Composite Word-Embeddings Based Improved Sentiment Analysis using Deep-Learning

A
Dissertation report
submitted in partial fulfilment of the
requirements for the award of degree of

Master of Technology
in
Computer Science and Engineering
by

Bharath T S
16535007
M.Tech (CSE)

under the guidance of

Dr. Durga Toshniwal

Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee- 247667, India

May 2018

CANDIDATE'S DECLARATION

I hereby declare that the dissertation entitled "**Composite Word-Embeddings Based Improved Sentiment Analysis using Deep-Learning**" submitted by me in partial fulfilment of the requirements for the award of the **Degree of Master of Technology in Computer Science and Engineering**, to the Department of Computer Science and Engineering **Indian Institute of Technology Roorkee** is my original work carried during May 2017 to April 2018 under the guidance of **Dr. Durga Toshniwal**, Associate Professor, Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee. The content presented in this dissertation has not been submitted by me for award of any other degree of this and any other institute.

Date:

Place: Roorkee Bharath T. S.

CERTIFICATE

This is to certify that Thesis Report entitled "**Composite Word-Embeddings Based Improved Sentiment Analysis using Deep-Learning**" which is submitted by **Bharath T.S. (16535007)** towards the fulfilment of the requirements for the award of the Degree of **Master of Technology in Computer Science & Engineering**, submitted to the Department of Computer Science & Engineering, **Indian Institute of Technology Roorkee**, India is carried out by him under my esteemed supervision and the statement made by the candidate in declaration is correct to the best of my knowledge and belief.

Date:

Place: Roorkee

Dr. Durga Toshniwal
Associate Professor
Department of Computer Science & Engineering
Indian Institute of Technology Roorkee

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor **Dr. Durga Toshniwal** for the continuous support of my study and research, for her patience, motivation, enthusiasm and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my study.

I am also grateful to the Department of Computer Science Engineering, IIT Roorkee for providing the valuable resources to aid my research.

I would like to thank all the lab colleagues, specially **Amit Agarwal** (Research Scholar) who was always willing to help and give the best suggestions.

Finally, hearty thanks to my parents and siblings, who encouraged me in good times, and motivated me in bad times, without which this dissertation would not have been possible.

Bharath T. S.

Abstract

Due to the explosive growth and popularity of Social Media sites, massive amounts of raw data are available that can be used for opinion mining and other pattern identification tasks. Identifying and summarizing the opinion or sentiment regarding any particular topic can be used to provide insights and can be taken as feedback to improve or address concerns regarding that topic. Most of the work done so far has focused on run of the mill, well-defined techniques like SVM, LDA and other machine learning techniques to classify a particular sample into one of several classes that indicates the sentiment inherently expressed in that sample. However, these techniques peak out, in terms of accuracy at a certain limit. Additional improvements to this accuracy has been reported when using deep neural networks. Segmenting dataset into multiple classes based on number or entities and the use of pre-trained word embeddings along with CNN has given the best results so far. The aim of the present work is to improve existing state of the art techniques based on word embeddings by leveraging the complementary nature of different kinds of word embeddings for sentiment classification. The performance of various models is evaluated on “Fine Food”, “IMDB Movie review”, “Yelp reviews” and “Twitter” datasets. The results obtained show that the proposed approach gives promising results

Contents

List of Figures

List of Tables

1 Introduction

Emergence of popular social networking sites like Facebook, Twitter and movie review databases like IMDB [13] has caused colossal amounts of public data which can be collected, processed and used to perform analysis based on that data to solve a vast variety of problems. Twitter and IMDB have millions of users who share their opinion regarding a wide range of topics, making it an invaluable platform for analyzing the sentiment regarding that topic. This analysis can provide insights and help in decision making in various domains. Due to the ease of availability and the quantity of this data the possible applications of sentiment analysis are varied like generating recommendation systems, box office revenue prediction, a lot of research [13] is underway in applying this knowledge effectively.

1.1. Introduction and Motivation

Sentiment analysis also known as Opinion mining, opinion extraction, sentiment mining, subjectivity analysis is a cost effective and fairly potent technique to determine public opinion [15]. This data that is available is in the textual form. It has been proved [12], [13] that there exists a positive correlation between the sentiment analyzed in the social data and the events in the actual world. In Customer-Relationship Management for large business companies it is essential that in addition to identifying trend of the opinion of products, additional information might be needed that can be put to use in taking business decisions to improve the product, or to find the areas where the customers are facing difficulties so that they can be addressed by the Company immediately. Although collection of data presents little problems, it is the interpretation of data that is challenging.

Sentiment analysis has been successfully used in various domains like determining the effective business strategy, in politics to improve the campaigning process by delivering speeches on topics that the public is most concerned about, which is obtained through sentiment analysis on their data available from social networking sites. For example, the Obama administration used sentiment analysis to gauge the public opinion to policy announcements and campaign messages prior to 2012 presidential elections [19]. There are however instances wherein the sentiment analysis failed [19], but this can be chalked up to

the wrong assumptions made regarding the input to the sentiment analysis software. Most of the data needed for sentiment is posted publicly by users.

Although collection of data presents little problems, it is the interpretation of data that is challenging. Traditional techniques have not performed well due to the presence of non-standard words which include misspelled words, presence of special characters, emoticons, regional slang as well as the high amount of noise that might be present in the communication.

Sentiment analysis can be broadly classified into two areas lexicon-based approach, and deep learning approach. In lexicon-based approaches the input data after modelling is compared with the predefined lexicon with labelled classes to determine the overall sentiment of the input text. This suffers from the drawback that it cannot identify subtleties inherent to the language like sarcasm, use of metaphors. Using a lexicon based approach for sentiment analysis can however provide a reasonable balance between the accuracy and the complexity needed. Clever use of the obtained results can be used immediately, since these computations are done by machines and within reasonable limits of error can substantially improve the decision-making capabilities.

For lexicon-based approaches, the input data, on which sentiment analysis is to be performed consists of several redundancies and noise. Thus, pre-processing to address these issues becomes a vital step in the process. This pre-processing involves stemming, removal of stop words, removal of noise, etc. Once this step is done the output of this process is given to topic modelling algorithm. These topic modelling algorithms return a set of words that are compared with a labelled lexicon to determine the overall sentiment of the words.

Sentiment analysis is the detection of attitudes. It involves identification of

1. Holder(source)
2. Target(aspect)
3. Type of attitude (set of types, polarity)
4. Attitude present in the text (present in either sentence or entire document)

Difficulty of tasks to be performed on the corpora ranging from easy to hard is as follows:

- Is the attitude positive or negative?

- Rank the attitude of the test in a range
- Detect the target about the sentiment followed by analysis of subtleties and causes regarding the sentiment expressed

Lexicon based approaches for sentiment analysis face problems with feature extraction. It is these feature that are provided to the subsequent classifiers for sentiment identification. Some of these problems include:

- Handle negation: one possible solution is to prepend "not" to all words between negation and the immediate punctuation)

- The entire input or subset of words to use to deter mine the sentiment

The deep learning approaches for sentiment analysis can implicitly model these relationships. Thus, these models are very effective but require enormous amount of data and computation to overcome the accuracy provided by the lexicon-based approaches.

1.2 Problem Description

Problem statement is as follows:

"*To Improve the performance of sentiment analysis systems using variants of deep neural network models*"

This is achieved by first dividing the input dataset into several parts depending on the named entities present per record identified using a Bi-LSTM CRF tagger and subsequently the use of pre-trained word embeddings like word2vec and Fasttext, mostly generated by hierarchical deep neural network architectures, to improve classification performance of sentiment classifiers like 1-Dimensional CNN on any dataset.

1.3 Organization of Report

The remaining report is organized as follows, section 2 describes the related work that has been done so far. Section 3 talks about the proposed scheme to improve the sentiment analysis using different variants of word embedding. Section 4 describes the experiments that have been conducted and the results obtained so far. At last the entire report has been concluded in section 5.

2 Literature Review

A lot of work has been done on performing sentiment analysis on social media text data. Many of the previously described models attempt to group the dataset into two classes the positive and negative classes and have achieved the best possible results in this area. Much of current research has been is focused on performing fine-grained sentiment analysis which involves classifying a given data sample one among several classes each indicating a varying degree of sentiment. Lately deep learning-based approaches have taken a lead in this regard. Sentiment analysis has been tackled by using several well-defined existing techniques like Naïve Bayes, SVM and much work has been done that established the trade-offs between using these techniques for several datasets [4], [16], [17]. These works attempt to identify which of the well-established techniques is best suited for sentiment analysis considering various parameters like accuracy, speed, scalability.

Each language has an inherent structure that various greatly between different languages [3], [18].Much work has been done [18] that attempts to identify the as particular variation and fine tuning of the parameters of the sentiment analysis model that are best suited for use on a particular language, like Chinese whose sentence semantic structure varies greatly from English[17].The structure of the dataset that is used for performing sentiment analysis comes in various formats each requiring a different method for pre-processing [19]. Data crawled from social media sites like twitter have samples in which the language is short, unstructured making it harder for identifying the sentiment [13], [19].

Hashtags within tweets provide additional information that has be used in pre-processing step of sentiment analysis models to identify the targets or the subject referred to in the tweet [17]. These hashtags also provide explicit information regarding the sentiment expressed by the tweet [19]. Identification of aspect targets before sentiment analysis has improved the results of conventional generative probabilistic models like LSI, LDA [13]. Tan et al. [19] have used a hierarchical LDA model to first identify candidate tweets. Based on these candidate tweets, the foreground and background tweets have been collected and sentiment analysis has been done specific to these tweets. Gibbs sampling [19] has been finally used to identify the sentiment in the respective candidate tweet. Latent Dirichlet Allocation (LDA) has been found to be the most efficient topic modelling when large amounts of data are to be processed. Only some of the drawbacks of LDA have been

addressed and possible approaches to perform complex sentiment analysis using variations of the basic LDA have been addressed in [19]. It has been reported that using two separate LDA where the results of one LDA has been utilized as input to the second LDA provides much better analysis of the sentiment [19].

