**International Journal of Web-Based Learning and Teaching Technologies**
Volume 16 • Issue 6

# Sentiment Weighted Word Embedding for Big Text Data

Jenish Dhanani, National Institute of Technology, Surat, India
Rupa Mehta, National Institute of Technology, Surat, India
Dipti Rana, National Institute of Technology, Surat, India
https://orcid.org/0000-0002-5058-1355

## ABSTRACT

Sentiment analysis is the practice of eliciting a sentiment orientation of people's opinions (i.e., positive, negative, and neutral) toward a specific entity. Word embedding techniques like Word2vec are effective approaches to encode text data into real-valued semantic feature vectors. However, they fail to preserve sentiment information that results in performance deterioration for sentiment analysis. Additionally, big-sized textual data consisting of large vocabulary and its associated feature vectors demand huge memory and computing power. To overcome these challenges, this research proposed a MapReduce-based sentiment-weighted Word2Vec (MSW2V) that learns the sentiment and semantic feature vectors using sentiment dictionary and big textual data in a distributed MapReduce environment, where the memory and computing power of multiple computing nodes are integrated to accomplish the huge resource demand. Experimental results demonstrate the outperforming performance of the MSW2V compared to the existing distributed and non-distributed approaches.

## KEYWORDS

Big Data, MapReduce, Sentiment Analysis, Sentiment Embedding, Word2vec

## INTRODUCTION

With the advancement of Internet technologies, platforms like Social Media, E-Commerce and Movie Streaming Services have directly reached the millions of individuals. Over such platforms, people express and share their emotions, observations and opinions through a piece of text for topics, products, services etc. Sentiment analysis (El Alaoui et al., 2018; Fang & Zhan, 2015; Liu, 2012; Medhat, Hassan, & Korashy, 2014; Pang, Lee, & others, 2008; Pang, Lee, & Vaithyanathan, 2002; Pouransari & Ghili, 2014; Ravi & Ravi, 2015) is performed to elicit a sentiment orientation (i.e. positive and negative) of shared textual information, which can enhance decision making of Governments, Product designers, Political organizations, Marketing organizations, etc. Thus, there is a strong requisite for efficient sentiment analysis approaches. In sentiment analysis, Machine Learning (ML) algorithms have been extensively exploited due to their excellent capability to gain admirable performance (Dhanani, Mehta, Rana, & Tidke, 2018; Pang et al., 2002; Parikh, Palusa, Kasthuri, Mehta, & Rana,

DOI: 10.4018/IJWLTT.20211101.oa2

2018; Xia, Wang, Hu, Li, & Zong, 2013; Yin, Wang, & Zheng, 2012; Zhang, Xu, Su, & Xu, 2015), where efficient feature extraction is the vital demand.

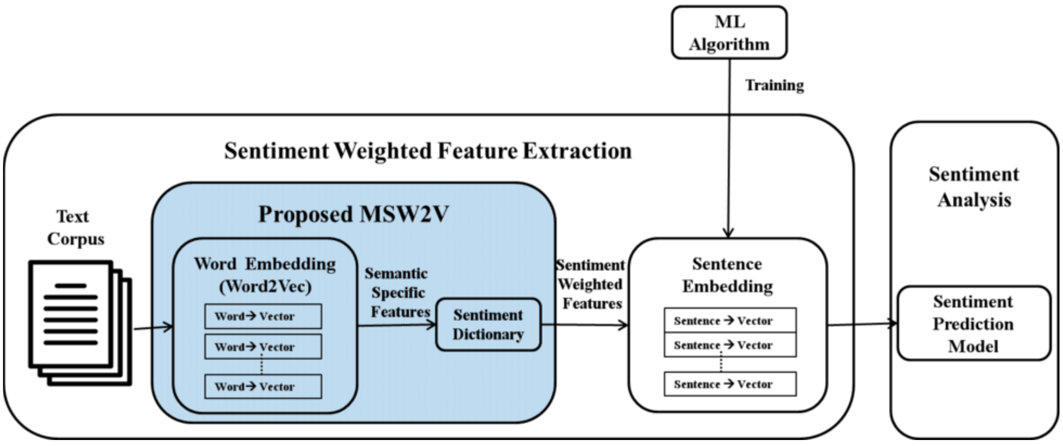
Word embedding techniques extract textual features by transforming raw words into real-valued vectors. Word2vec is a profound word embedding technique which learns the deep and implicit semantic information among the words vectors (also called as feature vectors, Word2vec embedding or word embedding) (Mikolov, Chen, Corrado, & Dean, 2013a; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013b). However, Word2vec fails to encode sufficient sentiment information into the feature vectors (Dhanani et al., 2018; Parikh et al., 2018; Tang et al., 2016, 2014; Yu, Wang, Lai, & Zhang, 2017). As a consequence, semantically similar words like "good" and "bad" are placed closer to each other, even though the sentiment orientations of these words are opposite, such as "good" is sentimentally positive and "bad" is sentimentally negative words. Hence, only semantic specific feature vectors could outcomes the declination in performance. In addition, real-life applications yield big sized textual data consisting of large vocabulary (i.e. unique words in the Text Corpus) (Dhanani et al., 2018; Ordentlich et al., 2016). Word2vec possesses scalability issues due to in-memory computation and accommodation of large vocabulary and its associated vectors (Dhanani et al., 2018; Ordentlich et al., 2016). For such big sized textual data, Word2vec demands huge memory and computing capability to achieve sufficient learning latency.

Many recent works attempted to solve the scalability issues by implementing Word2vec in a distributed environment (Apache Spark based Word2vec, n.d.; Dhanani et al., 2018; Ji, Satish, Li, & Dubey, 2016; Ordentlich et al., 2016). However, they are limited to learn the semantic specific Word2vec feature vectors. In contrast, several recent studies have focused on encoding sentiment information into the Word2vec feature vectors using prior sentiment knowledge (Parikh et al., 2018; Rezaeinia, Ghodsi, & Rahmani, 2017; Yu et al., 2017; Zhang et al., 2015). However, learning sentiment specific Word2vec feature vectors (i.e. which preserves both semantic and sentiment information) is computationally expensive for big sized textual data consisting of a large vocabulary. Existing sentiment analysis approaches learn sentiment and semantic specific feature vectors for big sized textual data, which possess the scalability issue (Dhanani et al., 2018; Parikh et al., 2018). To overcome these challenges, this research proposes a novel sentiment weighted word embedding approach using sentiment dictionary and distributed MapReduce environment.

The proposed MapReduce based Sentiment weighted Word2Vec (MSW2V) approach aimed to learn the feature vectors weighted by sentiment dictionary for big sized textual data. This research is centered on MapReduce, a prominent distributed programming model that splits big sized data to store and process across multiple computing nodes with high fault tolerance and reliability (Apache Hadoop, n.d.; Dean & Ghemawat, 2008). Sentiment dictionary contains the prior sentiment knowledge i.e. Sentiment Polarity Scores (SPSs) for all respective words. The proposed MSW2V learns the local sentiment weighted feature vector by assigning the SPSs to the local Word2vec embedding on multiple computing nodes of Map Phase. It results in a clear bifurcation of semantically similar words based on sentiment values (i.e. "good" and "bad"). The local sentiment feature vectors are integrated into the Reduce Phase.

