels in parliamentary speeches. Maas et al. [18] have used a model combining supervised and unsupervised techniques that captures semantic and sentiment information, and used it for document level sentiment analysis to provide greater accuracy for movie reviews. Tang et al. [31] have designed three neural network architectures to generate improved word embeddings which are sentiment-specific, and experimented on Twitter data. Zhang and Lan [37] have also produced sentiment integrated word embeddings using two models which are extensions of the continuous skip-gram model. They have tested these embeddings on English and Chinese data, to achieve improvement over the use of standard word embeddings. Rezaeinia et al. [26] have created improved word vectors that include Part-of-Speech (POS) information and sentiment scores as additional vectors to the existing word vectors, thereby enhancing their usage in sentiment analysis tasks. Yu et al. [36] have proposed an effective refinement model for word vectors by shifting the vectors to be closer to sentimentally similar and farther from sentimentally dissimilar words. Dragoni and Petrucci [7] have used neural word embeddings that are generated based on overlap between domains to facilitate cross-domain sentiment analysis. Fu et al. [10] have developed an architecture for sentiment-specific word embedding which is applicable to both sentence-level and document-level sentiment analysis, by utilizing both local and global sentiment context. Their technique shows improvements over generalized word embedding techniques.

In order to address the word embedding task, the objective for the present work is to improve the accuracy of sentiment analysis process, by enhancing the performance of conventional word embeddings obtained by Word2Vec and GloVe. For this purpose, the enhancement is made based on sentiment-based scores of individual words. By modifying the generalized word vectors to include sentiment scores, the vectors become more suitable for sentiment analysis. The present study has in fact drawn inspiration from the techniques put forth by [26] and [36], for developing an efficient method of creating modified word embeddings that are suitable for sentiment analysis. In this work, the authors attempt to represent input data as reflective of sentiment information, where the data is preprocessed to form word vectors or embeddings based on sentiment scores of the respective words. First, words are clustered using a Self-Organizing Map (SOM) according to their sentiment ratings. Based on the closeness of the word to other words, the original word representation is modified to get a new word representation using Particle Swarm Optimization (PSO). This new word representation is thus more suitable for the sentiment analysis task.

3 Motivation

Sentiment analysis is a classification problem, which takes a portion of text as input, and classifies it into one of the specified categories of sentiment. There are a number of efficient classifiers that are popularly used for such classification problems. Recently, classifiers based on neural networks have been effectively applied to a variety of domains, giving high accuracy on complex data. Neural network-based classifiers work on a specific mathematical model and provide a fixed number of outputs, wherein each output represents a target class, as defined by the problem. The input layer takes a fixed number of inputs, where each

input is a feature of the data. Thus, the input data has to be represented in an appropriate manner, so that it can be fed to the input layer. Sentiment analysis is a text processing problem, where the input data is usually a set of sentences, paragraphs or documents containing the opinion of the writer. This data is unstructured and needs to be preprocessed for the input layer. A common preprocessing technique is to represent the sentences as vectors of a fixed size. Word embeddings are one such kind of vector representations for textual data, which represent words as numeric vectors [18].

In the initial approaches to word embeddings, the vector size was equal to the number of words in the vocabulary. In the current scenario of large data size, this technique has become infeasible due to memory requirements. Subsequent methods like count vector, co-occurrence matrix and TF-IDF vector were more efficient and useful, as they considered word frequency in the text to determine their relevance to the subject. Despite their usefulness, these vectors represent limited information [29]. The application of neural networks in the field of word embeddings has generated prediction-based embeddings, which are based on probability and contain useful information like similarity, oddity, analogies, etc. These techniques are useful to classifiers for performing text analytics. Currently, Word2Vec and GloVe are the most popular techniques for word embeddings, and have proven useful for multiple natural language processing tasks. A brief description of both the word embedding techniques is made as follows.

Word2Vec Word Embeddings Word2Vec is a technique of word embedding introduced by Mikolov [19] in 2013. It uses a combination of two neural network models: the Continuous Bag of Words (CBOW) and the Skip-gram model, which use context information to generate word vectors according to the weights of neurons [29]. The generated vectors are based on the weightage of the word in the entire text corpus and hence hold extremely useful and versatile information, which can be used for various applications like finding word similarities, odd one out among a set of words, etc. In situations where there is less training data available, Word2Vec does not get enough information for proper training. Here, instead of training the model on the limited dataset, pre-trained word vectors can be used for better performance [35]. Google provides one such set of pre-trained word vectors, containing approximately 3 million words from its news dataset [6].

GloVe Word Embeddings GloVe, referring to Global Vectors, is another method for modeling words into vectors. Designed by Pennington [23] in 2014, it creates a word vector space by training on word-word co-occurrence counts. The models are trained on Wikipedia dumps, Gigaword 5 and Common Crawl texts, and apply unsupervised learning to create 300-dimensional vectors. The vectors represent semantic similarity between words. Their usage is similar to Word2Vec embeddings [24]. Both Word2Vec and GloVe are extremely useful when the dataset is small, and provide a better option of representing word features. The performances of both methods are comparable, and either can be better depending on the application areas.

Need for improving word embeddings for Sentiment Analysis Most of the time, generating word embeddings by applying Word2Vec or GloVe on the dataset in hand is the best way to discover proper relativity between the words. This gives higher accuracy when tested on the same domain. But this method has two major problems. Firstly, overfitting of the model takes place. This renders the model extremely suitable for the current domain but leads to lower accuracy on other datasets and domains. An ideal model should be

general enough to provide considerable accuracy on multiple datasets and domains. Secondly, as already mentioned, many situations do not have sufficient labeled data which can be used to properly train a model. Considering both these issues, it is advisable to use word embeddings which have been pre-trained on larger corpuses, and are more universal. Being generic in nature, these pre-trained Word2Vec or GloVe word embeddings have certain drawbacks when applied to specific purposes. They do not consider context of the text being trained. As a result, the word "Apple" will have the same vector representation, whether it is used as a brand name or a fruit name. In terms of sentiment analysis, the disadvantage is that there is no sentiment information being considered, and hence words like "good", "bad", "awesome", and "terrible" have similar vectors close to each other, all of them being adjectives. This leads to a drop in accuracy for the machine learning model [26].

In order to address these issues, the authors have proposed an efficient method that is incorporated in the process of sentiment analysis. The objective in the present study is to effectively perform the word embedding task so that the efficiency of the sentiment analysis model can be enhanced.

4 Proposed word embedding method for sentiment analysis

The schematic flow diagram of the proposed sentiment analysis model using an effective word embedding approach is shown in Fig. 1. The operational steps of the proposed model are similar to the generic sentiment analysis model, except the operation performed in the word embedding step. This paper aims to provide an improved method of informative word vector representation (the third step in Fig. 1). A brief description of the operational steps of the proposed sentiment analysis model is made in the following subsection.

The different steps of the workflow are detailed as follows.

4.1 Preprocessing

The input text is generally in the form of a review, tweet, or blog post. The purpose is to categorize it to a class (positive/negative etc.), or to a rating (1 star/5 stars etc.). The input text is unstructured and noisy. The first step is to clean the text by considering the individual tokens of the text, such as words, numbers or symbols, and removing symbols, hypertext, etc. which are not relevant to the sentiment presented by the text. Stopwords like *a, the, of,*

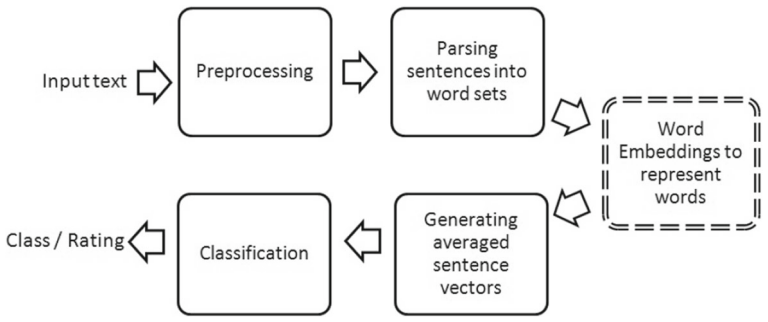


Fig. 1 Semantic representation of the workflow

is, etc. are also removed to provide better understandability. Finally the words are stemmed, i.e. reduced to their base form, by removing tense and converting them to singular. In this way, the general preprocessing is done.

4.2 Parsing

This step deals with specific preprocessing required by the model used in this work. Once cleaned, the reviews are parsed sentence-wise, representing each sentence as a group of words. A corpus is created out of all the sentences split in this manner. This is done because the method considers reviews as an aggregate of words. These words can combinedly represent the sentiment of the entire review.

4.3 Generating word embeddings to represent words

This step deals with feature representation, i.e. preparing the text in the form of numerical inputs, to make them ready to be fed into a classifier for analysis. In this step, a word embedding model is taken, which represents each word as a numeric vector of fixed dimensions. Various techniques exist for this, and the authors take neural network based embedding methods, namely Word2Vec and GloVe, which are previously explained.

