Distributed Semantics: Vector Models for NLP tasks in Hindi

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

Master of Technology

by
Pranjal Singh
Roll No.: 10327511

under the guidance of
Dr. Amitabha Mukerjee

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur

April, 2015

CERTIFICATE

It is certified that the work contained in this thesis entitled "*Distributed Semantics: Vector Models for NLP tasks in Hindi*", by *Pranjal Singh (Roll No. 10327511)*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr. Amitabha Mukerjee)
Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur
Kanpur - 208016

April, 2015

Abstract

In recent years, distributional semantics or word vector models have been proposed to capture both the syntactic and semantic similarity between words. Since these can be obtained in an unsupervised manner, they are of interest for under-resourced languages such as Hindi. We test the efficacy of such an approach for Hindi, first by a subjective overview which shows that a reasonable measure of word similarity seems to be captured quite easily. We then apply it to the sentiment analysis for two small Hindi databases from earlier work.

In order to handle larger strings from the word vectors, several methods - additive, multiplicative, or tensor neural models, have been proposed. Here we find that even the simplest - an additive average, results in an impressive accuracy gain on state of the art by 10% (from 80%) for two review datasets. The results suggest that it may be worthwhile to explore such methods further for Indian languages.

Dedicated to my Parents, Brother and Friends for their love and support

Acknowledgement

I would like to take this opportunity to thank the people who have helped me in preparation of this thesis.

First of all, I would like to express my sincere gratitude towards my thesis supervisor, Dr. Amitabha Mukerjee, for his constant support and encouragement. I am grateful for his patient guidance and advice in giving a proper direction to my efforts. I am grateful to him for providing me ample freedom to choose a topic of my interest and deciding how to pursue it.

I thank the Department of Computer Science and Engineering, IIT Kanpur, for providing the necessary infrastructure and a congenial environment for research work. I thank Tomas Mikolov for making the code of word2vec and sentence2vec publicly available.

I am also thankful to all my friends and specially wingmates who have made my stay at IIT Kanpur most enjoyable and who in general gave me strength and hope whenever I required. Last, but not the least, I thank my parents for always being there for me.

Pranjal Singh

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over a period of nearly a millenium, Indian grammarians have been trying to find whether sentence meaning accrues by combining word meanings, or whether words gain their meanings based on the context they appear in [11]. The former position, that meaning is *compositional*, has been associated with the fregean enterprise of semantics, whereas recent models, building on large corpora of text (and associated multimedia) a large degree of success has accrued to models that attempt to model word meaning based on their linguistic context (e.g. [8]). The latter line has resulted in strong improvements in several NLP tasks using word vectors [5, 24, 16, 23]. The advantage of these approaches is that they can capture both the syntactic and the semantic similarity between words in terms of their projections onto a high-dimensional vector space; further, it seems that one can tune the privileging of syntax over semantics by using local as opposed to large contexts [6].

For resource-poor languages, these approaches have the added lure that many of these methods are completely unsupervised and work directly with large raw text corpora, thus avoiding contentious issues such as deciding on a POS-tagset, or expensive human annotated resources such as treebanks. For Indian languages which are highly inflected, stemming or identifying the lemma is another problem which such models can overcome, provided the corpus is large enough. Nonetheless, this approach remains under-explored for Indian languages. At the same time, it must be noted that many approaches seek to improve their performance by combining

POS-tags and even parse tree structures into the models for higher accuracies in specific tasks [23].

Vector models for individual words are obtained via distributional learning, the mechanisms for which varies from document-term matrix factorization [8], various forms of deep learning [5, 24, 23], optimizing models to explain co-occurrence constraints [16, 21],etc. Once the word vectors have been assigned, similarity between words can be captured via cosine distances.

[4] [22] One problem in this approach is that of combining the word vectors into larger phrases. In past work, inverse-similarity weighted averaging appears to work to some extent even for complex tasks such as essay grading [9], but multiplicative models (based on a reducing the tensor products of the vectors) appears to correlate better with human judgements [18, 23]. Another complexity in composition is that composing words across phrasal boundaries are less meaningful than composing them within a phrase - this has led to models that evaluate the nodes of a parse tree, so that only coherent phrases are evaluated [23]. The results reported here, are based on applying the Skip-Gram model [13] to Hindi.

1.0.1 Sentiment Analysis

In order to evaluate the efficacy of the model, we apply it to the task of sentiment analysis. Here the problem is that of identifying the polarity of sentences (Liu et al. 2012); for example:

- Positive: रामू ने कहानी की रफ़्तार कहीं थमने नहीं दी [Ramu didn't allow the pace of the story to subside]
- Negative: पर्दे पर दिखाया जा रहा खौफ़ सिनेमाघर मे नहीं पसर पाता [The horror shown on the screen didn't reach the theater]

This is a problem that has attracted reasonable attention in Hindi (see section 2), since most sentiment analysis is oriented towards semantics, and one may bypass the syntactic processing which remains poor for Hindi. Methods that have been used are largely based on assigning a sentiment effect for individual words, and then combining these in some manner to come up with an overall sentiment for the document. Such methods ignore word order and have beencriticized since the import of a sentence can change completely simply by re-arranging the words, though the sentiment evaluation remains the same. Several groups have attempted to improve the situation by modeling the composition of words into larger contexts [10, 23, 7, 2].