Figure 1 shows the MSW2V based sentiment analysis, which is primarily divided into two sequential stages. 1) Sentiment weighted feature extraction: The proposed MSW2V learns sentiment specific Word2vec vectors which contain words and their respective vectors. The weighted feature vectors for the user text instances (i.e. sentences, reviews, tweets, comments, etc.) are generated by aggregating individual word vector of the relative text (i.e. Sentence Embedding) (Parikh et al., 2018). 2) Sentiment analysis using sentiment weighted features: A sentiment prediction model is trained by the supervised ML algorithm based on the learnt feature vectors. This research focuses on the Support Vector Machine (SVM) to learn the sentiment prediction model as it has demonstrated the outperforming results compared to the other state of the art techniques (Dhanani et al., 2018; Parikh et al., 2018).

Figure 1. ML based Sentiment Analysis Using Proposed MSW2V



To evaluate the effectiveness of the proposed MSW2V, this research performs an empirical analysis on the Large Movie Review Dataset (Large Movie Review Dataset, n.d.; Maas et al., 2011). Experimental results showcase that the proposed MSW2V outperforms the existing distributed MRW2V (Dhanani et al., 2018) and non-distributed Sentiment Polarized Word2vec (SPW2V) (Parikh et al., 2018) in terms of accuracy performance measures.

The article is organized as follows: Sections *Background and Related Work* present background and recent literature related to semantic and sentiment specific feature vectorization techniques along with distributed Word2vec approaches. Section *Proposed MSW2V* discusses the in-depth architecture of the proposed approach. The empirical results and analysis are presented in Section *Experimentation*. Finally, the research paper is concluded along with the future aspects and applicability in Section *Conclusion and Future Scope*.

## BACKGROUND

There are two key frameworks, namely Word2vec and MapReduce have significant involvement in the proposed approach, which are abridged in this section.

Word2vec (Mikolov et al., 2013a, 2013b) transforms the raw text (i.e. words) into real-valued semantic vectors using shallow Neural Network (NN). Contextual information is used to preserve semantic information among the word vectors (Harris, 1954). As a result, vectors of semantically similar words are embedded in close proximity in a vector space e.g. word vectors of electronic devices (i.e. mobile, laptop, tablet) are placed near to each other as compared to word vectors of fashion accessories (i.e. shoes, shirts). The underlying reason is that words from electronic devices and fashion accessories are different in terms of the context.

There are two Word2vec learning architectures, namely Continuous Bag of Word (CBOW) and Skip Gram (SG). In shallow NN training, the CBOW architecture puts the current word (i.e. target word) on the input layer, and company (i.e. context) words of the current word on the output layer to predict the current word based on the company words. In contrast, the SG architecture predicts the company words given the current word by keeping the context words on the input layer and current word on the output layer of the shallow NN. Both training architectures are mathematically described by Miklove et al. (Mikolov et al., 2013a, 2013b) and Rong et al. (Rong, 2014).

MapReduce is a Hadoop (Apache Hadoop, n.d.) based distributed processing tool which provides a scalable solution for managing big sized data (Dean & Ghemawat, 2008). MapReduce offers extensive scalability by accommodating a multi-nodes cluster which can be made-up from tens, hundreds, or even thousands of computing nodes. There are two main successive functions, namely Map (Mapper) and Reduce (Reducer) where, inputs and outputs are in the form of (Key, Value) pairs. The Mapper is responsible for inputting (Key, Value) pairs from the Data Block followed by performing the user-defined logic over the multiple computing nodes. Mappers produce intermediate outputs consisting (Key, Value) pairs which are logically partitioned by Key. A number of partitions is equivalent to the number of Reducers (i.e. defined by the user), which can allow parallel processing of partitions over multiple Reducers. Before starting Reducers, MapReduce internally performs *Shuffling* to transfer relevant (Key, Value) pairs from Mappers to Reducers, and *Sorting* to aggregate the Values by Key. So, the input to the Reducer is in the form of (Key, List <Value>) pairs on which the user-defined logic is applied. Finally, Reducers produce the output in the form of (Key, Value) pairs.

## RELATED WORKS

This section briefly introduces the previous work on two interrelated research domains viz. word embedding and distributed variants of Word2vec.

Millions of users share and express their opinions and views over the Internet platforms like Movie Streaming Services (i.e. IMDB, Netflix, Amazon Prime), Social Media (i.e. Facebook, YouTube) and E-Commerce (i.e. Amazon, Flipkart), etc. It has resulted in the evolution of big sized textual data which is loosely structured. Individuals, governments and private organizations extract and analyses the sentiments of the people for effective and advantageous decision making (El Alaoui et al., 2018; B. Liu, 2012; Ravi & Ravi, 2015). Sentiment analysis is the practice of categorizing and analyzing people's views and opinions into different polarity classes, like positive and negative. Ravi et al. (Ravi & Ravi, 2015) and Medhat et al. (Medhat et al., 2014) have presented a very extensive and in-depth literature review on the topic of sentiment analysis. It is observed that ML algorithms have been widely adopted in sentiment analysis due to their excellent learning capability which resulted in the admirable performance (Dhanani et al., 2018; Pang et al., 2002; Parikh et al., 2018; Xia et al., 2013; Yin et al., 2012; Zhang et al., 2015). For fruitful ML based sentiment analysis, semantic and sentiment information are the primary demand to be encoded in extracted features.

Word embedding techniques transform textual data into numerical feature vectors, through which ML algorithms can be trained for sentiment analysis. One hot encoding (Li & Yang, 2018) and Bag of Word (Goldberg, 2017) are rudimentary embedding approaches possess two major limitations: 1) Loss of the semantic and word order information 2) For big sized data, they encode the high dimensional vectors. Many research works have attempted to preserve the semantics using statistical techniques viz. Latent Semantic Analysis (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) and Latent Dirichlet Allocation (Blei, Ng, & Jordan, 2003). Miklove et al. (Mikolov et al., 2013a, 2013b) proposed Word2vec which trains a NN using contextual information for the effectual encoding of semantic information in word vectors. For big sized data, Word2vec is efficient in terms of computation cost and linear regularity preservation when compared to Latent Semantic Analysis and Latent Dirichlet Allocation (Mikolov et al., 2013a).