Instead of using generalized pre-defined word embeddings, the authors modify the word vectors to include the sentiment of the word, based on the part of speech the word belongs to. The modification enhances the performance of sentiment classification of the model which is trained on the modified word embeddings. This modification is done based on how close the word is to other words that share similar sentiment with it.

In this work, the authors have modified word embeddings generated by Word2Vec and GloVe using sentiment information based on a neural network algorithm known as self-organizing map (SOM). The modification takes place based on sentiments, i.e. sentimentally similar words (e.g. good and awesome) are clustered together, while sentimentally dissimilar words (e.g. good and bad) are moved farther from each other. The modified word embeddings are then used for sentiment analysis, and tested on different datasets and classifiers.

The technique proposed in this paper for modification of word embeddings can be described according to the following algorithm.

4.3.1 Proposed algorithm – adjusting word embeddings

1. Take a set of generalized pre-trained word embeddings, created by Word2Vec or GloVe.
2. Use a sentiment lexicon which contains words and their sentiment ratings.
3. Identify the affect words using Part-of-Speech (PoS) tagging.
4. Create a two-dimensional (2-D) mapping of affect words in the sentiment lexicon, using a Self-Organizing Map (SOM).
5. For an affect word A, find its pre-trained word embedding W.
6. Find its grid location G(x,y) from the 2-D mapping.
7. Find all other words present in G, which is the set of sentimentally similar words for A.
8. Adjust the word embedding W to W', using Particle Swarm Optimization (PSO) algorithm, and Eqn. 4 as its fitness function.
9. Replace the pre-trained word embedding W with the modified word embedding W'.

10. Repeat steps 5 to 9, for all affect words in the vocabulary.

The algorithm steps are detailed as follows.

4.3.2 Initial preparation (Steps 1-3)

Initially a sentiment lexicon is taken which contains sentiment scores of various words. For this work, the authors take E-ANEW [2] as the sentiment lexicon which contains words and their valence scores in a range of 0 to 10, 0 being the most negative and 10 being the most positive. A Part-of-Speech tagging is done to identify the affect words (words that convey feeling or emotion) in the lexicon.

4.3.3 Finding sentimentally alike words (Steps 4-7)

The identified affect words are considered and a self-organizing map is created to cluster these words based on their sentiment scores. The concept of a self-organizing map is explained as follows.

4.3.4 Self-organizing maps

Kohonen, in 1982, described a method of dimensionality reduction and clustering using neural networks, known as a Self-Organizing Map (SOM) [15, 16]. The concept behind SOM is to create a mapping that projects multi-dimensional data into a simple lower dimensional grid, generally two-dimensional, especially suitable for visualization. It works as a clustering method, since it preserves the proximity of data points in the generated 2-D map. Experiments show that the clustering performance is comparable with the most popular clustering techniques like k-means [8, 34].

SOM is an unsupervised neural network having only an input and an output layer. It follows a competitive learning approach to adjust the weights of the neurons in the output layer. It preserves topological connections between the input data, which means data points which are closer to each other in the multi-dimensional space, are also closer to each other in the output two-dimensional grid [30]. Figure 2 shows the concept of a SOM.

Every neuron in the output grid has a weight vector. The points in the input space are mapped onto one of the points in the output grid, whose weight vector is the nearest to the input point. Most of the time this leads to multiple input data points being mapped to the

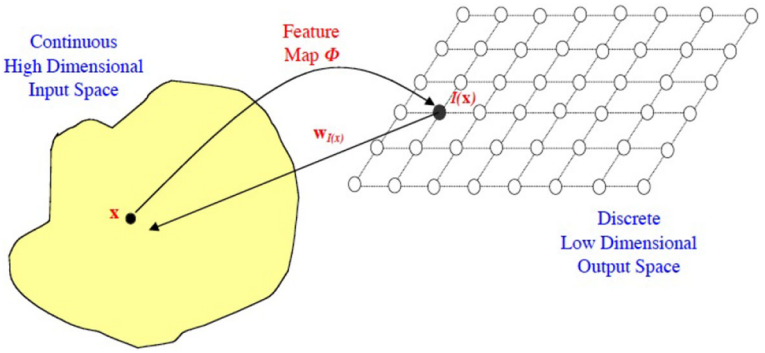


Fig. 2 Task performed by a SOM [30]

same output data point. This helps in simplification of visualization of the data, as well as clustering of the data. A simple algorithmic approach to implement a SOM is described as follows [29, 30].

4.3.5 Algorithm – implementing a SOM

1. Let the input data be a set of points I = $\{i_1, i_2, \ldots, i_p\} \in \Re^m$, where m is the number of dimensions. Let the output grid consist of nodes O = $\{o_{xy};\ x \in [0, p]\ and\ y \in [0, q]\ \}$, where $p \times q$ is the output grid size. Let α be the learning rate. r is the initial neighbor radius.
2. Initialize each node's weight randomly. W = $\{w_{xy};\ x \in [0, p]\ and\ y \in [0, q]\}$
3. For input data point i_k, find its distance from all output nodes, i.e. $d_{xy} = dist(i_k, w_{xy})$, where dist() is the Euclidean distance.
4. Find the node which gives the minimum distance, denoted by o_{win}, and its associated weight as w_{win}.
5. Locate all neighbouring nodes of o_{win}, using a neighbouring function $N(w_{win}, w_{xy}, r)$, which is a function of the iteration number and an initial neighbor radius, for the two nodes w_{win} and w_{xy}.
6. Update the weights of all neighbouring nodes to become closer to o_{win}. Mathematically,

$$w_{xy} = w_{xy} + \alpha * dist(i_k, w_{xy}) * N(w_{win}, w_{xy}, r) \tag{1}$$

7. Decrease values of α and r.
8. Repeat steps 2-6 till α approaches 0.

The neighbourhood function *N()* is designed such that the nodes closer to the winner node o_{win} will be updated by an amount larger than the farther nodes, which will be updated by a smaller amount. As the number of iterations progresses, the neighbor radius also decreases, which isolates the winner node more and more from the effect of its neighbours. On the other hand, the learning rate decreases over time, which eventually leads to convergence of the algorithm.

In this work, the authors have used a SOM for clustering of words. The clustering is done taking the sentiment ratings of the words as input vectors. The words in the same output grid are considered similar to each other sentimentally, while words in distant grids are considered sentimentally dissimilar. The SOM also maps this clustering into a two-dimensional vector, which makes it easy and less time-consuming to access the closest words. Thus, the words which have similar sentiment scores tend to get allotted to the same or nearby grid points in the map, whereas the words with dissimilar scores get allotted to grid points which are farther apart, as shown in Fig. 3. After this map is created, it is used for the next step, which is the modification of word embeddings of affect words.

4.3.6 Modifying words according to the SOM (Step 8)

For a given affect word, first all the words in the same grid point are identified. Then the vector of the word is modified, to move it closer to the vectors of all other affect words in the same grid point. This modification is done, keeping in mind that the word maintains almost equal distances from all similar words, while not moving too far from its original vector, so that its identity is preserved. By performing this modification, it is ensured that the affect words are closer to sentimentally alike words, and are farther from sentimentally different words. This is because the SOM makes sure that words with similar sentiment are clustered

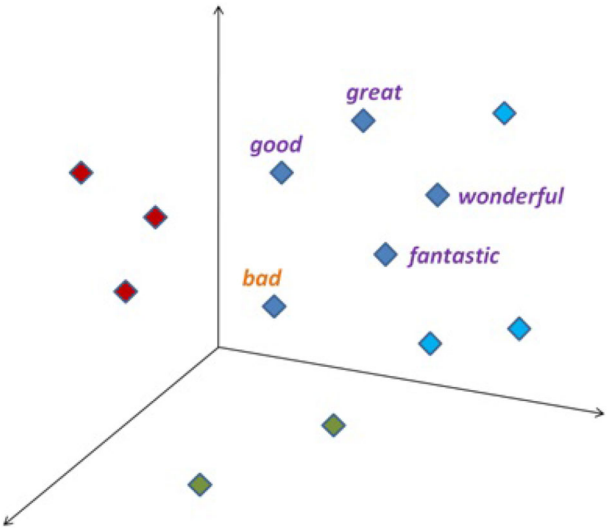


Fig. 3 An example of word vectors in a lower dimensional space generated by SOM

closer to each other and words with different sentiments are clustered separately. Thus, words like "good" and "awesome" will be closer to each other, being sentimentally positive. Similarly, words like "bad" and "terrible" will be closer to each other, being sentimentally negative. At the same time, both these groups of words will be farther from each other, due to the modification technique applied. Figure 4 shows an example, where the word "good" is progressively moved towards similar words like "great", "beautiful" and "fantastic", while simultaneously moving away from the dissimilar word "bad".