2.1 Probabilistic Techniques for Sentiment Analysis

The basic objective of any language modelling technique is to determine the likeliness of the occurrence of a word or a phrase or any sentence. In machine translation, the goal of language modelling is to assign a probability to a sentence. We would like to be able to distinguish between good and bad translations by their probabilities. For example,

$$P\,(high\ rise\ buildings) > P\,(big\ rise\ buildings)$$

This is evident by literature since high and rise go well together in the context. In Spell Correction from the present literature it can be found that:

$$P\,(colleague) > P\,(collegue)$$

Thus, it would make more sense to use the first spelling since, it's more likely. Also, language modelling can make certain distinctions using the semantic relationship among the words which can be especially useful when using machine speech translated words. For example,

$$P\,(I\ saw\ a\ van) > P\,(eyes\ awe\ of\ an)$$

Therefore, given any sequence of words $w1, w2, w3 \ldots wn$, we need to compute

$$P\,(W) = P\,(w1, w2, w3, \ldots\ldots\ldots\ wn)$$

Where W is the sentence and w_i is the word. This can then be later used to find the probability of the upcoming word in the sequence.

$$P\,(w5) = P\,(w5|w1, w2, w3, w4)$$

Therefore, a model [20] that is capable of computing either of these two probabilities, that is, joint probability of the entire string P (W) or the conditional probability of a particular word $P\,(wn|w1, w2, w3, \ldots\ldots\ wn)$ is called a language model. This models the grammar of the language.

Using the chain rule, it is possible to determine the probability of any specific instance of random variable using the joint distribution of a collection of random variables using just the conditional probabilities of the random.

$$P(A_n, \ldots . A_1) = P(A_n | A_{n-1}, \ldots . , A_1) . P(A_{n-1}, \ldots , A_1)$$

Recursively repeating the process for each final term yields:

$$P(\bigcap_{k=1}^{n} A_k) = \prod_{k=1}^{n} P(A_k | \bigcap_{i=1}^{k-1} A_k) \quad (1)$$

As stated in equation (1) this would require computing these probabilities would be too expensive since lot of sentences might be possible. Thus, a simplifying assumption is made, called Markov assumption.

Allows for the relaxation of number of terms needed to compute the posterior probability [20]. It states that

$$P(w_1 w_2 \ldots . w_n) \approx \prod_i P(w_i | w_{i-k} \ldots w_{i-1}) \quad (2)$$

Applying to the chain rule product for each component in the product and restricting the prefix to previous k prefix words in equation (2), following conditional probability is obtained:

$$P(w_i | w_1 w_2 \ldots . w_{i-1}) \approx P(w_i | w_{i-k} \ldots w_{i-1}) \quad (3)$$

The most commonly used sentiment lexicons are General Inquirer (positive vs. negative), LIWC (Linguistic Inquiry and Word Count) >70 classes, MPQA Subjectivity Cues Lexicon (6885 words from 8221 lemmas annotated with intensity), SentiWordNet (All wordnet words are automatically annotated for degrees of positivity, negativity, neutrality / objectiveness.

Unigram model is the simplest model in which the joint probability of a whole sequence of words is computed by computing the probabilities of each word separately and taking their product [20]

$$P(w_1 w_2 \ldots . w_n) \approx \prod_i P(w_i) \quad (4)$$

Using this equation (4) model to generate words would result in generation of words that are seemingly random assumption of independence is taken while generating each word.

E.g. the sample words generated by this model would be as follows:

a, he, there, plant, sun, cloud, drive, …etc.

A bigram model is a Slightly cleverer model, [21] in which we estimate the probability of a word by considering only the immediately preceding word

$$P(w_i|w_1w_2 \ldots\ldots w_{i-1}) \approx P(w_i|w_{i-1})$$

E.g.: The sample words generated by this model would be as follows: Outside, new, car, parking, lots, of, the, agreement, reached …etc.

The previous approach can be extended to trigrams, 4-grams, 5-grams. But intuitively it is clear that the model developed to represent the language is primitive and incomplete. In many languages including English there exist dependencies between several parts in the corpus that can only be determined from the context. This context can be determined by remembering the information of the context present in the beginning.

E.g.: After a long day of work, the man returned to his home tired.

Unless trying to model extensively complicated relationships N-gram models sufficient.

To compare and evaluate model an evaluation technique is needed that assigns a higher probability to real or frequently observed sentences than ungrammatical or rarely observed sentences. For the language modelling the most commonly used metric is perplexity. It is sometimes a bad approximation unless the test data looks just like the training data.

The best language model is the one that best predicts an unseen test set. Perplexity is the probability of the test set, normalized by the number of words :

$$PP(W) = P(w1, w2, \ldots., wn) - 1/N$$

Expanding using the chain rule we get,

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1w_2 \ldots\ldots w_{i-1})}}$$

For a bigram model this would be expanded as

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} \quad (5)$$

In equation (5) due to the inversion, minimizing perplexity is equal to maximizing probability.

2.2 Text Classification using Naïve Bayes classifier

Text classification is the technique that encompasses a wide variety of tasks such as assigning categories, topics, or genres, spam detection, authorship identification, sentiment analysis. By definition, given a document and a set of classes, to assign the document to one of the classes in the set. Some of the most commonly used classifiers are Naïve Bayes classifier, SVM [21].

A simple naïve classification is based on Bayes Rule. It relies on a very simple representation of the document known popularly as the "bag of words" representation. The bag of words model loses all information regarding the order of the words in the document. It is represented as a vector of word along with its associated count.

It represents a function that returns whether the class is positive or negative with respect to sentiment analysis. In this context, the entire collection of words or a subset of words can be used.

Given a corpus or a document d and the distribution of classes c

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

C_{MAP} is the mapping of the document to a particular class

$$c_{MAP} = argmax_{c\epsilon C} P(d|c)P(c)$$
$$= argmax_{c\epsilon C} P(x_1, x_2, \ldots . x_n|c)P(c)$$

Where $x1, x2, \ldots, xn$ refers to the random variables representing the features (vocabulary) for the document.

Independence assumption for multinomial Naïve Bayes is as stated above which is dependent on some of the simplifying assumptions made such as the "bag of words" assumption and conditional independence assumption [20]

In conditional independence assumption, it is assumed that the feature probabilities P (x_i, c_j) are independent given a class c .

$$P(x_1, \ldots . x_n|\, c) = P(x_1|\, c).P(x_1|\, c).P(x_2|\, c) \ldots \ldots P(x_n|\, c) \quad (6)$$

Thus, the equation (6) can be used for obtaining C_{MAP} as

$$c_{NB} = argmax_{c \epsilon C} \prod_{x \epsilon X} P(x|c) \quad (7)$$

Each class in a naïve Bayes classifier is a unigram language model. The class that assign the highest probability is the mapped as the class of that document.

It is a very fast algorithm, with low storage requirements. Its robust to irrelevant features (they cancel out each other), works very well in domains having many equally important features . If the independence assumption holds then Naïve Bayes is the optimal classifier, but this assumption rarely holds. Also, Naïve Bayes is a "high bias" classifier that works well with small amounts of data.

2.3 Generative Probabilistic Models for Sentiment Analysis

2.3.1 Latent Semantic Indexing

Latent semantic indexing [21] is a popular topic modelling technique in which in addition to the previously described techniques, which capture the number of words in a document, it also examines the document collection as a whole, to see which other documents contain the similar set of words. This method is similar to how a normal human being would categorize a set of document collection. When search is made on an LSI indexed collection, it returns those documents after looking at the similarity values it has calculated for every content word. LSI doesn't require an exact match, in cases where keyword search fails if there is no occurrence of that particular word, LSI will often return those documents that it believes has the closest match.

For example, consider three terms chocolate, vanilla, cake, now assume that these items are very closely related in that they occur almost always together. Now assume that a search is made for items "chocolate vanilla ", LSI is able to understand that due to high correlation between the above three terms it returns results which contain "cake" in addition to results that contain "chocolate vanilla".

Improvement in results can be obtained by using the concept of term weighting. It works on the principle that most frequent words are meaningful within a document but rarely occurring words are more interesting. The first part is called local weighting and the values

are generated after scaling by a logarithmic factor. The second part corresponds to the global term weighting. The most commonly used technique is inverse document frequency.

The final part is to use normalization of values generated by the individual documents. This is done to avoid bias towards large documents in comparison to smaller documents that have relatively fewer terms.

Single value decomposition reduces the matrix down to a smaller set of components. The algorithm returns a matrix of the same shape as the original, which can then be used as a lookup grid. This matrix is an approximation to the original input term document matrix. On viewing the resultant matrix it can be seen that there are very few zeroes and some negative values indicating a very large semantic distance between the two documents. For example, consider three columns ['blue', 'red', 'transparent'] this would be possibly reduced to [(1.263*'red' + 0.567*'blue'), transparent]

Let X be a matrix where the value in $(i\ ,j)th$ entry represents the occurrence of term i in document j .

$$\begin{matrix} & \mathbf{d}_j \\ & \downarrow \\ \mathbf{t}_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \end{matrix}$$

The dot product between the terms $t^{iT}t^{p}$ gives the correlation between the terms over the documents. The matrix product XX^T contains all possible dot products. From the theory of linear algebra, there exists a decomposition of X such that U and V are orthogonal matrices and $\sum$ is a diagonal matrix. This process of decomposing the original matrix is called single value decomposition (SVD).

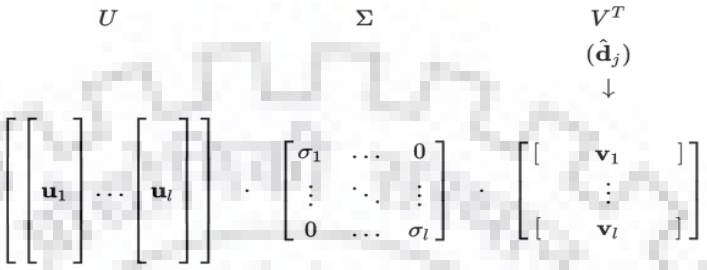
$$X = U\Sigma V^T \tag{8}$$

The correlation between the words and documents expressed as matrix products become:

$$XX^T = (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V^{T^T}\Sigma^T U^T) = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T$$
$$= U\Sigma^2 U^T$$

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) = (V^{T^T}\Sigma^T U^T)(U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T = V\Sigma\Sigma^T V^T$$
$$= V\Sigma^2 V^T$$

Since $\sum\sum T$ and $\sum T\sum$ are diagonal matrices U must contain Eigen vectors of XX^T, while V contains Eigen vectors of X^TX. Both have non-zero Eigen values given by non-zero entries of $\sum\sum^T$, or equally by, non-zero entries of $\sum^T\sum$



The document vector is an approximation of the corpus in lower dimensional space. This can be written as:

$$X_k = U_k \Sigma_k V_k^T \quad (9)$$

The resultant dimensions of X_K in equation (9) are difficult to interpret. The relationships that are formed due to the occurrence of words within the document which though justifiable mathematically have no interpretable meaning in natural language. This model cannot capture polysemy (one word having many meanings) since each occurrence of the word is treated as having the same meaning due to the word being represented as a single point in space.