Dhanani et al. (Dhanani et al., 2018) performed sentiment analysis using semantic feature vectors which were learned by Word2vec. However, it failed to capture sufficient sentiment information. To prevent this limitation, many recent studies have focused on encoding sentiment and semantic information in the feature vectors (Jing et al., 2018; Parikh et al., 2018; Rezaeinia et al., 2017; Yu et al., 2017; Zhang et al., 2015). Zhang et al. (Zhang et al., 2015) used ML based sentiment analysis, where sentiment specific features are extracted with the help of Word2vec, lexicons and part-of-speech. Rezaeinia et al. (Rezaeinia et al., 2017) exploited pre-built semantic word embedding (i.e. Word2vec embedding of Google News corpus) along with lexicons to preserve sentiment information. Yu et al. (Yu et al., 2017) refine the pre-trained word embedding using prior sentiment knowledge (i.e. sentiment polarity score) to preserve the semantic and sentiment relationships among words vectors. A conceptually similar approach has been proposed by Parikh et al. (Parikh et al., 2018), the approach assigns prior sentiment knowledge to Word2vec feature vectors for preserving sentiment information. Jing et al. (Jing et al., 2018) extracted attribute-specific features by adopting the Word2vec model and clustering for online reviews.

Traditional Word2vec is inefficient for big sized textual data due to the involvement of numerous mathematical operations on in-memory large vocabulary vectors. It demands huge computing resources which cannot be accommodated by commodity hardware. Recently, various distributed frameworks have been adopted to learn the Word2vec embedding over multiple computing nodes which can justify the huge resource demand (Apache Spark based Word2vec, n.d.; Dhanani et al., 2018; Ji et al., 2016; Ordentlich et al., 2016).

Ordentich et al. (Ordentlich et al., 2016) perform a column-wise distribution to learn distributed word embedding using the parameter server method. However, the column distribution limits the usability of the number of computing node (Stergiou, Straznickas, Wu, & Tsioutsiouliklis, 2017). Ji et al. (Ji et al., 2016) proposed the distributed Word2vec to learn the local word embedding from the individual computing node of a cluster. During learning, all computing nodes perform periodic synchronization to update the learnt word embedding. Similarly, Spark's MLLib (Apache Spark based Word2vec, n.d.) proposed the distributed variants of Word2vec using Master-Slave architecture. Each Slave node learns the local word embedding from the local Data Block. For each iteration, the Master node generates global word embedding by integrating local word embeddings which will be used by Slave nodes to update their local word embedding. This approach enlarges the processing capability due to distributed and in-memory processing but, demands vast memory for each computing node to preserve the complete embedding. Dhanani et al. (Dhanani et al., 2018) proposed MapReduce based Word2Vec (MRW2V) which learns the local word embedding with desired iterations to prevent the multiple updates. So, computing nodes requires the only memory to preserve the local word embedding instead of complete word embedding.

In sentiment analysis, previous research works have attempted to encode sentiment information along with Word2vec embedding. However, these approaches are insufficient to administrate big sized textual data. In contrast, there were many distributed Word2vec approaches presented in literature which can efficiently handle big sized textual data. But, they fail to preserve sufficient sentiment information. This research gap has strongly motivated our research to develop an efficient and effective distributed framework for learning sentiment and semantic word vectors. This paper presents a novel MSW2V approach to encode the sentiment information along with semantic feature vectors using sentiment dictionary for big sized textual data. The proposed approach is described in the next section.

## PROPOSED MAPREDUCE BASED SENTIMENT WEIGHTED WORD2VEC

The proposed MSW2V distributes data across a cluster of computing nodes in the form of small Data Blocks, for parallel access. The proposed approach is logically divided into two successive phases called *Map Phase* and *Reduce Phase*. In the Map Phase, each computing node parallelly learns the local word vectors (i.e. vocabulary and their respective vectors) from the available Data Block using Word2vec. Successively, prior sentimental knowledge (i.e. sentiment dictionary) is assigned to the local word vectors for learning the local sentiment weighted word vectors. In the Reduce Phase, global sentiment weighted word vectors are generated by integrating all the local sentiment weighted word vectors. The following sub-section describes the distributed learning of the proposed MSW2V.

### Distributed Learning in MSW2V

The architecture of the MSW2V is illustrated in Figure 2. A Text Corpus is a set of sequential instances (i.e. sentences, reviews, tweets, comments etc.) where, each instance is made of consecutive words (W). Let us assume that the Text Corpus is distributed to R blocks which are stored across multiple computing nodes. Sentiment dictionary is also uploaded on each computing node which contains

SPSs for vocabulary words. The proposed approach is highly flexible to use any readily available sentiment dictionary (or lexicon dictionary) that comprises of SPSs for given words.

This research uses the Expected Rating approach (Marneffe, Manning, & Potts, 2010; Potts, 2010) to take advantage of domain-specific SPS ranging from *-4.5* to *+4.5*. SPSs of sentimentally negative words are real negative values and sentimentally positive words are real positive values. The negative value of SPS indicates the negative polarity strength of words e.g. the less negative word "bad" has SPS of *-1.47* while the highly negative word "aweful" has SPS of *-2.38*. Similarly, the more positive value of SPS indicates the more positive polarity strength e.g. positive word "awesome" with SPS of *+1.68* is highly positive compared to the word "great" with SPS of *+1.10*. Once each computing node is ready with the necessary data, the MSW2V will sequentially start the Map Phase and Reduce Phase as described below.

## Map Phase

The Map Phase initiates the R Mappers to process each Data Block on multiple computing nodes. On each Mapper, Word2vec learns the local word vectors from the locally available Data Block. The local word vectors consist of vocabulary words and their associated vectors, as shown in Equations (1) and (2):

$$Vocab_R = \left\{W_{1,R}, W_{2,R}, \ldots W_{i,R}, W_{n,R}\right\} \ and \ \left|Vocab\right| = n_R \tag{1}$$

$$Vector_R = \left\{Vec_{1,R}, Vec_{2,R}, \ldots Vec_{i,R}, Vec_{n,R}\right\} \tag{2}$$

Here, $Vocab_R$ indicates the vocabulary having all the unique words $\left(W_{i,R}\right)$ from $R^{th}$ Data Block, $n_R$ indicates the size of vocabulary $Vocab_R$ and $Vector_R$ is the set of corresponding Vectors $\left(Vec_{i,R}\right)$ of word $\left(W_{i,R}\right)$ from $R^{th}$ Data Block. Successively, each Mapper generates the new sentiment weighted Vector $\left(SentVec_{i,R}\right)$ by assigning the Sentiment Polarity Score $\left(SPS_{i,R}\right)$ of the word $\left(W_{i,R}\right)$ to the respective Vector $\left(Vec_{i,R}\right)$ using sentiment dictionary, as shown in equation (3). As a result, semantically similar but sentimentally opposite vectors of the words (i.e. "good" and "bad") are bifurcated in opposite directions:

$$SentVec_{i,R} = Vec_{i,R} \times SPS_{i,R} \tag{3}$$

Finally, each Mapper outputs the local sentiment weighted word vectors as shown in Equations (4) and (5):

$$Vocab_R = \left\{W_{1,R}, W_{2,R}, \ldots W_{i,R}, W_{n,R}\right\} \ and \ \left|Vocab\right| = n_R \tag{4}$$

$$SentVector_R = \left\{SentVec_{1,R}, SentVec_{2,R}, \ldots SentVec_{i,R}, SentVec_{n,R}\right\} \tag{5}$$