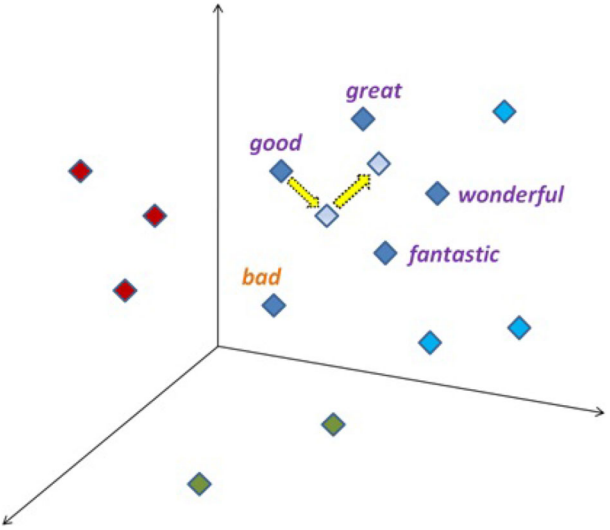


Fig. 4 An example of adjusting a word vector in the grid space generated by SOM

Let W be a word embedding for a random affect word, generated by a word embedding algorithm (Word2Vec or GloVe). The modified vector W' can be generated by shifting W closer to sentimentally similar words, but not too far from its original position W. Hence, this modification can be represented as a mathematical equation having two parts.

Let distance of W' from $W = D(W', W)$

Let total distance of W' from sentimentally alike words = $\sum_{k=1}^{n} D(W', W_k)$

where, W_k is the k^{th} sentimentally alike word to W, and n is the total number of sentimentally alike words.

There are various ways to calculate distances between two vectors. The authors have used one of the most widely used methods, i.e. Euclidean distance. *ED(x,y)* represents the Euclidean distance between two vectors x and y, given by

$$ED(x, y) = \sqrt{\sum_{d=1}^{m}(x_d - y_d)^2} \tag{2}$$

where m is the number of dimensions of the vectors. *D(x,y)* represents the square of Euclidean distance.

$$D(x, y) = ED^2(x, y) \tag{3}$$

To find the optimal value of W', the sum of both distances mentioned above need to be minimized. Depending on how much distance it is desirable for the embedding to be shifted from its original position, the corresponding weightage can be assigned to each distance component. Thus, the shifting process can be expressed as

$$Minimize\{\sigma * D(W', W) + (1 - \sigma) * \sum_{k=1}^{n} D(W', W_k)\} \tag{4}$$

Here, $\sigma \in \{0, 1\}$ is the weightage parameter assigned to control the movement of the word embedding. This helps maintain proportion between the two parts of the equation. A greater value of σ keeps W' closer to the original embedding W, whereas a smaller value of σ moves W' closer to the sentimentally alike words.

Equation 4 is the objective function for this problem, which is solved using an optimization algorithm, i.e. Particle Swarm Optimization (PSO). The concept of PSO is explained as follows.

4.3.7 Particle swarm optimization

Kennedy and Eberhart [9] in 1995 introduced the concept of PSO, which follows the behavior of birds searching for food. It is a swift and efficient technique that performs optimization on large search spaces, which is a NP-Complete problem. It uses a population of birds/particles starting at random locations, which search their neighborhood for food, i.e. solutions. Depending on their proximity to the solutions, they decide whether to proceed in that direction or change the search direction. Each bird retains its own best position (local best), and uses it to direct its search. This best data is also shared with other birds, to find out the best position among all birds (global best). Together, the local best and global best combine to direct the new velocity, and consequently position, so that the birds converge at the best solution after a certain number of iterations. This makes the method work in parallel, thereby achieving optimal results in a feasible amount of time [25, 33]. Figure 5 shows the working steps of a basic PSO algorithm.

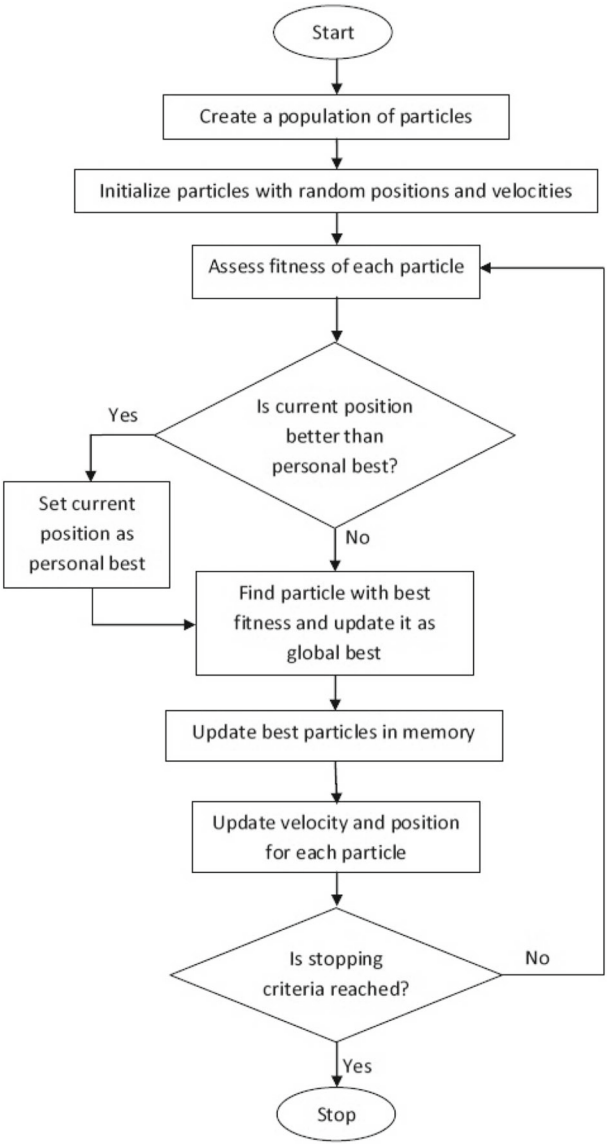


Fig. 5 Steps of the PSO algorithm

Here, the position and velocity of the particles is determined as follows.

$$V_i^{new} = V_i^{old} + a_1 * r_1 * (pbest_i^{old} - P_i^{old}) + a_2 * r_2 * (gbest^{old} - P_i^{old}) \quad (5)$$

$$P_i^{new} = P_i^{old} + V_i^{new} \quad (6)$$

In these equations, V_i^{old} is the velocity of i^{th} particle in the previous generation, V_i^{new} is the velocity of i^{th} particle in the current generation, P_i^{old} is the position of i^{th} particle in the previous generation, P_i^{new} is the position of i^{th} particle in the current generation, a_1 and a_2 are accelerating factors of local and global information respectively, r_1 and r_2 are random

values between 0 and 1. $pbest_i$ is the personal best of the i^{th} particle and $gbest$ is the global best among all particles [33].

In this work, PSO is employed to minimize the fitness function as mentioned in Equation 4.There are numerous swarm-based methods available for optimization. The authors chose PSO because of its simplicity and fast convergence in lesser time and iterations, which is beneficial for the task in hand. For the given affect word, a population of particles is taken, where each particle represents a solution, i.e. a possible modified word embedding for a word. The fitness of each particle is assessed, and the local and global best are found out. New velocity and positions are calculated. This process goes on iteratively till the best solution is found, which is the best possible modified word embedding for the affect word. For all the affect words, the PSO algorithm is re-executed with a fresh set of population particles. At the end, the set of modified word embeddings is obtained for the vocabulary of the pre-trained word embeddings.

4.3.8 Final steps (Steps 9-10)

The obtained word embedding is used to replace the original word embedding of the pre-trained Word2Vec or GloVe set. This is done for all affect words in the vocabulary. The remaining word embeddings stay as they are. Now the further steps of training the classifier can go on, using the modified set of word embeddings.

4.4 Averaged word vector generation

After obtaining the modified word embeddings, sentence vectors are generated by taking the average of the vectors of all words contained in a sentence. This is extended to obtain vectors for the entire input text. Now the input is ready in a numeric form to be passed to a classifier. The set of input vectors are then split into training and testing sets.

4.5 Classification

The training set feature vectors are used to train a classifier, against their class or rating. The authors experiment with various classifiers to find more suitable ones for sentiment analysis. The trained classifier is then tested on the testing set vectors to predict their class or rating. Parameters like accuracy, precision, recall and kappa coefficient are utilized to quantify the performance.

5 Implementation details

5.1 Datasets used

To demonstrate the effectiveness of the proposed sentiment analysis using a modified word embedding approach, we have used two different datasets in the present study. One is the International Movie Database (IMDb) movies reviews dataset, and the other is the Yelp dataset that contains restaurant reviews and corresponding star ratings.

The IMDb dataset is a collection of movie reviews and their ratings. This dataset contains 25,000 reviews, and their sentiment scores. The scores are scaled as per the ratings, i.e. ratings less than 5 have a score of 0, and above 7 have a score of 1, in order to make the dataset suitable for a binary classification problem.

The Yelp dataset consists of 10,000 restaurant reviews, having the star ratings between 1 and 5. However, the dataset is reduced to include only 1-star and 5-star ratings, in order to make it a binary classification problem.