2.3.2 Probabilistic Latent Semantic Analysis

The main goal of this technique is to model the co-occurrence of terms under some probabilistic framework obtained through the latent semantic structure present in the document. These forms a latent variable model. The probabilistic structure of the model is obtained from a statistical model called the aspect model, where the latent (hidden) variables model the topics/concepts which are associated with the observed variables which are the texts and documents . Similar to the LSI, pLSA takes a sparse co-occurrence matrix and reduces its dimensionality. Thus, pLSA uses the information in the co-occurrence

matrix to extract the inherent topics and model the document as a mixture of these topics [20].

pLSA [20] considers that the data can be represented using three sets of variables:

1. Documents: $d \in D = \{d_1, d_2, d_3 \ldots . d_N\}$ where N is the total number of documents in the corpus in consideration.
2. Words: $w \in W = \{w_1, w_2, w_3 \ldots . w_M\}$ where M is assumed to be the total size of the vocabulary or the distinct number of words.
3. Topics: $z \in Z = \{z_1, z_2, z_3 \ldots . z_k\}$ where K is the total number of possible topics (assumed and specified a priori) in the document.

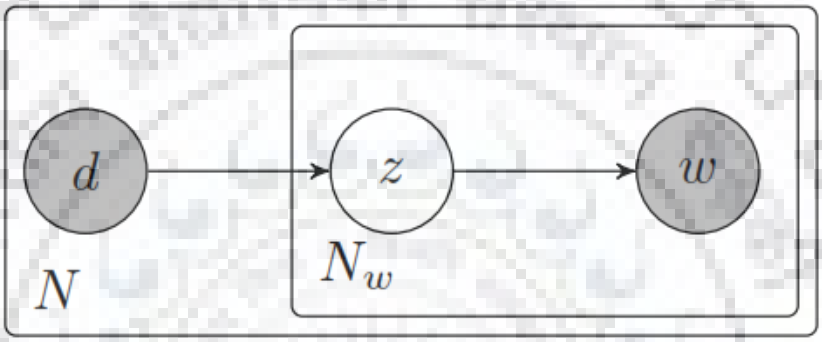


Fig 1 The graphical representation of latent variable model using plate representation

In Fig. 1 the shaded circles indicate the observed variables and the unshaded the hidden variables. Each word has an associated latent topic z. There are N_w number of words in the document. There are N such documents.

Some assumptions made by the model include

1. Bag of words: Each document is considered to be a collection of unordered words, mathematically the joint variable (d, w) is independently sampled and the joint distribution of these variables can be factorized as a product:

$$P(D, W) = \prod_{(d,w)} P(d, w)$$

2. Conditional independence: The words and documents are assumed to be conditionally independent given the topic :

$$P(w, d|z) = P(w|z) \, . P(d|z)$$

We obtain the $P(d, w)$ by using the product rule:

$$P(d, w) = P(d)P(w|d)$$

Using the conditional independence assumption, this is simplified as:

$$P(w,d) = \sum_{z \in Z} P(z).P(d|z).P(w|z) \tag{10}$$

In equation (10) the parameters of the model are $P(w|z)$ and $P(z|d)$. The number of these items are $(M-1)K$, respectively $N(K-1)$, which implies that the total number of parameters grows linearly with the size of the corpus.

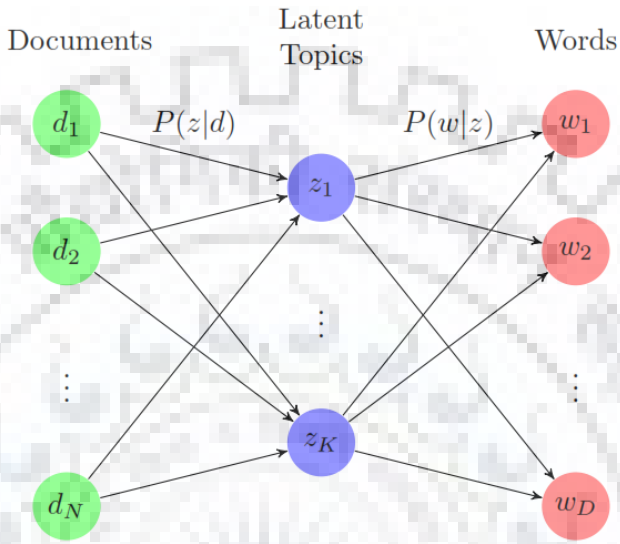


Fig 2 The general structure of pLSA model

The Fig 2 shows the intermediate layer of latent topics that links the documents and the words: each document can be represented as a mixture of concepts weighted by the probability $P(z|d)$ and each word expresses a topic with probability $P(w|z)$

$$A \approx \underline{A} = L.R$$

In terms of matrix notation this can be rewritten as

$$A = L.U.R$$

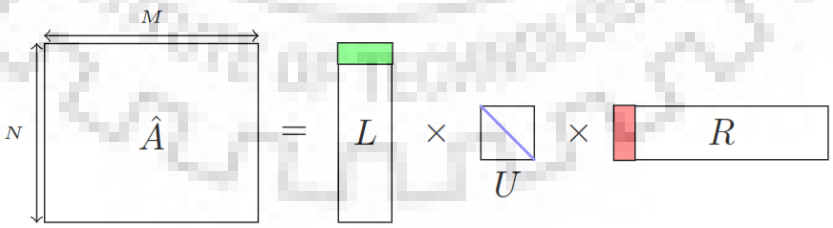


Fig 3 Visualizing the pLSA in terms of matrix decomposition technique

In Fig. 3 the matrix A denotes the document term matrix. The green row represents the probabilities over the document $P(d|z)$, the blue diagonal represents the probabilities over

all topics $P(z)$ and the red column represents the probabilities of the word being generated by each topic $P(w|z)$ Here, L contains the document probabilities $P(d|z)$, U the diagonal matrix of the prior probabilities of the topic $P(z)$ and R, the word probability $P(w|z)$.

These matrices are non-negative and normalized, as they represent a probability.

2.3.3 Latent Dirichlet Allocation(LDA)

LDA is a generative probabilistic model for collections of discrete data, [20]. Documents are represented as random mixtures over latent topics. It is a hierarchical three level Bayesian model, in which each document is modelled as a finite mixture over an underlying set of topics. Each topic in turn is modelled as an infinite mixture over an underlying set of topic probabilities. It is essentially a dimensionality reduction technique that preserves the latent topic implicit in the data, making it an effective technique applicable not only in topic modelling for textual data but also for image processing [20].

In the context of text modelling some notations used are:

- Word: is the basic unit of discrete data, is an indexed item from the vocabulary $\{1, \ldots . V\}$, represented as unit basis vectors. Using the superscripts to denote components, the v^{th} word in the vocabulary is represented by a V-vector w such that $w^v = 1$ and $w^u = 0$ for all $u \neq v$
- document: is a sequence of N words denoted by $w = (w_1, w_2, \ldots, w_N)$, where w_n is the n^{th} word in the sequence
- corpus: is a collection of M documents denoted by $D = \{w_1, w_2, \ldots w_M\}$

A finite set of random variables is said to be exchangeable if the joint distribution is invariant to permutation [1]. If π is a permutation of integers from 1 to N :

$$p(z_1, z_2, \ldots . . z_n) = P(z_{\pi(1)}, z_{\pi(2)}, \ldots . . z_{\pi(N)})$$

An infinite sequence of random variables is infinitely exchangeable if every finite subsequence is exchangeable. De Finetti's representation theorem states that the joint distribution of and infinitely exchangeable sequence of random variables is as if a random parameter were drawn from some distribution and the random variables were independent and distributed conditioned on that parameter. In LDA we assume that words are generated by topics and those topics are infinitely exchangeable within a document. By De Finetti's theorem the probability of a sequence of words and topics must therefore have the form :

$$p(w,z) = \int p(\theta) \left(\prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n)\right)d\theta \tag{11}$$

In equation (11) θ is a random parameter of multinomial over topics.

Taking the product of the marginal probabilities of the single documents, we obtain the probabilities of the corpus :

$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn},\beta)\right)d\theta_d \tag{12}$$

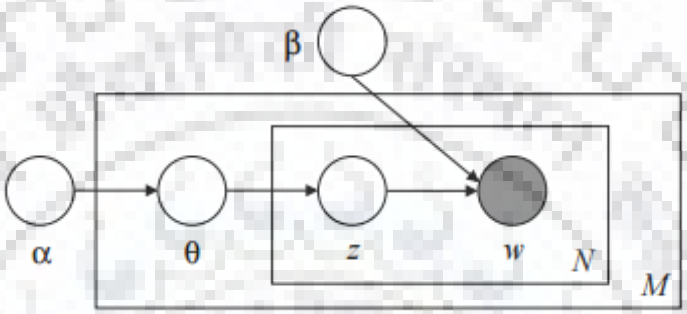


Fig 4 The LDA as a graphical probabilistic model

The rectangular boxes indicate the replications, and these are represented as "plates". The plates on the outside represent documents, and the inner plates represent document topics and words preset within a document.

In equation (12) the parameters $\alpha$and β are corpus level parameters assumed to be sampled once while generating the entire corpus. The variables θ_d are document level variables sampled once per document. The variables z_{dn} and w_{dn} are word level variables and are sampled once for each word in each document.