**Algorithm 1. Map Phase of MSW2V**

```
Input: $(Key, Value)$ pairs, $Key$ is the starting offset byte of the Instances, $Value$ is the corresponding Instances ($Ins$)

Output: $(Key_i, Value_i)$ pairs, $Key_i$ is the local vocabulary Word $(W_i)$, $Value_i$ is the corresponding Vector $(SentVec_i)$ of the Word $(W_i)$

1:  for each $Ins_i$ in Data Block ($DB_R$) do
2:      Map Phase Input (Key, Value)
3:      $List \leftarrow List \cdot Append(Tokenize(Ins_i))$
4:  end
5:  WordEmbedding $\leftarrow$ Word2Vec (List)
6:  for each $word(W_i)$ in $WordEmbedding$ do
7:      $Key_i \leftarrow W_i$
8:      $Value_i \leftarrow Vec_i \times SPS_i$
9       Map Phase Output $(Key_i, Value_i)$
10: end
```

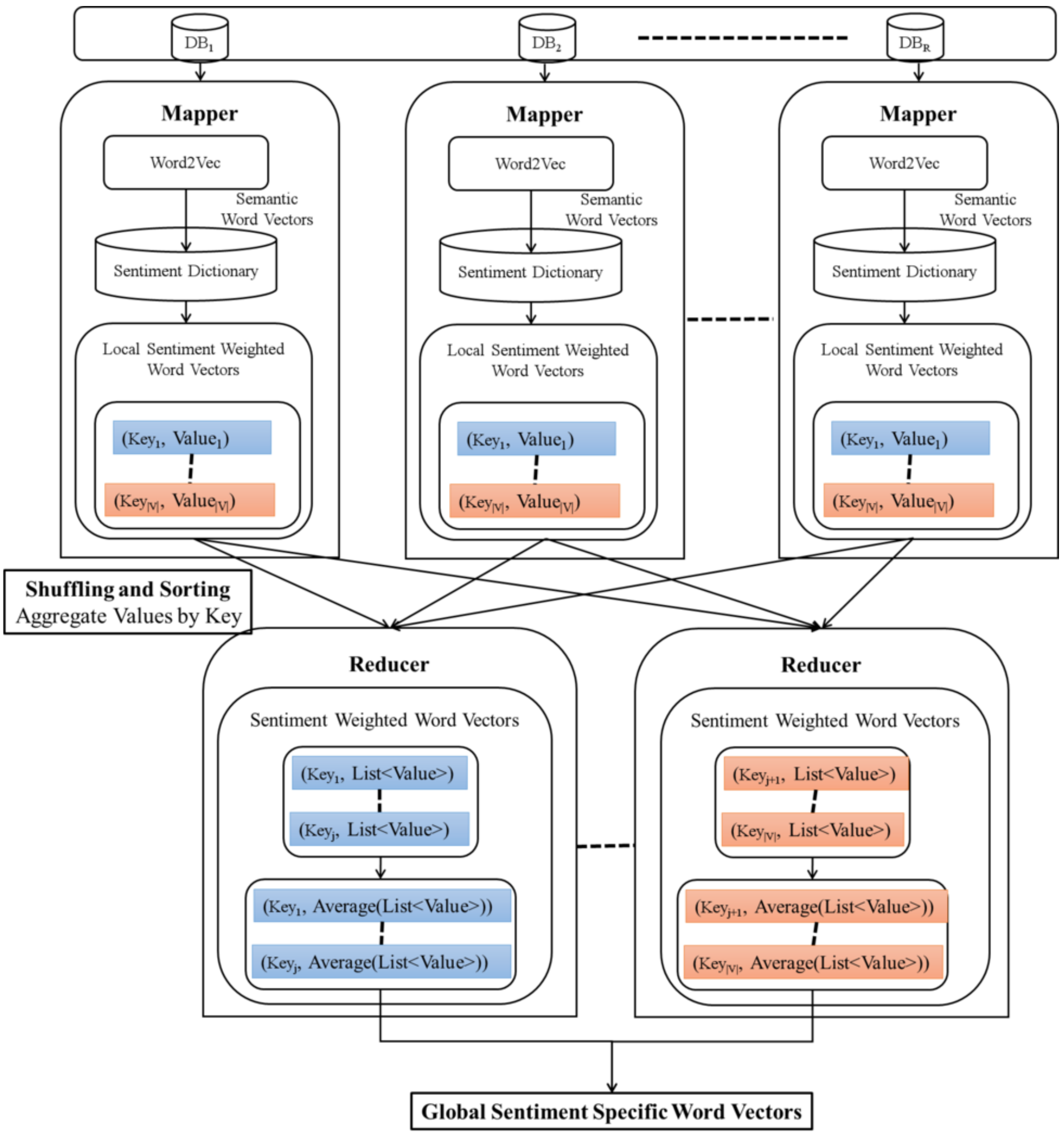
Here, $Vocab_R$ indicates the vocabulary, $n_R$ indicates the vocabulary size and $SentVector_R$ is the set of sentiment weighted Vector ($SentVec_{i,R}$) of word $(W_{i,R})$ from $R^{th}$ block. The process is described in the Algorithm 1, where each Mapper fetches the input in the form of $(Key, Value)$ pairs from the Data Block, as follows:

(**Key:** offset, **Value:** Instances from $R^{th}$ Block) pairs

$List$ accumulates all input $(Key, Value)$ pairs as shown in Line No: 1 to 4 of Algorithm 1. Next, Mapper builds a local word vectors by applying Word2vec to the $List$, as shown Line No: 5 (i.e. Equations (1) and (2)). It generates a sentiment weighted word embedding by assigning the SPSs to the respective vectors as shown Line No: 6 to 10 (i.e. Equation (3)). Finally, the intermediate output is represented in the form of $(Key, Value)$ pairs where Key is the local vocabulary Word $(W_i)$ and Value is respective sentiment weighted vector ($SentVec_i$), as follows:

(**Key:** $Word(W_i)$, **Value:** $SentVec_i$) pairs

Figure 2. Architecture of MSW2V


DB$_1$
DB$_2$
DB$_R$
Mapper
Word2Vec
Semantic Word Vectors
Sentiment Dictionary
Local Sentiment Weighted Word Vectors
(Key$_1$, Value$_1$)
(Key$_{|V|}$, Value$_{|V|}$)
Shuffling and Sorting
Aggregate Values by Key
Reducer
Sentiment Weighted Word Vectors
(Key$_1$, List<Value>)
(Key$_j$, List<Value>)
(Key$_1$, Average(List<Value>))
(Key$_j$, Average(List<Value>))
(Key$_{j+1}$, List<Value>)
(Key$_{|V|}$, List<Value>)
(Key$_{j+1}$, Average(List<Value>))
(Key$_{|V|}$, Average(List<Value>))
Global Sentiment Specific Word Vectors