5.2 Classifiers used

For a comparative analysis of the proposed sentiment classification model, we have used different classifiers that are given both the generalized word embedding (Word2Vec and GloVe) based inputs along with their modified versions. The classifiers considered here are Gaussian naive bayes, random forest, decision tree, gradient boosting, support vector machine, multi-layer perceptron, convolutional neural network(CNN), and CNN layered with long short term memory (LSTM) classifiers.

5.3 Experimental setup

The simulation is performed using Python 3.5 on an Intel i5 desktop with 32 GB RAM and 2.71 GHz frequency. The significant packages used are NLTK for NLP tasks, Keras, Theano and Tensorflow for implementation of deep architectures, and Scikit-learn and SciPy for standard machine learning architectures and performance measurements.

For the classification task, the whole dataset is partitioned randomly into two parts. One part is used for training the model and the other is for testing the performance. Three different ratios (80%, 70%, and 60%) of training and test (20%, 30%, and 40%) are prepared for the experiment. Each model is trained and tested with 10 rounds of experiments with the same train-test splits, and the average of 10 readings is depicted in the result section. The random split, tests the consistency of the model performance over varying sets of training and testing data during the multiple iterations.

6 Performance measurement indices

In the present study, various performance measurement indices are used for comparative analysis; they are Precision (P), Recall (R), Accuracy (A), Kappa Score (K), and Receiver Operating Characteristic (ROC) curve.

True Positives is the number of data items predicted to be true, which are actually true in the dataset. *False Positives* is the number of data items predicted to be true, which are actually false in the dataset. *True Negatives* is the number of data items predicted to be false, which are actually false in the dataset. *False Negatives* is the number of data items predicted to be false, which are actually true in the dataset.

Accuracy(A) signifies the fraction of correct classifications out of the total number of data items provided. It represents the ability of the model to correctly identify data items belonging to each of the classes.

Precision(P) represents the fraction of positive data items correctly classified out of the positive data items provided. It shows the ratio of the relevant cases found correctly, out of all the cases that are found to be relevant.

Recall(R) is the fraction of positive data items correctly classified out of the total data items provided. It shows the ratio of the relevant cases found correctly, out of all the cases that are actually relevant in the entire dataset.

Kappa score(K) compares the obtained accuracy with the accuracy of a random system. It controls data items that might have been correctly classified by chance, by measuring how closely the data items classified by the model match the data items labelled as ground truth. A kappa value of 1 denotes perfect match, while a kappa value of 0 denotes no match. Equation 7 gives the formula for Kappa score.

$$Kappa\ Score(K) = \frac{N\ (True\ Positives + True\ Negatives) - X}{N^2 - X} \tag{7}$$

where,

$$X = (True\ Positives + False\ Positives) * (True\ Positives + False\ Negatives) + (True\ Negatives + False\ Negatives) * (True\ Negatives + False\ Positives) \tag{8}$$

Here, N denotes the total number of data items in the dataset.

Receiver Operating Characteristic (ROC) curves are used to provide graphical analysis of the results. ROC curve plots true positive rate and false positive rate for each classifier. The area under the curve (AUC) provides an aggregate measure of the performance, i.e. more the area, better the model.

In addition to the above indexes, we have used the *standard deviation of accuracy* (A_{std}) over the ten rounds of each model in order to analyze the statistical consistency in performance. This analysis demonstrates the variation of the model performances, and a lower value of standard deviation signifies higher stability.

7 Results and discussion

The proposed sentiment analysis model along with the different classifiers are applied on both the datasets. At first, the sentiment analysis for the dataset is performed using the generalized Word2Vec and GloVe embeddings, and the same process is repeated using the modified Word2Vec and GloVe embedding approach. Performances of various classifier-based models on these datasets using both types of embeddings are shown in Tables 1–4.

Table 1 shows the performance of models with different classifiers using the modified Word2Vec embeddings of the IMDb dataset as input, as compared to the generalized Word2Vec embeddings. The results provide better accuracy, precision, recall and kappa score by all classifiers. The MLP, CNN and SVM based models gave the best accuracies for all train-test split ratios using the modified word embeddings. But these models have higher standard deviations, showing some inconsistency over different train-test split sets. The Gradient Boosting and CNN classifiers showed the most consistent results, as seen by the minimum standard deviation values. In all classifiers, the usage of modified word embeddings have shown better performance for the parameters than generalized ones.

Figure 6 shows the ROC curves of the top three classifiers,i.e. MLP, CNN and SVM, that have performed sentiment analysis using modified Word2Vec embeddings on the IMDb dataset in comparison to the generalized Word2Vec embeddings. The ROC curves reiterate similar results as seen in the table. The area under the curve is higher in all cases for the modified Word2Vec embeddings.

Table 2 shows the performance of models with different classifiers by taking the modified GloVe embeddings of the IMDb dataset as input, as compared to the generalized GloVe embeddings. Here too, the results provide better accuracy, precision, recall and kappa score by all classifiers. The CNN, MLP and SVM based models gave the best accuracies for all train-test split ratios using the modified word embeddings. The accuracies of GloVe

Table 1 Performance comparison of generalized and modified Word2Vec embeddings on IMDb dataset using various classifiers

Classifier	Word2Vec embeddings					Modified Word2Vec embeddings				
	P	R	A	K	A_{std}	P	R	A	K	A_{std}
Train-Test split ratio – 80:20										
Random forest	0.81	0.81	0.811	0.624	0.0042	0.83	0.83	0.832	0.665	0.0040
Gaussian naïve bayes	0.73	0.73	0.728	0.493	0.0030	0.76	0.76	0.755	0.511	0.0027
Decision tree	0.70	0.70	0.692	0.408	0.0109	0.72	0.72	0.719	0.439	0.0100
SVM	0.82	0.82	0.819	0.681	0.0083	0.85	0.85	0.854	0.709	0.0077
MLP	0.84	0.84	0.837	0.702	0.0078	0.87	0.87	0.869	0.738	0.0075
Gradient boosting	0.82	0.82	0.817	0.636	0.0022	0.84	0.84	0.838	0.677	0.0021
CNN	0.85	0.85	0.844	0.712	0.0078	0.87	0.87	0.866	0.732	0.0071
CNN-LSTM	0.76	0.75	0.762	0.533	0.0136	0.79	0.79	0.788	0.577	0.0123
Train-Test split ratio – 70:30										
Random forest	0.82	0.82	0.828	0.634	0.0068	0.84	0.84	0.841	0.676	0.0051
Gaussian naïve bayes	0.76	0.76	0.760	0.528	0.0055	0.78	0.78	0.779	0.564	0.0032
Decision tree	0.70	0.70	0.698	0.409	0.0104	0.73	0.72	0.725	0.441	0.0088
SVM	0.84	0.84	0.837	0.695	0.0078	0.86	0.86	0.858	0.728	0.0063
MLP	0.85	0.85	0.852	0.714	0.0096	0.88	0.88	0.880	0.762	0.0087
Gradient boosting	0.82	0.81	0.815	0.707	0.0045	0.84	0.84	0.842	0.740	0.0026
CNN	0.87	0.87	0.869	0.743	0.0065	0.89	0.89	0.890	0.788	0.0050
CNN-LSTM	0.78	0.78	0.780	0.565	0.0104	0.80	0.80	0.798	0.582	0.0096
Train-Test split ratio – 60:40										
Random forest	0.83	0.83	0.833	0.645	0.0077	0.85	0.84	0.848	0.688	0.0052
Gaussian naïve bayes	0.78	0.78	0.780	0.556	0.0040	0.80	0.80	0.802	0.592	0.0022
Decision tree	0.72	0.72	0.719	0.474	0.0089	0.74	0.74	0.738	0.503	0.0078
SVM	0.84	0.84	0.843	0.708	0.0069	0.86	0.86	0.861	0.734	0.0063
MLP	0.85	0.84	0.846	0.692	0.0077	0.87	0.87	0.870	0.743	0.0068
Gradient boosting	0.81	0.81	0.811	0.699	0.0030	0.83	0.83	0.832	0.727	0.0021
CNN	0.88	0.88	0.876	0.762	0.0068	0.90	0.90	0.901	0.808	0.0059
CNN-LSTM	0.78	0.77	0.770	0.558	0.0104	0.81	0.80	0.804	0.597	0.0100

embeddings are slightly lower than Word2Vec embeddings for the IMDb dataset. But the variation shown by the classifiers over different train-test split sets using GloVe embeddings is lower, as seen in the standard deviation column. Hence, GloVe embeddings are more stable than Word2Vec embeddings for most classifiers. The CNN classifier showed the most consistent results, as seen by the minimum standard deviation values. In all classifiers, the usage of modified word embeddings have shown better performance for the parameters than generalized ones.

Figure 7 shows the ROC curves of the top three classifiers,i.e. MLP, CNN and SVM, that have performed sentiment analysis using modified GloVe embeddings on the IMDb dataset in comparison to the generalized GloVe embeddings. The ROC curves graphically provide the results seen in the table. The area under the curve is higher in all cases for the modified Word2Vec embeddings.