A sentiment classification model can be built on the output of LDA. The classifier most commonly used is a SVM since it has a well-known property of being capable of handling very high dimensional data [4]. The entire working using LDA model can be described as:

1. The input corpus is pre-processed, which includes removing stop words and stemming
2. This resultant matrix is given as input to the LDA model. The associated parameters are inferred by Gibbs sampling, providing value that indicate latent values for the matrix of document rows and topic columns

3. After obtaining the above matrix, this data is used to train a classifier. Here a Support Vector Machine (SVM) [6]. SVM is a supervised learning approach. The class labels for individual words are obtained by referring to some existing pre-classified lexicons

4. Various weighting schemes can be used to combine and aggregate the results of the classification for individual terms [8].

5. The aggregated result is used to indicate the topic (classification based on sentiment) for the entire document

2.4 Named Entity Recognition for Parts-Of-Speech Tagging

The best-known model [12] for sentiment classification has been achieved through sentence classification using target-based segmentation with the help of various Parts-Of-Speech (POS) taggers which is a subtask of Named-Entity-Recognition [14]. In information extraction, the task of Named entity recognition (NER) is to identify real world targets present in structured information, namely sentences. These targets may refer to person, location, time or any particular aspect that is identifiable by a human being. This involves forming relationships between different parts or 'Chunk's in a sentence. The best systems for NER have achieved close to state of the art performance [10]. The best system, MUC-7 has achieved an F1-score of 93.39% whereas human annotators were able to achieve 97.60% [12]. Much of the research [12] in this field now focusses on semi-supervised approaches to be capable to effectively tagging datasets in completely different domains. Deep learning approaches have taken a lead in this cross-domain semi-supervised learning approaches as they do not require handcrafted features specific to the domain to be capable of performing classification [11] accurately. Hovy et al. [11] have proposed and developed a Parts-Of-Speech (POS) tagger and NER system taking advantage of the Bi-LSTM neural network architecture along with Conditional Random Fields (CRF).

2.5 Representing words in the Corpus

2.5.1 Term Frequency and inverse document frequency

Vector representation doesn't consider the ordering of words in a document. The term frequency $tf_{t,d}$ of the term t in document d is defined as the number of times that t occurs in d. [11]

Rare terms are more informative than frequent terms. Consider a term in a query that is rare in a collection. A document containing this term is very likely to be relevant. Frequent terms are less informative than rare terms. But consider a query term that is frequent in the collection, therefore a document containing such a term is more likely to be relevant than a document that isn't. The document frequency df is used to capture this notion.

df_t is the document frequency of t: the number of documents that contain t.

- It is an inverse measure of the relevance of t
- $df_t <= N$

Inverse document frequency of t is:

$$idf_t = log_{10}(N/df_t) \tag{13}$$

Log value is taken to dampen the effect of idf.

The tf-idf weight of a term is the product of its scaled tf term and its idf weight

$$w_{t,d} = (1 + logtf_{t,d}) * log_{10}(N/df_t) \tag{14}$$

The equation (14) has been experimentally proved [21] to be the best-known weighting scheme in information retrieval. Its value increases with the occurrences within a document and with the rarity of the term in the collection.

2.5.2 Word Embeddings

Each unique word in the vocabulary has to be represented using a vector. Using binary number for each word introduced unrelated dependencies between unrelated words. This is addressed with one-hot vector encoding for words. But this notation resulted in each vector for a word having the length equal to the size of the vocabulary, which was impractical to work with even for moderately sized vocabularies. This was addressed with the development of word embeddings for individual words.

Each word in the vocabulary of the dataset is represented as a unique vector. Thomas Mikolav et al [1], [2] have developed a system that generates a word vector for each word in the vocabulary that represents the sematic and syntactic meaning with the help of a unique neural network model, namely skip-gram and continuous bag of words model. These models generate the word capture the context associated with any word in a shallow window.

Jeffrey Pennington et al [4] developed another system that follows a global co-occurrence count based approach to generate a word vector for each unique word. This approach clearly captures the aspect referring to the topic which the word belongs to. This approach is easily scalable with extremely large dataset sizes. The word2vec model however gives better results but has a much higher training time.

Armand Joulin et al. [5] from Facebook research have recently developed a context-based word-vector generation systems that generates a word vector taking advantage of character level information present in the language that is much more scalable having much less training time compared to all previous approaches that generates word vectors than can directly be used to perform simple syntactic tasks like word analogy, synonyms, etc. directly using the generated Word-embeddings.

2.6 Sentiment Analysis

Sentiment analysis is done at different levels of granularity, namely document level, sentence level and word level as described by Zhao et al. [13]. With the decrease in the granularity of the sentiment analysis the accuracy of the system performing sentiment decreases [13].

Deep neural networks like RNN, LSTM have shown promising results in sequence identification tasks [12]. Yoon Kim [6] has describes a deep convolution neural network approach for sentiment analysis that takes advantage of pre-trained word embedding, namelyword2vec to perform sentiment analysis. This described approach improves upon previous works in this field. Tao Chen et al. [12] have described a sentiment analysis system that builds upon the work done by Yoon Kim [6] to improve the accuracy of the sentiment analysis systems by first identifying the aspect targets in sentences and classifying the input dataset into different classes based on the number of these aspect targets.

2.7 Research Gaps Identified

Use of pre-trained word embeddings have improved the accuracy of the of any sentiment classifiers [8]. There are several complementary word embeddings that capture different aspect of the meaning that a word supposedly represents. Much of the currently available work have used only single word embedding during training. Complementary word embeddings have not yet been used to further improve the results of the sentiment analysis model. Thus, a combination of word embeddings can be used to capture a more complete manner that can possible improve the accuracy of the classification system.

3 Proposed Work

In the proposed work, initially target-based segmentation is used to segment the dataset based on the number of targets identified per sentence. Each of these datasets are then individually used to train independent classifiers. This segmentation of dataset based on number of targets has been proven to improve the overall classification accuracy [12].

For classification a one-dimensional CNN classifier with different embeddings are used. Complementary word embeddings have not yet been used to further improve the results of the sentiment analysis model. Thus, a combination of word embeddings can be used to capture a more complete manner that can possible improve the accuracy of the classification system. The framework following this approach shall henceforth be referred to as Target Based Segmented Reduced Composite (TBS-RCE) model.

The outline of the proposed framework is shown in the figure 1.

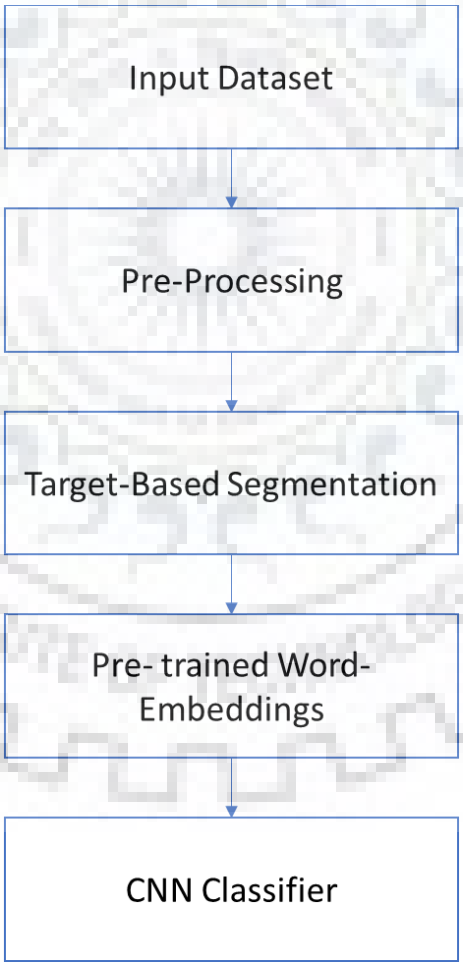


Fig 5 Proposed Framework

Initially the number of targets present in the sentence is identified by using a trained Bi-LSTM-CRF network. Based on the number of targets in a sentence the dataset is divided into zero-target, one-target and multiple target sentences. The vector representation for these sentences are generated using word embeddings obtained by several pre-trained models for comparative analysis. These vectors along with corresponding labels are used to train a 1-D CNN and finally the accuracy of the system is evaluated on the test dataset.

3.1 Target Identification and Sentence Type Classification Model

A sequence model is used to scan the input model to identify the number of aspect targets referred to in a single sample. This model assigns IOB (Inside, Outside and Beginning) tags to each word in a sentence. This task is similar to Named Entity Recognition (NER). These systems locate and tag the named entities in the sample dataset to categories like people, organization, time, location, etc. These sequence models require the training dataset to be labelled by either the Part Of Speech (POS) tags or Inside-Outside-Beginning (IOB) tags. For example, given a sentence like

"Rahul bought thirty chocolate ice-creams from Baskin-Robbins in July"

The task for named entity recognition aims to annotate this sentence to give:

[Rahul] $_{PER}$ *bought thirty chocolate ice-creams from [Baskin-Robbins]* $_{ORG}$ *in [July]* $_{TIME}$

Most of the previously defined techniques are domain-specific and perform poorly on general dataset different from the training dataset [9]. State of the art results have been achieved by using deep neural network sequence models for tagging general data such as tweets, movie reviews [9].

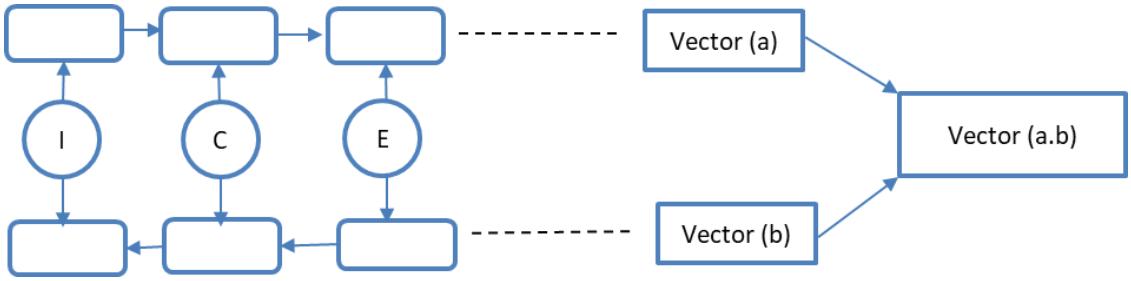


Fig 6 Bi-LSTM framework for NER

Bi-LSTM-CRF network is one such model that has shown to provide considerable success in the task of NER [14]. Sequence learning is done by "Bi-directional Long term short term memory (Bi-LSTM)" along with "Conditional Random Fields (CRF)" together forming a unit. BiLSTM [11] is a variant of RNN which incorporates two LSTM, a forward LSTM used for learning the information from the preceding tokens to generate the label for the current token and backward LSTM that learns to generate the label for the current token from its succeeding tokens. The forward LSTM and backward LSTM together form the basic hidden unit for the Bi-LSTM architecture.