## Reduce Phase

Local sentiment weighted word vectors need to be synchronized for building a global sentiment weighted word vectors. Reduce Phase initiates multiple Reducers to accumulate the relevant words and their respective vectors from all local sentiment weighted word vectors (Intermediate outputs of Mappers) which differ in vocabulary words. Reduce Phase performs a union operation on local vocabularies as shown in Equation (6). Similarly, the respective vectors of words, which are present in the $k$ different local sentiment weighted word vectors, are averaged as shown in Equations (7) and (8):

**Algorithm 2. Reduce Phase of MSW2V**

Input: $\left(Key_i, ListValue_{i,1}, Value_{i,2}, \ldots, Value_{i,k}\right)$ pairs, $Key_i$ is the vocabulary Word $\left(W_i\right)$, $Value_{i,k}$ is the respective sentiment weighted Vector $\left(SentVec_i\right)$ of Word $\left(W_i\right)$ from $k^{th}$ Mapper

Output: $\left(Key_j, Value_j\right)$ pairs, $Key_j$ is the Word $\left(W_j\right)$ of the global vocabulary, $Value_j$ is the respective sentiment weighted Vector $\left(SentVec_j\right)$ of Word $\left(W_j\right)$

```
1:    Reduce Phase Input $\left(Key_i, List\left\langle Value_{i,1}, Value_{i,2}, \ldots, Value_{i,k}\right\rangle\right)$
2:    for each $Key_i$ in ReducerPhaseInput do
3:        $Key_j \leftarrow Key_i$
4:        $SentVec_j \leftarrow 0$
5:        Count ⟵ 0
6:        for each $Value_{i,k}$ in $ListValue_{i,1}, Value_{i,2}, \ldots, Value_{i,k}$ do
7:            $SentVec_j \leftarrow SentVec_j + Value_{i,k}$
8:            Count ⟵ Count + 1
9         end
10:       $Value_j \leftarrow \left(SentVec_i \div Count\right)$
11:       Reduce Phase Output $\left(Key_i, Value_j\right)$
12    end
```

$$Vocab_{global} = \bigcup_{i=1}^{R} Vocab_i \tag{6}$$

$$SentVec_{global,i} = \frac{\sum_{j=1}^{R} SentVec_{i,j}}{K_i} \tag{7}$$

$$SentVector_{global} = \left\{ SentVec_{global,1}, SentVec_{global,2}, ...SentVec_{global,n} \right\} \quad (8)$$

Here, $k_i$ indicates the number local sentiment weighted word vectors (i.e. Data Blocks) in which Word $\left(W_i\right)$ is present. Reduce Phase results the global sentiment weighted word vectors ($Vocab_{global}$ and $SentVector_{global}$).

The process of Reduce Phase is described in the Algorithm 2. For parallel computation, the local word vectors are logically partitioned by word (i.e. Key). Each partition consists of fixed set of word vectors which are processed by a particular Reducer. Here, MapReduce internally perform *Shuffling* and *Sorting*, where Shuffling is responsible for transferring the relevant $\left(Key, Value\right)$ pairs from Mappers to Reducers. Successively, Sorting is also performed to aggregate Vectors (which are collected from multiple Mappers) by words (i.e. Aggregate the Values by Key). So, the input to the Reducers is in the form of $\left(Key, List\left\langle Value\right\rangle\right)$ pairs as shown in Line No: 1 of Algorithm 2, as follows:

(**Key:** $Word\left(W_i\right)$, **List <Values>:** List $< SentVec_{i,1}, SentVec_{i,2}, ...SentVec_{i,k} >$) pairs

Reducers perform an element wise vector average on all $Values$ in the List (i.e. $List\left\langle Value\right\rangle$) for the same $Key$ (i.e. $Word$), as shown in Line No: 2 to 12 of Algorithm 2 (i.e. Equations (7)). Finally, Reducer outputs the $\left(Key, Value\right)$ pairs where $Key$ is the global word vectors $Word\left(W_j\right)$ and $Value$ is the respective global sentiment weighted $Vector\left(SentVec_j\right)$, are as follows:

(**Key:** $Word\left(W_j\right)$, **Value:** $SentVec_j$) pairs

## EXPERIMENTATION

In order to evaluate the proposed approach, experimentation is performed in two stages, as shown in Figure 1. Sentiment weighted feature vectors are learned using the proposed approach, from which sentence vectors are also prepared. The proposed MSW2V was implemented with the help of Gensim (for Word2vec) (GensimWord2vec Tool, n.d.) in a pseudo-distributed mode of Hadoop (for MapReduce) (Apache Hadoop, n.d.) consisting 4 Mappers and 1 Reducer. It is compared to the existing distributed approach MRW2V (Dhanani et al., 2018) and the non-distributed approach SPW2V (Parikh et al., 2018), for CBOW and SG architecture. The MRW2V was also implemented with the above experimental setup. While, SPW2V learned the feature vectors using Gensim in standalone mode.

In the second stage, sentence vectors are utilized for training the sentiment prediction model using ML technique. In several previous studies, SVM has demonstrated an admirable performance compared to the other state of the art classification techniques (Dhanani et al., 2018; Pang et al., 2002; Parikh et al., 2018; Zhang et al., 2015). Moreover, a number of experiments have also been conducted with other classification techniques (i.e. NN, Logistic Regression, Decision Tree, Random Forest and Linear SVM (LSVM)), in which LSVM has shown the most favorable performance. So, this research has adopted the LSVM (from Scikit-learn (Scikit-learn, n.d.)) for training the sentiment prediction model. In the experiments, the default parameter setting has been utilized for Gensim and Scikit-learn.

### Data Set and Performance Measure

This research utilized the pre-processed variant (Pre-processed Large Movie Review Dataset, n.d.) of the popular Large Movie Review Data Set (Large Movie Review Dataset, n.d.; Maas et al., 2011) which contains 25000 positive, 25000 negative and 50000 unlabeled reviews for a set of movies. A review is considered to be positive with a rating of more than six, and negative with a rating of less than five. To build the sentiment prediction model, training is performed over 12500 positive and 12500 negative reviews. The prediction model is evaluated with the rest 12500 positive and 12500 negative reviews. The Data Set is balanced so this research focuses on Prediction Accuracy of each model as the performance measure.

### Experimental Parameters

Vector size (V) and window size (W) are two key parameters in the Word2vec with a significant impact on performance as well as on space and time complexity (Ordentlich et al., 2016). Vector size represents the dimensionality of the feature vectors. Smaller vector sizes may not capture significant semantics, while larger may not capture any additional adequate information but severely increases the space and time complexity. Window size represents the range of co-occurring words for the specific target word. Smaller window sizes may learn the rigid feature vectors capturing the word focused information. In contrast, larger window size may learn the domain-specific word embedding capturing the topical information. Experiments are conducted for vector size V = 200 to 1000 with the interval of 200 and window size W = 3 to 15 with an interval of 4, to obtain the optimum values for vector and window size.