Table 2 Performance comparison of generalized and modified GloVe embeddings on IMDb dataset using various classifiers

Classifier	GloVe embeddings					Modified GloVe embeddings				
	P	R	A	K	A_{std}	P	R	A	K	A_{std}
Train-Test split ratio – 80:20										
Random forest	0.76	0.75	0.752	0.528	0.0082	0.78	0.78	0.778	0.557	0.0079
Gaussian naïve bayes	0.70	0.70	0.701	0.431	0.0145	0.73	0.73	0.732	0.464	0.0121
Decision tree	0.63	0.63	0.627	0.289	0.0082	0.65	0.65	0.652	0.304	0.0082
SVM	0.80	0.80	0.791	0.625	0.0066	0.82	0.82	0.820	0.640	0.0065
MLP	0.80	0.79	0.788	0.612	0.0073	0.82	0.82	0.823	0.646	0.0067
Gradient boosting	0.76	0.76	0.755	0.541	0.0105	0.79	0.79	0.788	0.577	0.0100
CNN	0.81	0.80	0.798	0.637	0.0060	0.83	0.83	0.828	0.656	0.0058
CNN-LSTM	0.72	0.72	0.711	0.459	0.0084	0.74	0.74	0.742	0.485	0.0081
Train-Test split ratio – 70:30										
Random forest	0.78	0.78	0.778	0.572	0.0077	0.80	0.80	0.799	0.599	0.0070
Gaussian naïve bayes	0.62	0.63	0.626	0.251	0.0110	0.65	0.65	0.645	0.289	0.0092
Decision tree	0.65	0.65	0.644	0.321	0.0075	0.68	0.68	0.676	0.352	0.0071
SVM	0.79	0.79	0.788	0.779	0.0054	0.82	0.82	0.818	0.636	0.0050
MLP	0.82	0.82	0.820	0.643	0.0068	0.84	0.84	0.838	0.676	0.0065
Gradient boosting	0.78	0.78	0.779	0.588	0.0099	0.80	0.80	0.800	0.601	0.0099
CNN	0.80	0.80	0.798	0.622	0.0050	0.83	0.83	0.827	0.655	0.0048
CNN-LSTM	0.74	0.73	0.732	0.505	0.0078	0.76	0.76	0.764	0.528	0.0075
Train-Test split ratio – 60:40										
Random forest	0.78	0.78	0.777	0.591	0.0083	0.80	0.80	0.803	0.606	0.0075
Gaussian naïve bayes	0.65	0.65	0.646	0.292	0.0122	0.66	0.66	0.655	0.310	0.0102
Decision tree	0.66	0.66	0.657	0.334	0.0075	0.68	0.68	0.678	0.357	0.0071
SVM	0.80	0.80	0.800	0.611	0.0068	0.82	0.82	0.819	0.639	0.0063
MLP	0.81	0.81	0.809	0.652	0.0073	0.84	0.84	0.839	0.678	0.0071
Gradient boosting	0.79	0.78	0.785	0.589	0.0090	0.81	0.81	0.806	0.613	0.0081
CNN	0.80	0.80	0.798	0.622	0.0058	0.82	0.82	0.822	0.645	0.0056
CNN-LSTM	0.76	0.76	0.757	0.538	0.0081	0.79	0.78	0.784	0.569	0.0076

Table 3 shows the performance of models with different classifiers by taking the modified Word2Vec embeddings of the Yelp dataset as input, as compared to the generalized Word2Vec embeddings. Like the IMDb dataset, here also the results provide better accuracy, precision, recall and kappa score by all classifiers. The MLP, CNN, SVM and CNN-LSTM based models gave the best accuracies for all train-test split ratios using the modified word embeddings. The variation shown by these classifiers over different train-test split sets is quite considerable, due to the smaller size of the dataset, as seen in the standard deviation column. More entries in the dataset can help stabilize the variations. The Decision Tree classifier showed the most consistent results, as seen by the minimum standard deviation values, although its accuracy is average. In all classifiers, the usage of modified word embeddings have shown better performance for the parameters than generalized ones.

Table 3 Performance comparison of generalized and modified Word2Vec embeddings on Yelp dataset using various classifiers

Classifier	Word2Vec embeddings					Modified Word2Vec embeddings				
	P	R	A	K	A_{std}	P	R	A	K	A_{std}
Train-Test split ratio – 80:20										
Random forest	0.85	0.85	0.851	0.452	0.0144	0.87	0.87	0.874	0.478	0.0127
Gaussian naïve bayes	0.80	0.64	0.611	0.202	0.0289	0.82	0.65	0.625	0.229	0.0267
Decision tree	0.79	0.79	0.788	0.321	0.0053	0.82	0.82	0.809	0.349	0.0047
SVM	0.86	0.86	0.860	0.519	0.0122	0.88	0.89	0.887	0.547	0.0118
MLP	0.88	0.87	0.868	0.604	0.0156	0.90	0.90	0.902	0.637	0.0154
Gradient boosting	0.85	0.85	0.847	0.509	0.0119	0.87	0.88	0.875	0.538	0.0113
CNN	0.86	0.86	0.862	0.568	0.0171	0.89	0.90	0.897	0.594	0.0164
CNN-LSTM	0.85	0.86	0.857	0.533	0.0228	0.88	0.88	0.882	0.565	0.0219
Train-Test split ratio – 70:30										
Random forest	0.86	0.86	0.858	0.447	0.0138	0.88	0.88	0.876	0.489	0.0115
Gaussian naive bayes	0.79	0.68	0.685	0.283	0.0292	0.82	0.70	0.695	0.321	0.0276
Decision tree	0.80	0.80	0.799	0.412	0.0050	0.82	0.82	0.819	0.421	0.0046
SVM	0.89	0.89	0.892	0.676	0.0156	0.92	0.92	0.917	0.705	0.0121
MLP	0.90	0.90	0.900	0.732	0.0188	0.93	0.92	0.923	0.761	0.0171
Gradient boosting	0.89	0.89	0.887	0.681	0.0101	0.91	0.91	0.911	0.709	0.0097
CNN	0.91	0.91	0.913	0.738	0.0132	0.94	0.93	0.931	0.783	0.0116
CNN-LSTM	0.88	0.88	0.877	0.656	0.0237	0.90	0.90	0.898	0.684	0.0214
Train-Test split ratio – 60:40										
Random forest	0.86	0.86	0.857	0.424	0.0166	0.89	0.88	0.878	0.469	0.0136
Gaussian naïve bayes	0.82	0.71	0.716	0.341	0.0266	0.84	0.74	0.740	0.383	0.0243
Decision tree	0.80	0.79	0.794	0.386	0.0042	0.83	0.82	0.821	0.418	0.0029
SVM	0.91	0.91	0.911	0.708	0.0176	0.93	0.93	0.927	0.735	0.0154
MLP	0.90	0.90	0.898	0.699	0.0118	0.92	0.92	0.919	0.733	0.0096
Gradient boosting	0.88	0.88	0.877	0.624	0.0093	0.90	0.90	0.899	0.656	0.0076
CNN	0.91	0.91	0.914	0.710	0.0122	0.93	0.93	0.931	0.746	0.0118
CNN-LSTM	0.87	0.87	0.869	0.603	0.0236	0.89	0.89	0.890	0.628	0.0202

Figure 8 shows the ROC curves of the top three classifiers,i.e. MLP, CNN and SVM, that have performed sentiment analysis using modified Word2Vec embeddings on the Yelp dataset in comparison to the generalized Word2Vec embeddings. The ROC curves graphically provide the results seen in the table. The area under the curve is higher in all cases for the modified Word2Vec embeddings.

Table 4 shows the performance of with models different classifiers by taking the modified GloVe embeddings of the Yelp dataset as input, as compared to taking the generalized GloVe embeddings. In this case also the results provide better accuracy, precision, recall and kappa score by all classifiers. The CNN, SVM, MLP and Gradient Boosting models gave the best accuracies for all train-test split ratios using the modified word embeddings. The variation shown by these classifiers over different train-test split sets is moderate, as seen in the standard deviation column. It is observed from this that the performance of GloVe vectors is