3.2 Word Embedding for Individual Words

Word embedding for the individual words are used to direct the neural network That perform sentiment analysis, to interpret the meaning of individual words correctly so that the network being trained minimizes the errors due to the falling into local minima, when the neural network is optimized, using mini-batch gradient descent technique. Each unique word in the vocabulary can be represented using a number from a counter that increments for each new word encountered. This however takes up a lot of space in the memory. To account for this each word may be represented by the binary number indicating the corresponding number in the original counter. This however introduces non-existent relationships between the words due to the number of set and unset bits in the binary notation of unrelated words. To avoid the formation of non-existent relationships each word is represented in the one-hot vector notation. This however results in unmanageable size vectors when the size of the vocabulary of the corpus grows too large. To overcome this each word is represented as a point in a fixed dimension space. Thus, each word is embedded into an n-dimension space, giving rise to the name word-embeddings. This

notation provides for representing words with limited size vectors of real numbers. There are several methods of arriving at the position for each word. Some of the methods to arrive at embeddings for each word in the corpus is described below:

3.2.1 Random Vectors:

These vectors provide a baseline model upon which each successive model will be evaluated. In this approach each word is represented by a random vector of fixed dimension. This implies that each word is randomly embedded into a fixed dimension space. This introduces non-existent relationship between two completely unrelated words by the virtue of their proximity to each other in the embedding space. These vectors are used to train the basic CNN model.

3.2.2 Pre-trained embedding vectors:

Word2vec vectors are generated from a skip-gram model [2].

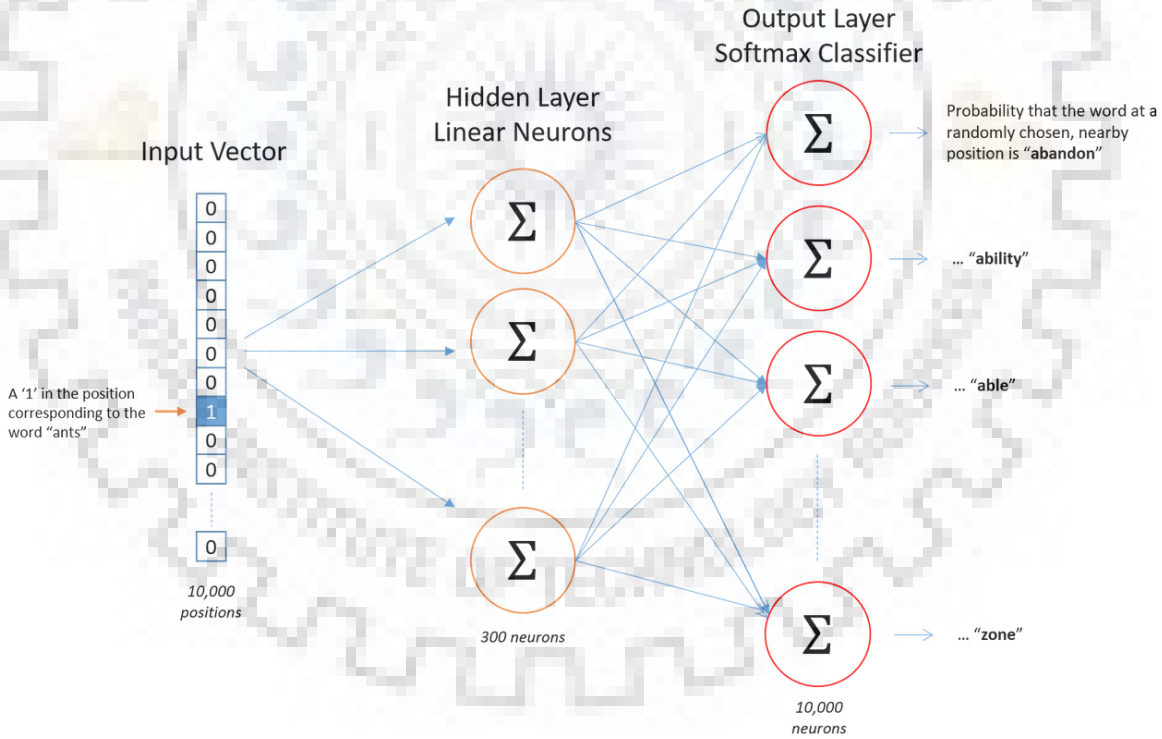


Fig 7 Word2vec model

A skip-gram model is predictive model that that tries to improve the predictive ability of predicting a target word given a set of context words taken from a shallow window [2]. The objective of this model is to minimize the following cost function:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log \mathrm{p}(w_{t+j}/w_t) \tag{15}$$

In equation (15) T is the total size of the vocabulary, c is the window size, t is the current word. The task of the model is to learn the weights for the 300 neurons in the hidden layer. These weights themselves represent the distributed representation for a particular word. The output of the output layer represents the probability distribution for the remaining words. A softmax (multinomial logistic regression) classifier is applied to output the most relevant word as a one hot word vector [1]. Each word has two vectors, one represents it in the context of a center word and the other as a word in context for the center word. This separation allows a distinction in case of words like New Delhi, where a there should be a high probability of new being in context of Delhi, considering Delhi as a center word with new being in the context window, whereas the probability of Delhi as a context word for new as center should be low, to reflect the distribution of occurrence of this word after new [1]. The model that uses this embedding shall henceforth be referred to as wor2vec embedded (W2VE) classifier

Glove embedding Vectors combines the count-based word embedding generation model with the skip-gram model for word generation to improve the training speed. This involves the generation of a co-occurrence matrix(X) for each pair of words. The objective function of this model is given by:

$$J=\sum_{i,j=1}^{V} f(X_{i,j})(w_i^T\widetilde{w_j}+b_i+b_j-\log X_{i,j})^2 \tag{16}$$

In equation (16) f is the weighting function to clip the weights of those pair of words that have disproportionately high counts which results in the neural network being trained over and over again for each new occurrence of the pair. The terms b_i and bj represent the bias terms that are added to maintain the symmetry of the words i and j. wi, wj represent the inside and outside vectors similar to the one used in training skip gram vectors. In this

model we take the sum of these to vectors to represent the individual words [4]. The model that uses this embedding shall henceforth be referred to as glove embedded (GE) classifier.

Fasttext [5] interprets a single word as a collection of n-gram characters obtained from the individual word. Thus, a word like 'windy' is represented by the collection of chunks of the original word, that is, the chunks ['windy', 'wind ', 'win', 'wi', 'w'] represent the original word. This is useful for generating embedding for rare words and words that appear from out of the vocabulary of the training dataset [5]. The word2vec and glove models fails to generate word vectors correctly for these kinds of words that imply the closest meaning to the intended correct interpretation.

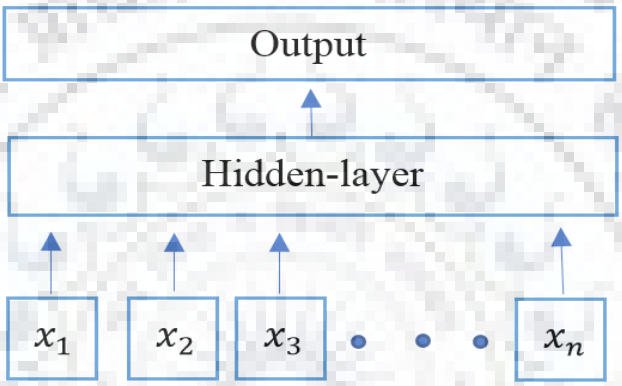


Fig 8 Fasttext model

In fig. 3.4 $x_1, x_2, \ldots . x_n$ represent the n-gram features (chunks) of words that are fed together to the hidden layer, the vector obtained by summing these values

The model is built to be able to deal with extremely large datasets efficiently [5]. It uses a tree hierarchical classifier by grouping words belonging to different categories under different branches and then building the tree based on the frequencies of the words in each branch using Huffman algorithm [5]. This deals with uneven distribution of categories. A chunk of the entire text is represented by the sum of vectors of individual chunks, called hidden state, which is shared among each branch. Experimental results have shown the performance of the system is on par with state of the art classification techniques [5] while the training time is much lower than the state of the art techniques. The model that uses this embedding shall henceforth be referred to as Fasttext embedded (FTE) classifier.

3.2.3 Composite Embedding vectors:

Word2vec vectors capture the local information about a word well, glove embedding vectors capture the global information. Fasttext embeddings intuitively determine the information regarding the root word for a family of words. This is similar to how words in English language are formed. Word2vec model has the best distinction capability by considering a window for each word, in each of its occurrence and thus aptly takes the longest to train [2], whereas Fasttext takes only a fraction of this time while also capturing the root meaning of individual words [5]. The vector embeddings for a word, generated by these two vectors are combined to get a joint representation for a word which is then used to represent a single word. This joint embedding is provided as input to the neural network for sentence classification. The classifier that uses this embedding is referred to as Composite model. The model that uses a combination of word2vec and Fasttext embedding shall henceforth be referred to as Composite Word Embedding Based (CWEB) classifier.

3.2.4 Reduced-composite embedding vectors:

The effect of dimensions of word embedding on the accuracy of sentence classification has been extensively studied by Oren Melamud, et al [8]. The choice of dimensionality of the input embedding vector for words significant impact on the convergence and accuracy of the neural networks. An increase in dimensionality doesn't guarantee an improvement in the performance of the neural network [8]. Once improvement in performance of the systems peaks at a particular embedding dimension, no further improvement in the model performance is obtained by further increasing the dimensions of the embedding vector. Since the number of dimensions of the embedding model greatly impact the performance of the system [8], single value decomposition has been used to bring the number of dimensions down to 300. The classifier that uses this embedding is referred to as Reduced Composite model (RC).