### Result Analysis

For the sentiment analysis, models of the proposed MSW2V, existing distributed MRW2V and non-distributed SPW2V are learned using CBOW and SG architectures with different vector and window size. The following subsection demonstrates the comparative performance analysis for the proposed MSW2V and MRW2V followed by MSW2V and SPW2V.

## MSW2V VS. MRW2V

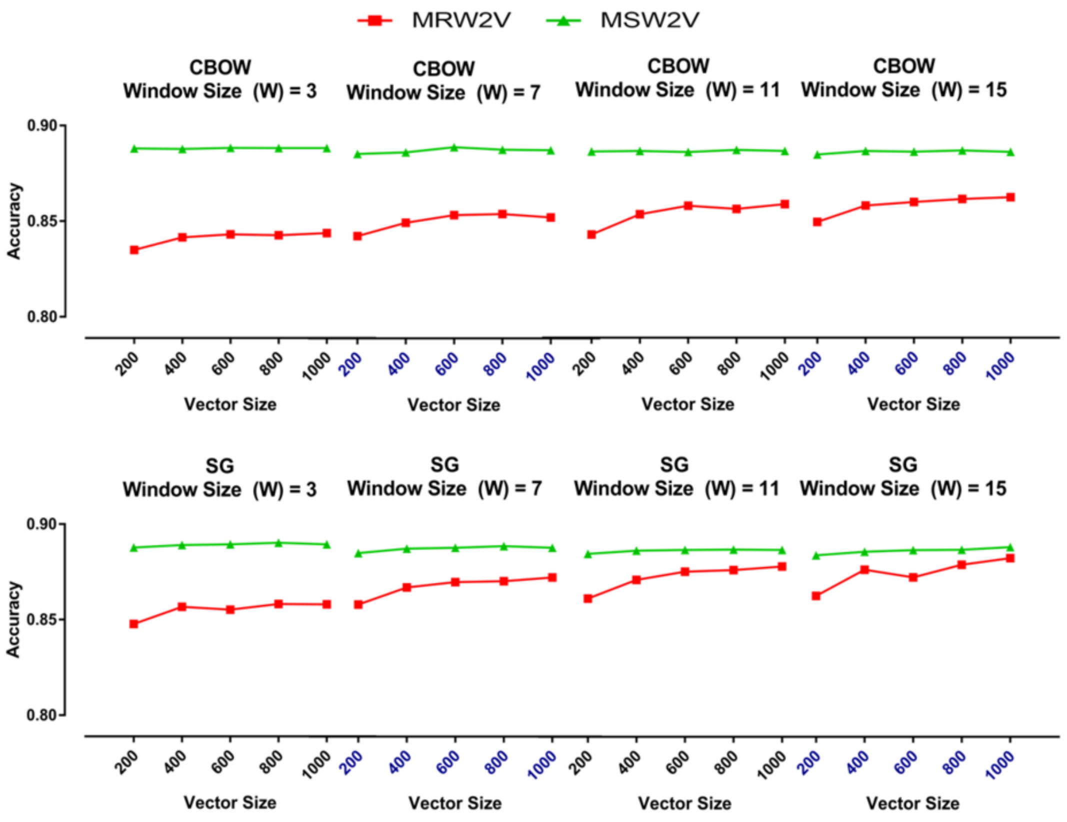
Figures 3 and 4 demonstrate the comparison between the proposed MSW2V and MRW2V for increasing vector sizes and window sizes respectively. Here, legends show the feature vectorization techniques, the vertical axis indicates the Accuracy. The horizontal axis indicates the vector size in Figure 3 and window size in Figure 4.

The outperforming accuracy of the proposed MSW2V is clearly depicted in the graph, compared to the MRW2V for both CBOW and SG architectures. It is also observed that the increase in vector size and window size results in a gradual improvement along with a slight fluctuation in the accuracy of MRW2V, for both CBOW and SG, whereas the proposed model produces consistent accuracy for varying range of vector and window size. The MRW2V efficiently preserves the semantic information but with smaller window and vector size it fails to preserve sentiment information. However, it progressively captures sentiment information with increments in the vector and window sizes. In contrast, the proposed approach achieves stable and higher accuracy since it learns the sentiment weighted word embedding. Even smaller vectors and window sizes capture sufficient sentiment information that significantly reduces the space and computational complexity.

## MSW2V VS. SPW2V

One of the major objectives of the proposed approach is to learn the sentiment weighted embedding in a distributed environment without compromising the performance compared to the non-distributed variants. This research compares the proposed MSW2V with the existing non-distributed SPW2V. The results are shown for CBOW architecture in Figure 5 and SG architecture in Figure 6. In graphs, the horizontal axis indicates varying vector and window sizes whereas the vertical axis indicates the Accuracy. Looking to results of Figure 3 and 4, V = 200, 400 and W = 3, 7 has demonstrated the

**Figure 3. Accuracy of MSW2V and MRW2V with varying Vector Sizes for CBOW and SG**



optimum results. So, the comparison of the proposed approach with SPW2V is continued with the same settings.

Both approaches utilize sentiment weighted embedding which results in superior accuracy compared to MRW2V. The proposed distributed approach is aimed to achieve comparable accuracy with respected to the existing non-distributed sentimented approach SPW2V. Graphs in Figure 5 and 6 clearly demonstrate the comparable accuracy of the proposed distributed approach. It clearly depicts that the proposed approach can learn the sentiment weighted embedding in distributed environment without compromising the quality of the feature vectors, makes the proposed model suitable for analyzing the large volume of data in distributed environment.

## CONCLUSION AND FUTURE SCOPE

Semantic and sentiment specific feature vectors are plays a significant role in ML based sentiment analysis. Word2vec effectively learns the semantic feature vectors, but fails to preserve the sufficient sentiment information. This practice leads to performance loss as semantically similar and sentimentally opposite words are in close proximity. Moreover, Word2vec is a computationally expensive technique, as a result, commodity hardware is insufficient to administrate big sized textual data. To address the aforesaid limitations, this paper proposed the MSW2V approach which can efficiently learn the sentiment and semantic feature vectors using the sentiment dictionary in the distributed environment. The experimental results are evidence that the proposed approach has shown the considerably better performance when compared to the existing distributed MRW2V in terms of