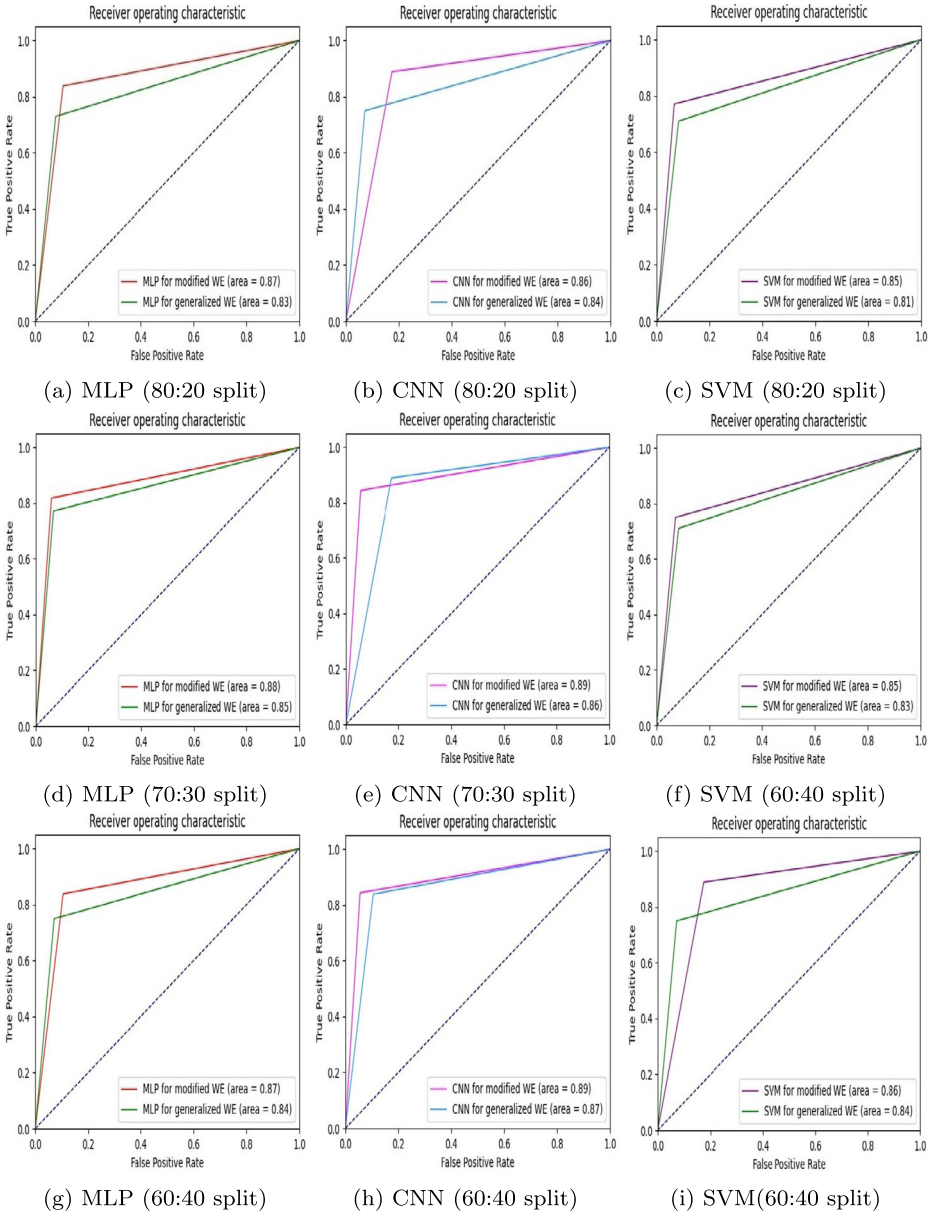
Table 4 Performance comparison of generalized and modified GloVe embeddings on Yelp dataset using various classifiers

Classifier	GloVe embeddings					Modified GloVe embeddings				
	P	R	A	K	A_{std}	P	R	A	K	A_{std}
Train-Test split ratio – 80:20										
Random forest	0.86	0.86	0.859	0.432	0.0075	0.89	0.88	0.878	0.468	0.0072
Gaussian naïve bayes	0.81	0.71	0.710	0.337	0.0340	0.83	0.74	0.735	0.368	0.0300
Decision tree	0.80	0.80	0.801	0.363	0.0152	0.82	0.82	0.812	0.380	0.0134
SVM	0.90	0.90	0.899	0.729	0.0058	0.93	0.93	0.931	0.757	0.0049
MLP	0.90	0.90	0.901	0.703	0.0096	0.92	0.92	0.918	0.728	0.0089
Gradient boosting	0.89	0.88	0.882	0.674	0.0178	0.91	0.91	0.910	0.698	0.0160
CNN	0.91	0.91	0.911	0.773	0.0066	0.94	0.93	0.941	0.793	0.0064
CNN-LSTM	0.85	0.85	0.850	0.538	0.0183	0.87	0.88	0.882	0.562	0.0164
Train-Test split ratio – 70:30										
Random forest	0.83	0.83	0.828	0.316	0.0082	0.85	0.85	0.849	0.341	0.0079
Gaussian naïve bayes	0.75	0.51	0.504	0.158	0.0398	0.79	0.54	0.539	0.161	0.0364
Decision tree	0.76	0.76	0.758	0.292	0.0144	0.79	0.79	0.787	0.315	0.0142
SVM	0.87	0.87	0.871	0.579	0.0046	0.89	0.90	0.895	0.608	0.0040
MLP	0.87	0.87	0.866	0.618	0.0108	0.89	0.89	0.893	0.647	0.0104
Gradient boosting	0.83	0.82	0.828	0.487	0.0192	0.85	0.85	0.854	0.512	0.0181
CNN	0.87	0.87	0.871	0.624	0.0058	0.90	0.90	0.898	0.662	0.0046
CNN-LSTM	0.78	0.79	0.792	0.290	0.0204	0.80	0.82	0.820	0.317	0.0186
Train-Test split ratio – 60:40										
Random forest	0.83	0.83	0.822	0.284	0.0078	0.86	0.85	0.854	0.316	0.0074
Gaussian naïve bayes	0.80	0.52	0.556	0.163	0.0387	0.82	0.57	0.571	0.199	0.0328
Decision tree	0.75	0.75	0.752	0.226	0.0168	0.77	0.77	0.768	0.240	0.0151
SVM	0.86	0.86	0.864	0.559	0.0044	0.89	0.89	0.892	0.578	0.0034
MLP	0.86	0.86	0.857	0.589	0.0088	0.88	0.89	0.886	0.612	0.0076
Gradient boosting	0.84	0.84	0.840	0.474	0.0181	0.86	0.86	0.862	0.509	0.0174
CNN	0.86	0.86	0.859	0.601	0.0057	0.89	0.90	0.898	0.630	0.0040
CNN-LSTM	0.79	0.80	0.801	0.298	0.0157	0.81	0.83	0.832	0.319	0.0139

more stable than Word2Vec vectors. The SVM classifier showed the most consistent results, as seen by the minimum standard deviation value. In all classifiers, the usage of modified word embeddings have shown better performance for the parameters than generalized ones.

Figure 9 shows the ROC curves of the top three classifiers,i.e. MLP, CNN and SVM, that have performed sentiment analysis using modified GloVe embeddings on the Yelp dataset in comparison to the generalized GloVe embeddings. The ROC curves graphically reiterate the results seen in the table. The area under the curve is higher in all cases for the modified Word2Vec embeddings.

Word vectors are representative of linguistic properties of words. They can be applied for most natural language processing tasks, to give acceptable results. By focusing specifically on the sentiment aspect of words, the proposed method tries to make the word embeddings more suitable and sentiment-representative, and hence achieves better results for sentiment analysis.



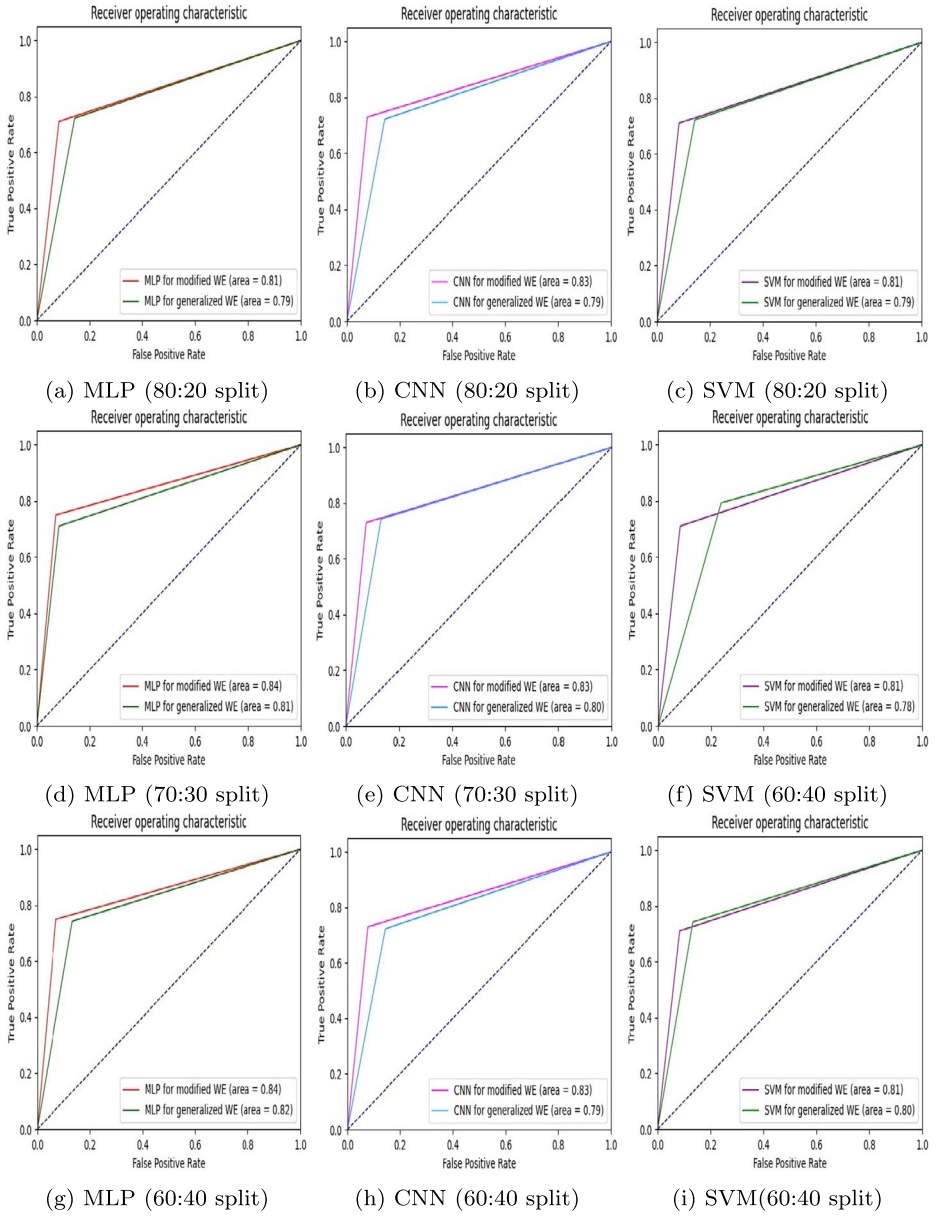
(a) MLP (80:20 split) (b) CNN (80:20 split) (c) SVM (80:20 split)