3.3 One-Dimensional CNN for Sentiment Classification

Convolutional Neural Networks have achieved state of the art performance in several computer vision tasks [6]. These neural networks have the capability of identifying the local information present in the image such as edges.

3.3.1 Convolutional Neural Networks Basics

Deep neural networks gave acceptable performance for image classification tasks for small images. But as the size of the images grew the number of connections required between different layers of the fully connected layers proliferated to unmanageable numbers. To address is blow-up of connections required even for moderately sized images convolutional neural networks were developed.

Each image is represented as a 3-dimensional array in which each dimension represents the height width and the channel of the image. The three channels correspond to the red, green and blue channel. Each value in a cell corresponds to intensity of a particular pixel in a specific channel. Therefore, an entire pixel in an image is represented by (1*1*3) values. Each convolutional neural network is composed of three basic layers:

a. Convolutional Layer
b. Pooling Layer
c. Fully-connected Layer

Convolutional layer comprises of a filter of weights and a feature map obtained by the activation of a neuron on passing the obtained by convolving the filter with the area of the image (array) present under the filter. A filter is composed of the same number of channels as the number of channels as in the input. This filter is slid horizontally and vertically by the number of positions equal to the stride of the filter. Thus, at each new position, this filter, which is a collection of weights is convolved with the underlying array to get the activation for that position. The collection of the obtained activations is called the feature maps. These feature maps are provided as input to the next layer.

Pooling layer are usually present after each convolutional layer that take as input the feature maps generated and output values that are usually the down sampled value for the input feature map. This acts as a compression technique since the size of the data to be passed to the next layer is usually reduced. Pooling also acts as a technique to prevent over-

fitting of the convolutional neural network to any particular pattern. Max-Pooling and Average pooling are the two most commonly used pooling techniques. Pooling works in a manner similar to convolution. In Max-Pooling, the area of values under an imaginary filter is considered and the maximum of the values is selected for that position. Like the stride during the convolution the maximum value obtained is passed as activation to obtain the feature map to be passed as output to the next layer. Thus, each distinguishing feature is identified by the max-pooling layer and this information is passed on to the next layer.

Fully connected Layer in CNN refers to the flat layers having feed forward connections that are usually provided at the end of the layer. These layers are usually succeeded by a softmax layer for performing multiclass classification. Also, the activation functions for each neuron that comprise the fully connected layer are non-linear in nature.

In this architecture each word is represented as a vector of fixed dimension (d). All sentences are padded with zeros to obtain sentences of equal length (n). The pre-trained word embeddings are used to represent the word. For words not in the vocabulary, a random vector of appropriate embedding size is used. Thus, each sentence is represented concatenation of the vectors for individual words [6].

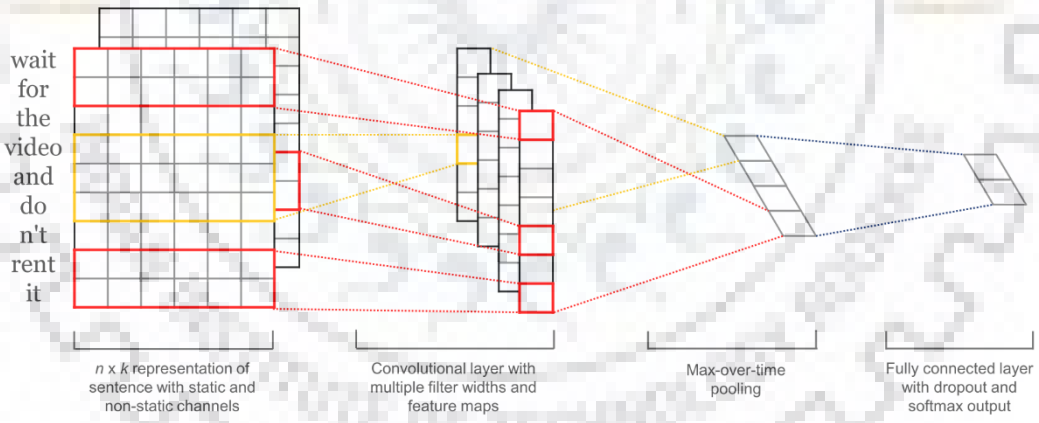


Fig 9 CNN architecture from Yoon Kim [6]

Smaller sentences are padded to obtain a vector of uniform length and dimensions that represent a single sentence. The activation function used in ReLU since it has been proven to greatly accelerate the speed at which convergence is achieved compared to vanilla

sigmoid or tanh activation function, while also being computationally cheaper to compute [7].

Convolution layer performs convolution of words using filter of varying lengths. This architecture uses filters of lengths 3, 4 and 5, representing context windows of respective sizes. Max-pooling of the feature map is done, and the resultant distinctive vector is determined out of this window. This is then concatenated into a single large feature vector. Dropout and L2-Regularizations are used to randomly deactivate a set percent to neuron activations, to avoid over fitting to the training data, particularly if the dataset has limited vocabulary [6].

4 Experimental Results and Analysis

4.1 Opinion Target Tagging

4.1.1 Dataset Description

The POS tagger is trained on a corpus consisting of annotated sentences tagged with IOB tags obtained after combining Conll-2003 and GMB corpus for entity classification consisting of 13,54,149 tagged words.

4.1.2 Results

The Bi-LSTM CRF tagger when trained and evaluated on the training dataset obtained an F1-score of 0.932, on training for 20 epochs with 10-fold cross validation. This model when applied to the "Fine Food" and "Movie Review" dataset gave the following results

Table 1 IOB-tag distribution

Tag	Fine Food	IMDB Movie Review	Yelp Reviews	Twitter dataset
I	38711	17362	36895	863
O	9986277	182156	478267	2560
B	148290	19320	28937	866

Based on the number of B-tags per review, the entire dataset is segmented into three classes, namely zero target, one target and multi-target classes. This dataset is then used to individually train separate classifiers.

4.2 Sentiment Classification Datasets

4.2.1 Fine Food dataset

The dataset consists of 586454 records, each having 10 attributes. Only 'Review Heading', 'Review Text' and 'Score' are selected. Records with null, missing or incomplete values are removed. The dataset is then shuffled, and top 110000 records are selected. The

'Review Heading' and 'Review Text' are concatenated to obtain the complete 'Review'. The Score attribute for each review ranges from 1("highly negative") to 5("highly positive"). The distribution of these classes over the entire dataset is as follows:

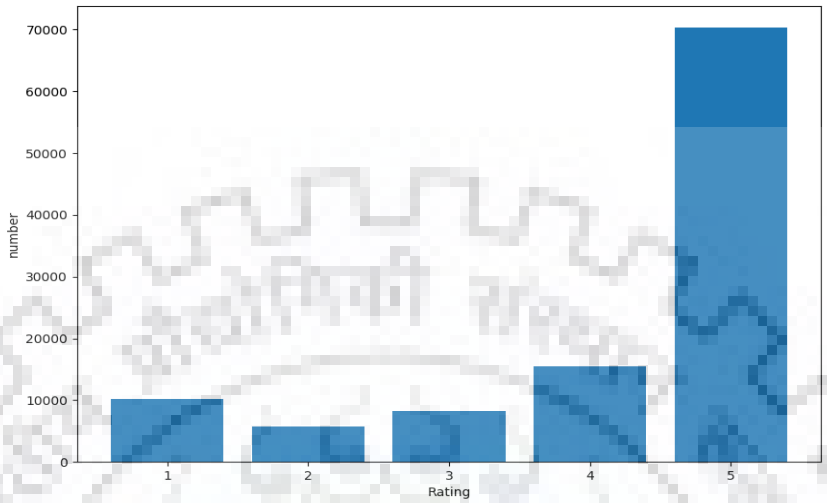


Fig 10 Sentiment Score distribution for processed Fine Food dataset

The number of unique words in the vocabulary of the corpus is 64793. The maximum length of any review in the dataset is 3789.

4.2.2 IMDB Movie Review Dataset:

This dataset consists of 10662 reviews. In this 5331 are positive and the rest 5331 are negative reviews. Initially all these reviews are pre-processed to remove punctuations. The number of unique words in the vocabulary is 18757 after pre-processing. The maximum length of any sentence is 56.

4.2.3 Twitter Dataset

This dataset consists of tweets collected and annotated manually by Sanders Lab. These tweets are collected mainly from 2007 to 2011. These tweets are filtered to include only those tweets that concern four companies. Each of these tweets have been assigned "Positive", "Negative", "Neutral" or "Irrelevant". Those tweets are labelled positive that indicate the opinion regarding the company spoken about is positive. Tweets showing negative disposition are labelled as negative. Finally tweets that are purely informative or

descriptive in nature with no clear indication of opinion or sentiment are classified as neutral.

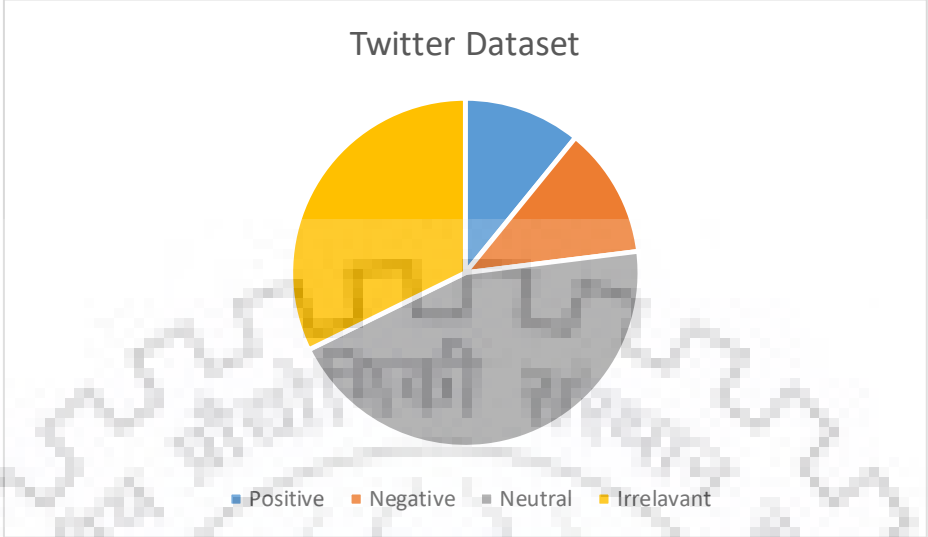


Fig 11 Class distribution for Twitter dataset

For the evaluation "positive" and "negative" labelled are selected and the various described models are trained on this subset of the entire dataset.