Figure 4. Accuracy of MSW2V and MRW2V with Varying Window Sizes for CBOW and SG

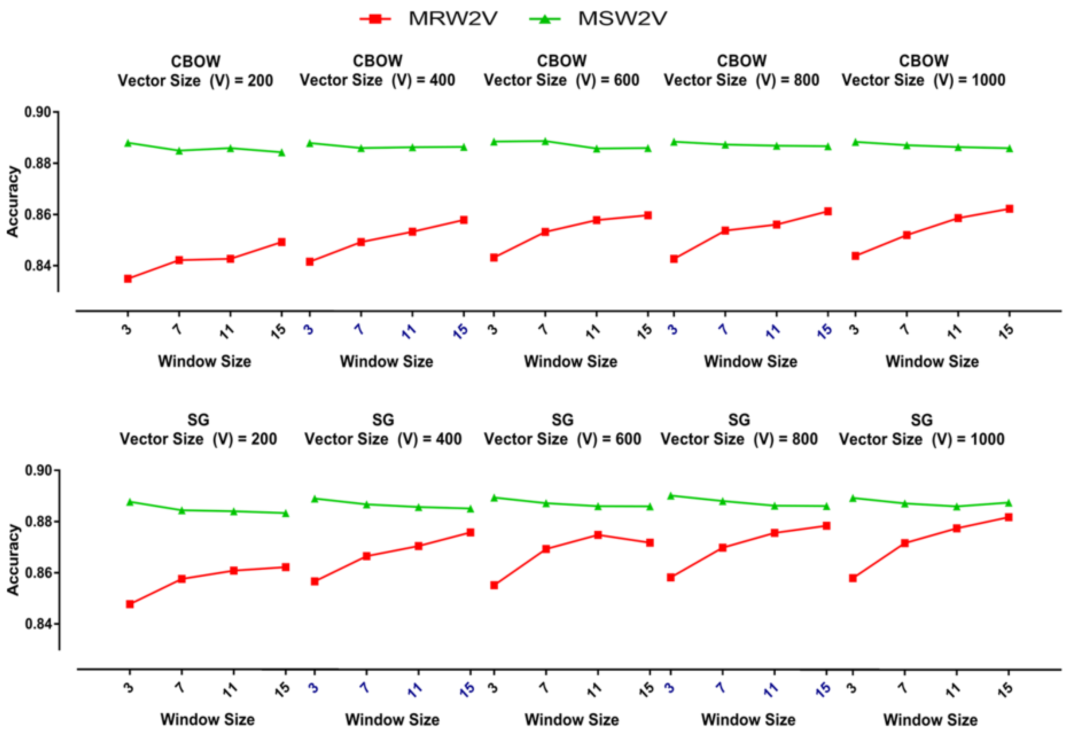


Figure 5. Accuracy of the MSW2V and SPW2V for CBOW

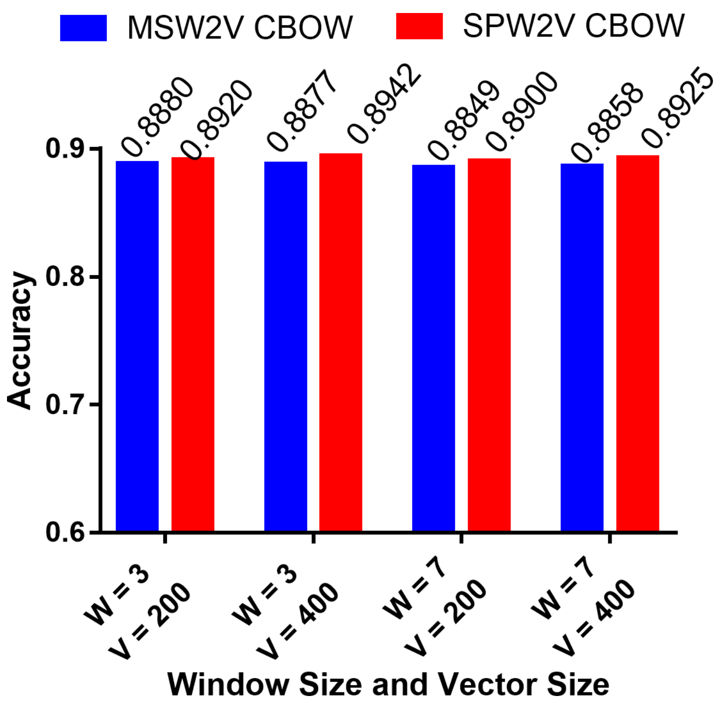
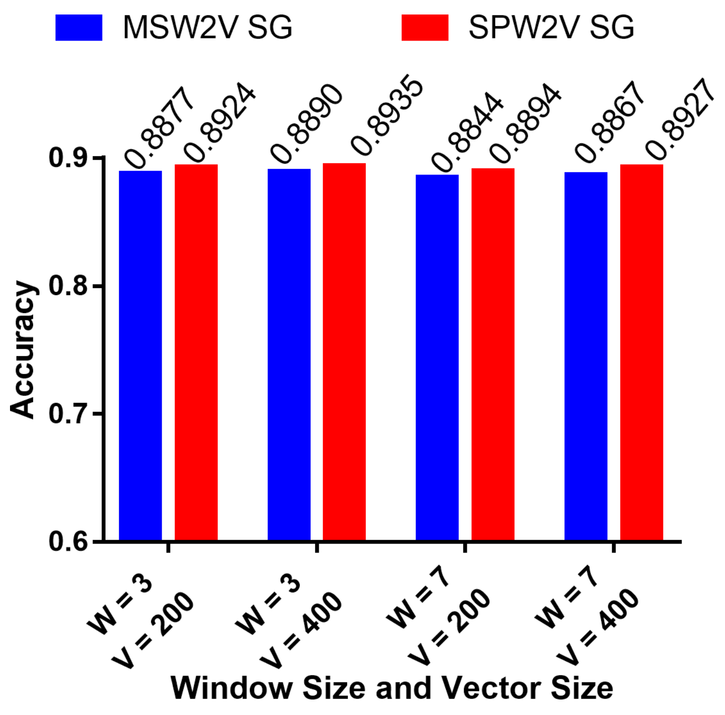


**Figure 6. Accuracy of the MSW2V and SPW2V for SG**



accuracy, and non-distributed SPW2V in terms of scalability. In sentiment analysis, the admirable performance of the proposed approach makes it practical for big data platforms like Social Media, E-commerce and Movie Streaming Service.

The proposed approach makes use of a pseudo-distributed mode of Hadoop (i.e. MapReduce) which is a batch processing system. The in-memory implementation (i.e. Apache Spark) of the proposed distributed technique is a motivating aspect for further consideration. The proposed method can be further refined by considering a fully distributed environment as well. Moreover, work can be extended for the sentiment analysis of the big streaming data, which is one of the essential demands of big data platforms.

## REFERENCES

Apache Hadoop. (n.d.). Retrieved from https://hadoop.apache.org/

Apache Spark based Word2vec. (n.d.). Retrieved from https://spark.apache.org/docs/latest/mllib-feature-extraction.htmlword2vec

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113. doi:10.1145/1327452.1327492

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. doi:10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Dhanani, J., Mehta, R., Rana, D., & Tidke, B. (2018). Sentiment analysis using novel distributed word embedding for movie reviews. In *Proceedings of 10th international conference on advanced computing (icoac)*, (pp. 138–145). IEEE.