(d) MLP (70:30 split) (e) CNN (70:30 split) (f) SVM (60:40 split)

(g) MLP (60:40 split) (h) CNN (60:40 split) (i) SVM(60:40 split)

Fig. 6 ROC curve comparison of top 3 classifiers on IMDb dataset using generalized and modified Word2Vec

Some additional comparative analysis shows that the performance of different classifiers is closer to one another on the IMDb dataset than the Yelp dataset. This can be because of the fact that the Yelp dataset is smaller in size, and hence enough training data is not provided. The larger size of the IMDb dataset trains the classifiers well and makes them more accurate. The Word2Vec embeddings perform better on the IMDb dataset, whereas the GloVe embeddings work better on the Yelp dataset. This goes to show that different datasets



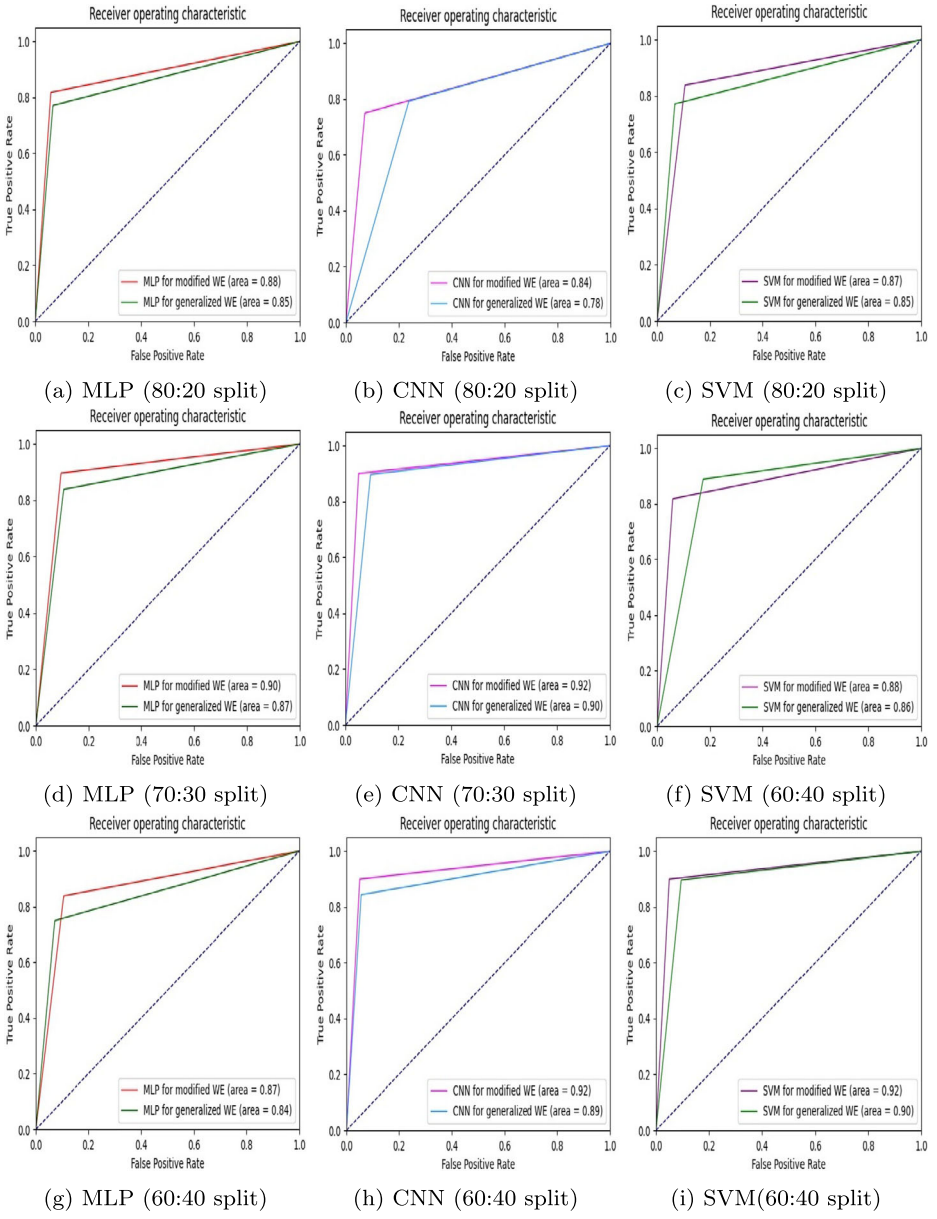
(a) MLP (80:20 split) (b) CNN (80:20 split) (c) SVM (80:20 split)

(d) MLP (70:30 split) (e) CNN (70:30 split) (f) SVM (60:40 split)

(g) MLP (60:40 split) (h) CNN (60:40 split) (i) SVM(60:40 split)

Fig. 7 ROC curve comparison of top 3 classifiers on IMDb dataset using generalized and modified GloVe

are characteristically different, and the suitability of a particular type of embedding can vary over different datasets. So there is no clear superiority of either of the embeddings. An additional observation of our work is the assessment of classifier suitability for sentiment analysis. In all cases, the SVM, MLP and CNN classifiers perform well and have relatively better accuracies. The better performance of these classifiers is because they are suitable for high-dimensional feature spaces. Since we use 300-dimensional word embeddings, SVM



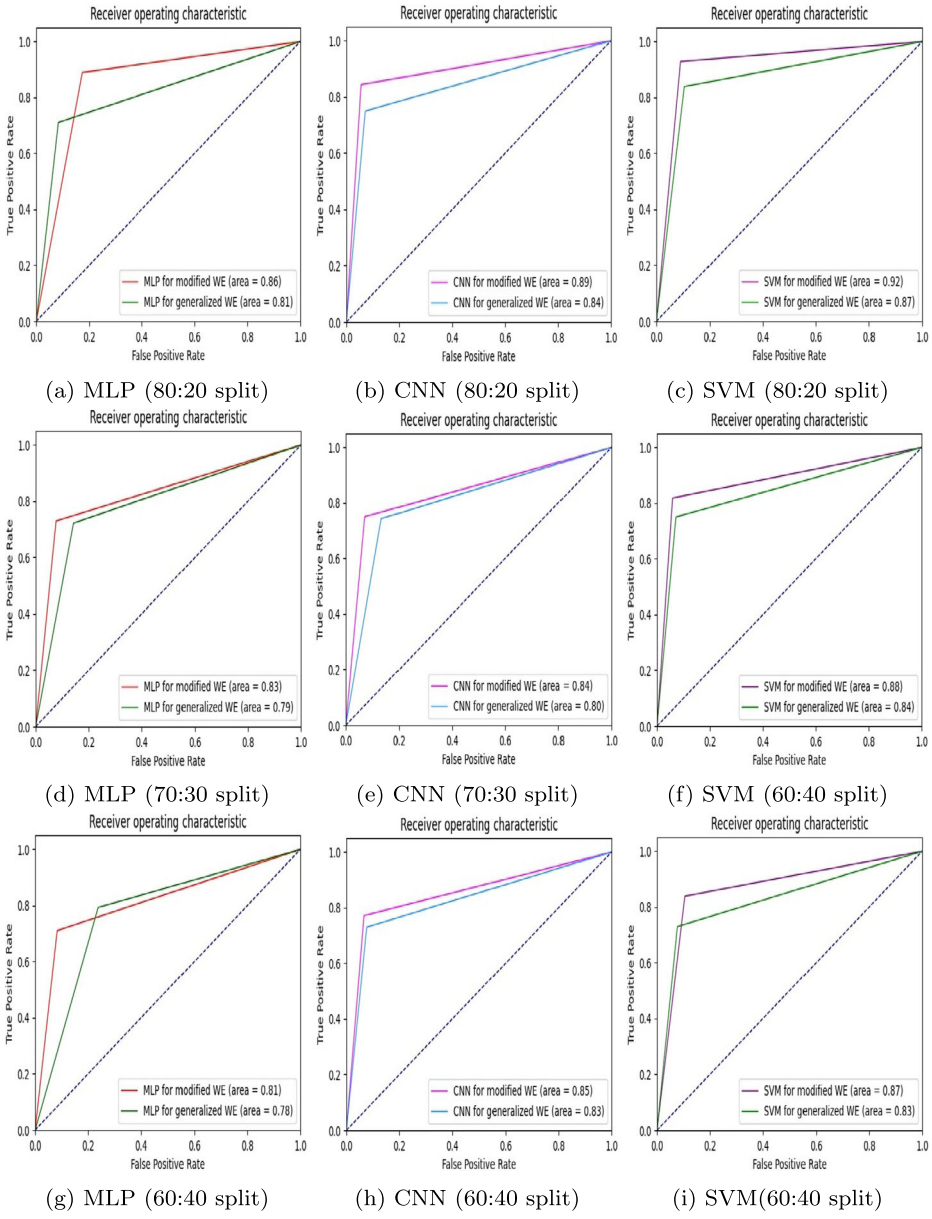
(a) MLP (80:20 split) (b) CNN (80:20 split) (c) SVM (80:20 split)