4.2.4 Yelp Reviews

This dataset consists of reviews made about several local businesses in USA. This dataset is available in Json format. Each record of this dataset consists of several attributes. For the purpose of the evaluation only the "review-text" and "rating" attributes are selected. Rating attributes column consists of numerical values ranging from 1 to 5 with increments of 0.5. These indicate the degree of positivity or the sentiment regarding the product or business. The sentiment for records having rating values between 0 and 2 are considered to be negative. Similarly, for ratings ranging from 2 to 3 and 3 to 5 are considered to be neutral and positive respectively. Since on online platforms mostly customers choose to review if the product exceeded their expectations, or failed to meet their expectations, the number of reviews falling under each class varies vastly. This results in comparatively large number of reviews for positively viewed products and businesses. This results in a problem with dataset called "Class Imbalance" problem. Any result obtained by any classifier trained on this dataset would be unreliable measure of the performance of the classifier. To overcome this limitation, the number of records selected for each class is the same. For evaluation of

different models only 6000 records are selected, with 2000 records for each class, namely 2000 for each of “positive”, “neutral” and “negative” classes.

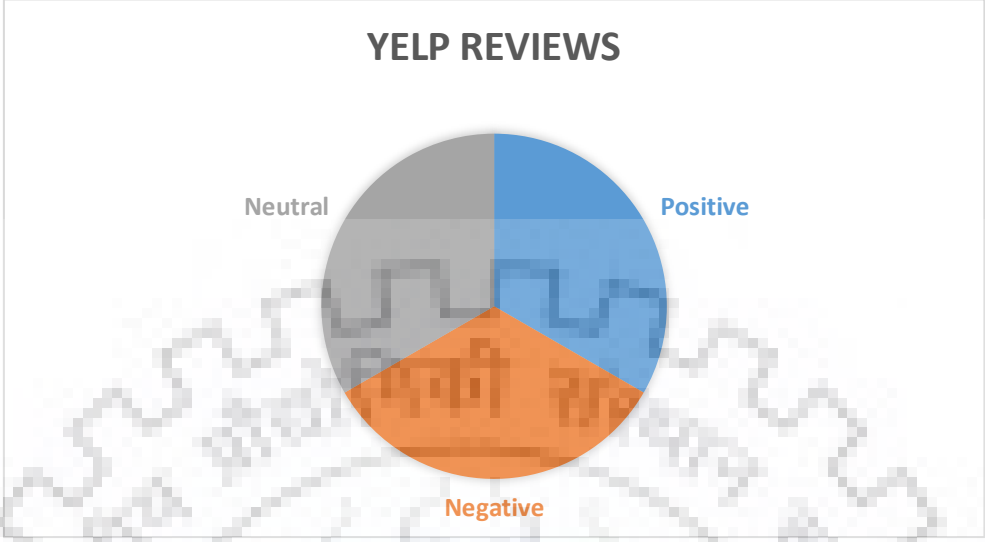


Fig 12 Class distribution for processed Yelp Reviews

The train, test and Dev splits of each dataset to be provided as input to each model:

Table 2 Train-Test-Dev splits for each dataset

Target	Train	Test	Dev
Fine Food Data			
complete	99988	9012	1000
0-target	45271	4080	453
1-target	25269	2277	253
n-target	29448	2655	294
IMDB Movie Review			
complete	8529	1706	427
0-target	1518	304	76
1-target	2648	529	133
n-target	4363	872	219
Twitter Dataset			
complete	903	150	76
0-target	454	75	38
1-target	241	42	18
n-target	209	37	15
Yelp Reviews			
complete	4800	1000	200
0-target	3203	602	198
1-target	798	145	53
n-target	801	147	54

4.3 Results

4.3.1 Sentiment Classification

The performances of the various models on the test datasets are tabulated below:

Table 3 Experimental Results

Model	Embedding Size	Fine food dataset accuracy	Movie Review dataset accuracy	Twitter Dataset Accuracy	Yelp Reviews Accuracy
Existing Approaches					
CNN	300	76.82	64.4	78.12	61.30
word2vec embedded model	300	77.96	76.8	79.05	81.21
Proposed Approaches					
GEW model	300	77.67	73.1	77.63	80.05
FTE model	300	77.83	74.0	78.61	79.97
CEWB model	600	77.97	78.1	83.77	84.48
RCE model	300	**78.33**	78.3	83.81	**85.68**
TBS-RCE model	300	76.9	**78.4**	**83.89**	84.49

Each model is trained to run to up to 10 epochs with early stopping and the model having the best Dev accuracy is used to evaluate the test dataset. It is observed that for most datasets the best accuracy is observed in either RC embedded model or Segmented RC model. It is also evident that majority of the proposed models outperform the State-of-the-Art models by an acceptable margin.

4.3.2 Effect of Dimensionality on the Accuracy

The effect of dimensionality on the accuracy of trained models is studied by evaluating the same model with vectors of different dimensions. The obtained results are plotted on a graph and shown in Fig. 13

It is observed that there is an upward trend in accuracy for all datasets as the dimensions of the word embedding is increased up to 300. But any further increase in dimensions results in deteriorating performance of the trained models.

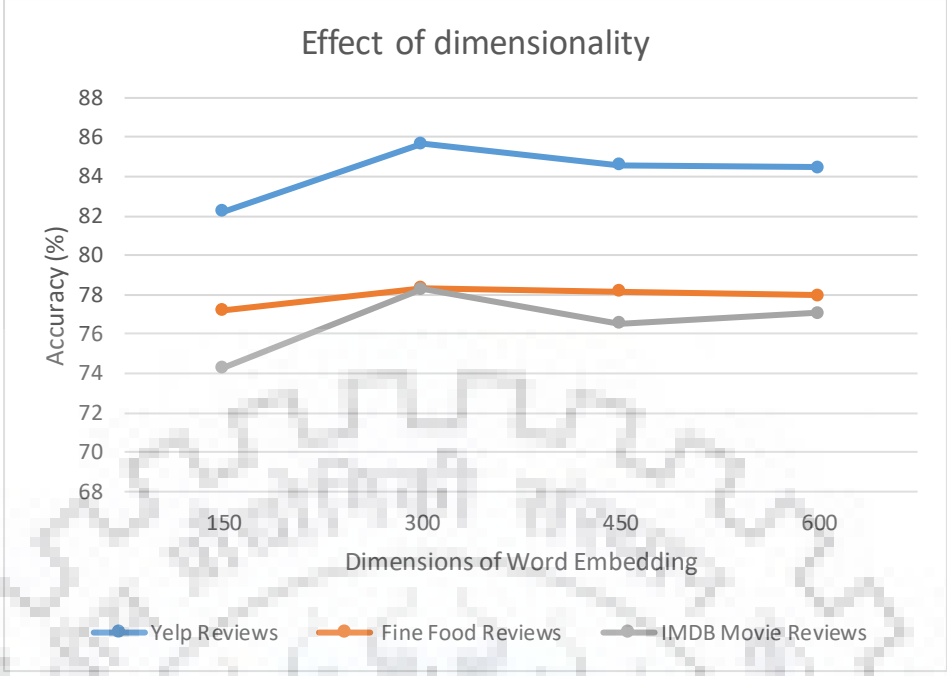


Fig 13 Effect of dimensionality on the performance of classifiers

It is evident that concatenating word embeddings obtained by training different models gives better results compared to when the respective embeddings are used independently. Superior performance in terms of accuracy is obtained by using dimensionality reduction on the concatenated embedding vectors. It also observed that faster convergence to the best accuracy is obtained on the (reduced composite model)

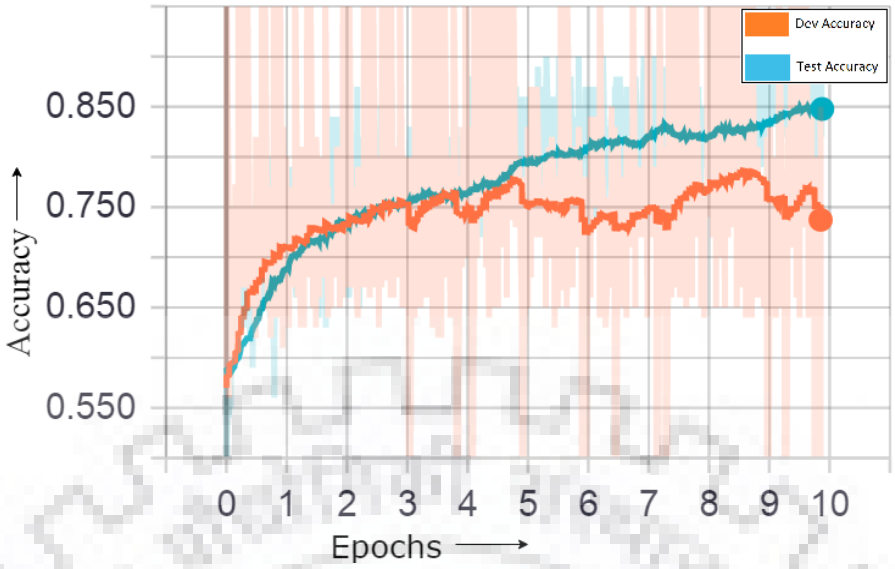


Fig 14 Accuracy curves for Composite model on Fine Food dataset

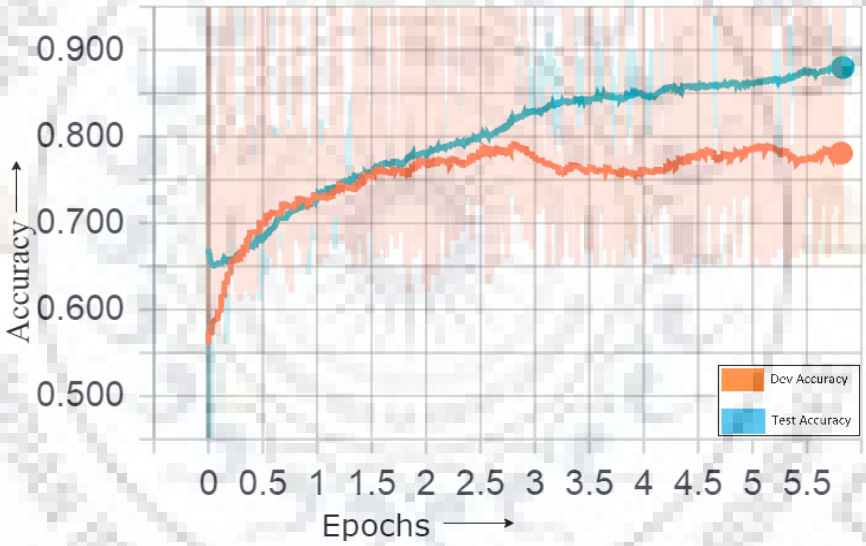


Fig 15 Accuracy curves for Reduced-Composite model on Fine food dataset

The entire dataset is partitioned into mini batches and then mini-batch gradient descent with Adam optimization is used to train the deep neural network model. A smoothening factor of 0.98 is used to average the y-axis values before displaying. Check pointing is done to store the model with best Dev accuracy with is then later used for evaluating the corresponding test set.