El Alaoui, I., Gahi, Y., Messoussi, R., Chaabi, Y., Todoskoff, A., & Kobi, A. (2018). A novel adaptable approach for sentiment analysis on big social data. *Journal of Big Data*, *5*(12), 1–18. doi:10.1186/s40537-018-0120-0

Fang, X., & Zhan, J. (2015). Sentiment analysis using product review data. *Journal of Big Data*, *2*(5), 1–14.

GensimWord2vec Tool. (n.d.). Retrieved from https://radimrehurek.com/gensim/models/word2vec.html

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, *10*(1), 1–309. doi:10.2200/S00762ED1V01Y201703HLT037

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146–162. doi:10.1080/00437956.1954.11659520

Ji, S., Satish, N., Li, S., & Dubey, P. K. (2019). Parallelizing word2vec in shared and distributed memory. *IEEE Transactions on Parallel and Distributed Systems*, *30*(9), 2090–2100. doi:10.1109/TPDS.2019.2904058

Jing, X., Wang, P., & Rayz, J. M. (2018). Discovering Attribute-Specific Features From Online Reviews. *International Journal of Software Science and Computational Intelligence*, *10*(2), 1–24. doi:10.4018/IJSSCI.2018040101

Large Movie Review Dataset. (n.d.). Retrieved from https://ai.stanford.edu/~amaas/data/sentiment/

Li, Y., & Yang, T. (2018). Word embedding for understanding natural language: A survey. In S. Srinivasan (Ed.), *Guide to big data applications* (pp. 83–104). Springer International Publishing. doi:10.1007/978-3-319-53817-4_4

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, *5*(1), 1–167. doi:10.2200/S00416ED1V01Y201204HLT016

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & 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* (vol. 1, pp. 142–150). Association for Computational Linguistics.

Marneffe, M.-C. de, Manning, C. D., & Potts, C. (2010). Was it good? It was provocative. learning the meaning of scalar adjectives. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, (pp. 167–176). Association for Computational Linguistics.

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. doi:10.1016/j.asej.2014.04.011

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). *Efficient estimation of word representations in vector space.* arXiv Preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 3111–3119.

Ordentlich, E., Yang, L., Feng, A., Cnudde, P., Grbovic, M., Djuric, N., & Owens, G. et al. (2016). Network-efficient distributed word2vec training system for large vocabularies. In *Proceedings of the 25th acm international on conference on information and knowledge management*, (pp. 1139–1148). ACM. doi:10.1145/2983323.2983361

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1–2), 1–135. doi:10.1561/1500000011

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* (vol. 10, pp. 79–86). Association for Computational Linguistics.

Parikh, Y., Palusa, A., Kasthuri, S., Mehta, R., & Rana, D. (2018). Efficient word2vec vectors for sentiment analysis to improve commercial movie success. In S. Bhattacharyya, T. Gandhi, K. Sharma, & P. Dutta (Eds.), *Advanced computational and communication paradigms* (pp. 269–279). Springer Singapore.

Potts, C. (2010). On the negativity of negation. *Semantics and Linguistic Theory*, *20*, 636–659.

Pouransari, H., & Ghili, S. (2014). *Deep learning for sentiment analysis of movie reviews. Technical report*. Stanford University.

Pre-processed Large Movie Review Dataset. (n.d.). Retrieved from https://github.com/linanqiu/word2vec-sentiments

Ravi, K., & Ravi, V. (2015). A survey on opinion mining and sentiment analysis: Tasks, approaches and applications. *Knowledge-Based Systems*, *89*, 14–46. doi:10.1016/j.knosys.2015.06.015

Rezaeinia, S. M., Rahmani, R., Ghodsi, A., & Veisi, H. (2019). Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications*, *117*, 139–147. doi:10.1016/j.eswa.2018.08.044

Rong, X. (2014). Word2vec parameter learning explained. *arXiv Preprint arXiv:1411.2738.*

Scikit-learn. Machine learning library. (n.d.). Retrieved from https://scikit-learn.org/

Stergiou, S., Straznickas, Z., Wu, R., & Tsioutsiouliklis, K. (2017). Distributed negative sampling for word embeddings. *Proceedings of the Thirty-first aaai conference on artificial intelligence*.

Tang, D., Wei, F., Qin, B., Yang, N., Liu, T., & Zhou, M. (2016). Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, *28*(2), 496–509. doi:10.1109/TKDE.2015.2489653

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. *Proceedings of the 52nd annual meeting of the association for computational linguistics*, 1555–1565. doi:10.3115/v1/P14-1146

Xia, R., Wang, T., Hu, X., Li, S., & Zong, C. (2013). Dual training and dual prediction for polarity classification. *Proceedings of the 51st annual meeting of the association for computational linguistics*, 521–525.

Yin, P., Wang, H., & Zheng, L. (2012). Sentiment classification of chinese online reviews: Analysing and improving supervised machine learning. *International Journal of Web Engineering and Technology*, *7*(4), 381–398. doi:10.1504/IJWET.2012.050968

Yu, L.-C., Wang, J., Lai, K. R., & Zhang, X. (2017).Refining word embeddings for sentiment analysis. *Proceedings of the 2017 conference on empirical methods in natural language processing*, 534–539. doi:10.18653/v1/D17-1056

Zhang, D., Xu, H., Su, Z., & Xu, Y. (2015). Chinese comments sentiment classification based on word2vec and svmperf. *Expert Systems with Applications*, *42*(4), 1857–1863. doi:10.1016/j.eswa.2014.09.011

*Jenish Dhanani obtained his B.Tech and M.E degrees in Computer Engineering from Sardar Vallabhbhai National Institute of Technology, Surat and Sarvajanik College of Engineering & Technology, Surat, India respectively. He is currently pursuing a PhD from Sardar Vallabhbhai National Institute of Technology, Surat, India. He published many papers in reputed international journals and conferences. His research interests comprise Big data mining, Stream Data Analytics, Machine Learning and Natural Language Processing.*

*Rupa Mehta is an Associate professor in Department of Computer Engineering, Sardar Vallabhbhai National Institute of Technology, Surat, India. She completed her M.Tech (Research) and Ph.D from Sardar Vallabhbhai National Institute of Technology, Surat. She has several years of experience in teaching and research, also published many papers in reputed international journals and conferences. Her area of research includes Data Mining, Big Data and Artificial Intelligence.*

*Dipti Rana is an assistant professor in Department of Computer Engineering, Sardar Vallabhbhai National Institute of Technology, Surat, India. She completed her M.Tech (Research) and Ph.D. from Sardar Vallabhbhai National Institute of Technology, Surat. She has several years of experience in teaching and research, also published many papers in reputed international journals and conferences. Her area of research includes DBMS, Web Data Mining, Pattern Recognition, Prediction Mining and Big Data.*