(d) MLP (70:30 split) (e) CNN (70:30 split) (f) SVM (60:40 split)

(g) MLP (60:40 split) (h) CNN (60:40 split) (i) SVM(60:40 split)

Fig. 8 ROC curve comparison of top 3 classifiers on Yelp dataset using generalized and modified Word2Vec

is able to deal with such high dimensionality using appropriate kernel functions. MLP has higher approximation quality within a single hidden layer, and can efficiently work on various feature combinations. CNN, because of its ability to identify features at higher levels of abstraction, is also effective on text processing, since it can identify word conections and sentence structures. Thus, it is observed that these classifiers are quite suitable for text processing, especially sentiment analysis.



(a) MLP (80:20 split) (b) CNN (80:20 split) (c) SVM (80:20 split)

(d) MLP (70:30 split) (e) CNN (70:30 split) (f) SVM (60:40 split)

(g) MLP (60:40 split) (h) CNN (60:40 split) (i) SVM(60:40 split)

Fig. 9 ROC curve comparison of top 3 classifiers on Yelp dataset using generalized and modified GloVe

8 Conclusion and future work

The proposed method is a supporting step in the automatic analysis of sentiments and ratings, which reduces manual effort and time. The technique captures sentiment at the word level, in correlation to its neighboring and similar words. It improves the performance of sentiment analysis, as opposed to using generalized pre-trained word embeddings. The

method can work on smaller as well as larger datasets and provide a reasonably good performance. It works well on both Word2Vec and GloVe embeddings and enhances their performance, which demonstrates that it is generalized enough to work on different embedding methods. The experiment highlights the suitability of CNN, SVM and MLP classifiers towards text processing in general and sentiment analysis in particular.

This work shows an attempt to achieve more accurate results in sentiment classification, using a mechanism to modify pre-trained word embeddings, namely Word2Vec and GloVe. These embeddings are modified based on the word's sentiment values, which have been referred from an existing sentiment corpus. Words which are sentimentally similar are brought closer to one another, using PSO algorithm. This helps the classifier learn their inter-relationships better, and in turn, provide a higher accuracy. The mechanism is tested on various datasets and classifiers, and improvement in accuracy is observed. Future work can involve speeding up the process of minimizing the distances between word vectors. Besides PSO, other optimization methods can also be used, and tested for faster or better results.

References

1. Aydoğan E, Akcayol MA (2016) A comprehensive survey for sentiment analysis tasks using machine learning techniques. In: 2016 International symposium on INnovations in intelligent systems and applications, INISTA, IEEE, pp 1–7
2. Bradley MM, Lang PJ (1999) Affective norms for english words (anew): Instruction manual and affective ratings. Tech. rep., Technical report C-1, the center for research in psychophysiology
3. Çano E, Morisio M (2019) Word embeddings for sentiment analysis: a comprehensive empirical survey. arXiv:190200753
4. Caschera MC, Ferri F, Grifoni P (2016) Sentiment analysis from textual to multimodal features in digital environments. In: Proceedings of the 8th International Conference on Management of Digital EcoSystems, pp 137–144
5. Chaturvedi I, Cambria E, Welsch RE, Herrera F (2018) Distinguishing between facts and opinions for sentiment analysis: Survey and challenges. Information Fusion 44:65–77
6. Code G (2013) [dataset]. https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing
7. Dragoni M, Petrucci G (2017) A neural word embeddings approach for multi-domain sentiment analysis. IEEE Trans Affect Comput 8(4):457–470
8. D'Urso P, De Giovanni L, Massari R (2020) Smoothed self-organizing map for robust clustering. Inf Sci 512:381–401
9. Eberhart R, Kennedy J (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, Citeseer, vol 4, pp 1942–1948
10. Fu P, Lin Z, Yuan F, Wang W, Meng D (2018) Learning sentiment-specific word embedding via global sentiment representation. In: Thirty-second AAAI conference on artificial intelligence
11. Haddi E, Liu X, Shi Y (2013) The role of text pre-processing in sentiment analysis. Procedia Computer Science 17:26–32
12. Hussein DMEDM (2018) A survey on sentiment analysis challenges. Journal of King Saud University-Engineering Sciences 30(4):330–338
13. Ju S, Li S, Su Y, Zhou G, Hong Y, Li X (2012) Dual word and document seed selection for semi-supervised sentiment classification. In: Proceedings of the 21st ACM international conference on Information and knowledge management, ACM, pp 2295–2298
14. Kaur A, Gupta V (2013) A survey on sentiment analysis and opinion mining techniques. Journal of Emerging Technologies in Web Intelligence 5(4):367–371
15. Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biological Cybernetics 43(1):59–69
16. Kohonen T (1990) The self-organizing map. Proc IEEE 78(9):1464–1480
17. Liu B (2012) Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies 5(1):1–167

18. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1, Association for Computational Linguistics, pp 142–150
19. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv:13013781
20. Ortigosa-Hernández J, JD Rodríguez, Alzate L, Lucania M, Inza I, Lozano JA (2012) Approaching sentiment analysis by using semi-supervised learning of multi-dimensional classifiers. Neurocomputing 92:98–115
21. Pang B, Lee L (2005) Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd annual meeting on association for computational linguistics, Association for Computational Linguistics, pp 115–124
22. Pang B, Lee L, Vaithyanathan S (2002) Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, Association for Computational Linguistics, pp 79–86
23. Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
24. Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/
25. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intelligence 1(1):33–57
26. Rezaeinia SM, Ghodsi A, Rahmani R (2017) Improving the accuracy of pre-trained word embeddings for sentiment analysis. arXiv:171108609
27. Rudkowsky E, Haselmayer M, Wastian M, Jenny M, Emrich Š, Sedlmair M (2018) More than bags of words:, Sentiment analysis with word embeddings. Communication Methods and Measures 12(2-3):140–157
28. Sagnika S, Pattanaik A, Mishra BSP, Meher SK (2020) A review on multi-lingual sentiment analysis by machine learning methods. J Eng Sci Technol Rev 13(2):154–166
29. Sarwan NS (2017) Intuitive understanding of word embeddings: From count vectors to word2vec. https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/
30. Shaikhha H (2017) Github- hammadshaikhha/math-of-machine-learning-course-by-siraj. https://github.com/hammadshaikhha/Math-of-Machine-Learning-Course-by-Siraj
31. Tang D, Wei F, Yang N, Zhou M, Liu T, Qin B (2014) Learning sentiment-specific word embedding for twitter sentiment classification. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp 1555–1565
32. Turney PD (2002) Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics, pp 417–424
33. Wang D, Tan D, Liu L (2018) Particle swarm optimization algorithm: an overview. Soft Comput 22(2):387–408
34. Yang HC, Lee CH, Wu CY (2018) Sentiment discovery of social messages using self-organizing maps. Cognitive Computation 10(6):1152–1166
35. Yang X, Macdonald C, Ounis I (2018) Using word embeddings in twitter election classification. Information Retrieval Journal 21(2-3):183–207
36. Yu LC, Wang J, Lai KR, Zhang X (2017) Refining word embeddings for sentiment analysis. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 534–539
37. Zhang Z, Lan M (2015) Learning sentiment-inherent word embedding for word-level and sentence-level sentiment analysis. In: 2015 International Conference on Asian Language Processing (IALP), IEEE, pp 94–97