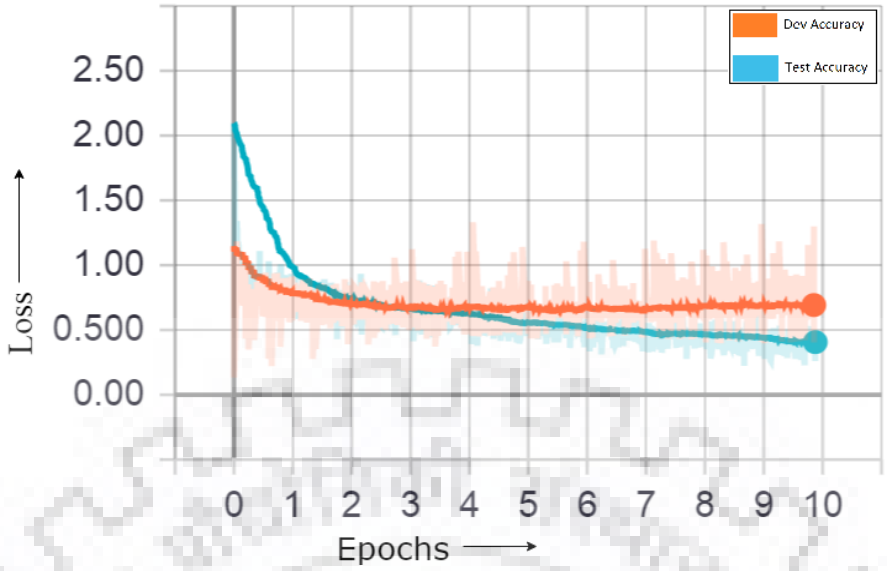


Fig 16 Loss curves of composite model

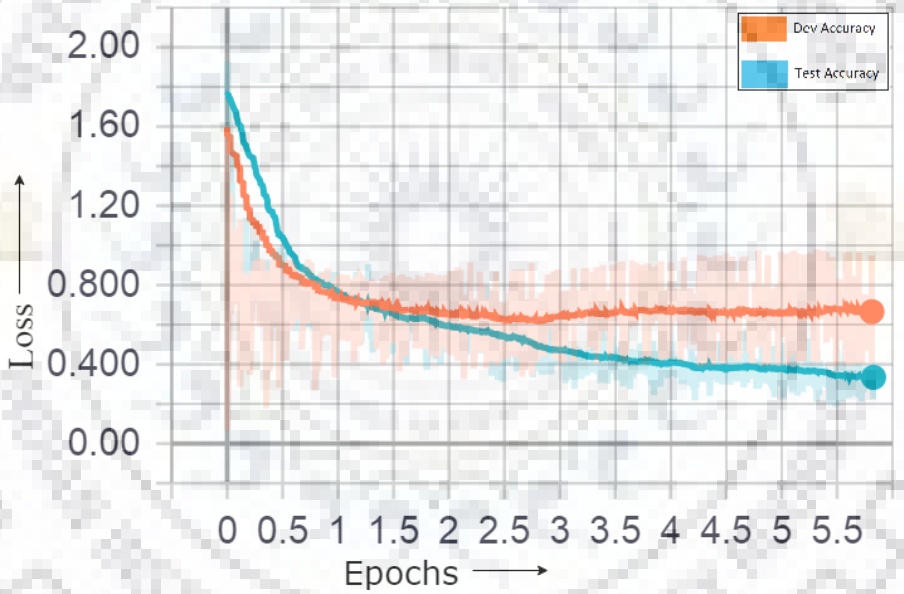


Fig 17 Loss curves of Reduced composite model

From the figures 4.5, 4.6, 4.7 and 4.8 it is evident that convergence to the best model (based on Dev accuracy) is obtained much quicker in the Reduced-Composite model compared to composite model.

5 Conclusion

The accuracy of the proposed models obtained when trained on the “Fine Food”, “IMDB movie reviews”,” Twitter” and “Yelp Review” datasets is comparable to the state of the art techniques used for sentiment analysis. The use different word embeddings has been explored. Fasttext being one the newest word-embedding generating technique has not yet been explored completely and hence provides whole window of research to exploit this newly developed technique to effectively improve upon the work done before it.

The use of complementary word embeddings for pre-trained embeddings has shown to provide better results. The use of dimensionality reduction on the composite word embeddings has shown to further improve the results by a small margin on the existing model. Thus, the promising results shown by the proposed system provides a base system which can be refined and improved upon further.

References

1. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean-"Efficient Estimation of Word Representations in Vector Space". In Proceedings of Workshop at ICLR, 2013
2. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality". In Proceedings of NIPS, 2013
3. Frederic Morin and Yoshua Bengio- "Hierarchical probabilistic neural network language model". In Proceedings of the international workshop on artificial intelligence and statistics, pages 246–252, 2005.
4. Jeffrey Pennington, Richard Socher, and Christopher D. Manning.: Global Vectors for Word Representation- Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, October 25-29, 2014, Doha, Qatar
5. Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov -Bag of Tricks for Efficient Text Classification, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 427–431, Valencia, Spain, April 3-7, 2017
6. Yoon Kim -Convolutional Neural Networks for Sentence Classification Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, October 25-29, 2014, Doha, Qatar
7. M. D. Zeiler; M. Ranzato; R. Monga; M. Mao; K. Yang; Q. V. Le; P. Nguyen; A. Senior; V. Vanhoucke; J. Dean; G. E. Hinton- On rectified linear units for speech processing-IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013
8. Oren Melamud, David McClosky, Siddharth Patwardhan, Mohit Bansal-The Role of Context Types and Dimensionality in Learning Word Embeddings – Proceedings of NAACL-HLT 2016, pages 1030–1040, San Diego, California, June 12-17, 2016.
9. Alan Ritter, Sam Clark, Mausam and Oren Etzioni - Named Entity Recognition in Tweets: An Experimental Study-Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1524–1534, Edinburgh, Scotland, UK, July 27–31, 2011

10. Lev Ratinov, Dan Roth-Design Challenges and Misconceptions in Named Entity Recognition-In Proceedings of the Thirteenth Conference on Computational Natural Language Learning, pp. 147-155 Association for Computational Linguistics (ACL), June 2009.
11. X. Ma, E. Hovy- "End-to-end sequence labelling via bi-directional LSTM-CNN-CRF", arXiv preprint arXiv:1603.01354, 2016.
12. Tao Chen, Ruifeng Xua,, Yulan Hec, Xuan Wanga - Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN, Expert Systems With Applications 72 (2017)pages 221–230, Apr 2017
13. Zhao W.X. et al. (2011) Comparing Twitter and Traditional Media Using Topic Models. In: Clough P. et al. (eds) Advances in Information Retrieval. ECIR 2011. Lecture Notes in Computer Science, vol 6611. Springer, Berlin, Heidelberg
14. Guillaume Lample, Miguel Ballesteros, Subramanian, Kazuya Kawakami, Chris Dyer-Neural Architectures for Named Entity Recognition- Proceedings of NAACL-HLT 2016, pages 260–270, San Diego, California, June 12-17, 2016.
15. P. Waila, Marisha, V.K. , M.K. Singh - Evaluating Machine Learning and Unsupervised Semantic Orientation approaches for sentiment analysis of textual reviews-2012 IEEE International Conference on Computational Intelligence and Computing Research
16. Zhifeng Hao; Ruichu Cai; Yiyang Yang; Wen Wen; Lixin Liang-A Dynamic Conditional Random Field Based Framework for Sentence-Level Sentiment Analysis of Chinese Microblog-2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)
17. Jun Li, Maosong Sun-Experimental Study on Sentiment Classification of Chinese Review using Machine Learning Techniques-2017 International Conference on Natural Language Processing and Knowledge Engineering
18. Chih-Hua Tai, Zheng-Han Tan, Yue-Shan Chang-Systematical Approach for Detecting the Intention and Intensity of Feelings on Social Network-IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 20, NO. 4, JULY 2016
19. Shulong Tan, Yang Li, Huan Sun, Ziyu Guan, Xifeng Yan, Jiajun Bu, Chun Chen, Xiaofei He, "Interpreting the Public Sentiment Variations on Twitter", IEEE Transactions on Knowledge and Data Engineering (Volume: 26, Issue: 5, May 2014)

20. D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” Journal of Machine Learning Research, vol. 3, no. 5, pp. 993-1022, 2003.
21. D. M. Blei, “Probabilistic topic models,” Communications of thACM, vol. 55, no. 4, pp. 77–84, 2012.

List of Publications

Bharath T.S., Durga Toshniwal, Amit Agarwal – "Composite Word-Embeddings: Augmenting Deep Learning Classifiers with Pre-Trained Word Embeddings to improve classification performance for Sentiment Analysis", 20th International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2018) - *Communicated*