However, most of the work on sentiment analysis in Hindi has not attempted to form richer compositional analyses. For the type of corpora used here, the best results, obtained by combining a sentiment lexicon with hand-crafted rules (e.g. modeling negation and "but" phrases), reach an accuracy of 80% [19].

In this work, we first learn a distributional word vector model based on the wikipedia Hindi corpus as well as the sentiment corpus, and then we use this to discern the polarities on the existing corpora of movie and product reviews. To our own surprise, we find that even a simple additive composition model improves the state of the art in this task significantly (a gain of nearly 10%). When used for the much better-researched, larger datasets of English the system does respectably, but well behind the very best models that attempt more complex composition models. So the question arises as to whether the very significant gains in Hindi are due to some quirk in the dataset, or could it be that Hindi word vectors are particularly informative, e.g. owing to more highly inflected nature of its surface forms. Also, if the results are not corpus-specific, it also raises the possibility that word vector methods may result in significant gains in other similar problems for Hindi.

For example, the sentences below give a brief idea of what positive and negative sentiment means.

- Positive: यह फिल्म अच्छी है
- Negative: यह समान बहुत खराब है

Majority of the existing work in this field is in English (Pang and Lee, 2008).

[0]en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers

Our model uses word vector averaging to build document vectors which constitute a feature set for our SVM classier. We have used skip-gram model (Mikolov et al., 2013b) for building word vectors because deep network model of Collobert et al.(2008) takes too much time for training. Also the word embeddings obtained by them are not as good as those obtained by Mikolov et al.(2013b). On NER task, skip-gram obtained F1-score of 93.1 while CW obtained F1-score of 92.2. We have also experimented with tf-idf for building our feature set. Our model shows slight improvement in performance if we filter out certain corpus based stop words. This is a first attempt to use word vectors for sentiment analysis in Hindi. Word vectors capture both semantic and syntactic information for a given corpus. Words which are semantically and syntactically related tend to be closer in high-dimensional space.

1.1 The Problem

The mood of a music enthusiast can be determined by the songs she has heard recently. The mood usually changes gradually over time and can be modeled by the genre of the songs. Mood once modeled can be compared with those of other similar users and thus help conclude the preference of a song after a set of given songs (which determine the mood) as heard by other "similar" users.

1.2 Contributions of this Thesis

There are two main contributions of this thesis. The first is to determine the rolling current mood of a user based on her recently played tracks. The challenge here is to optimize the monitored rolling time duration, to come up with a method to compare two set of moods a set of weights for each of the contributing factor, i.e., mood, user's preferences and collaborative filtering to obtain a confidence score for each recommendation at any given instant. The second is to build a tool that not only shows the recommendations but also lets you customize the determining weights and parameters of the recommendation engine.

1.3 Organization of this Thesis

The rest of this thesis is organized as follows. Chapter 2 presents related work done. Chapter 3 discusses the various background work that one must be acquainted with in order to understand the work presented. Chapter 4 discusses the data sources and APIs used to set up a sample infrastructure of the recommendation system. Chapter 5 then discusses in detail the implemented methods and algorithms. Chapter 6 presents a summary of the results that was achieved and also talks about what can be done further.

Chapter 2

Related Work

Sentiment analysis is a well-known research area in NLP today (see reviews in [20] and Pang et al. (2008), and also challenge in SemEval-2014). Early work on movie review sentiments achieved an accuracy of 87.2% (Pang et al. 2004) on a dataset that discarded objective sentences and used text categorization techniques on the subjective sentences. Le and Mikolov (2014) use word vector models and obtain 92.6% accuracy on IMDB movie review dataset. They used distributed bag-of-words model, which they call as *paragraph vector*. More difficult challenges involve short texts with nonstandard vocabularies,as in twitter. Here, some authors focus on building extensive feature sets (e.g. Mohammad et al.(2013); F-score 89.14).
Wang et al. (2014) propose a word vector neural-network model, which takes both sentiment and semantic information into account. This word vector expression model learns word semantics and sentiment at the same time as well as fuses unsupervised contextual information and sentence level supervised labels.

There has been limited work on sentiment analysis in Hindi – see review in [12], who surveys sentiment analysis in non-English languages). Joshi et al. (2010) compared three approaches: In-language sentiment analysis, Machine Translation and Resource Based Sentiment Analysis. By using WordNet linking, words in English SentiWordNet were replaced by equivalent Hindi words to get H-SWN. The final accuracy achieved by them is 78.1%.

[1] traversed the WordNet ontology to antonyms and synonyms to identify po-

larity shifts in the word space. Further improvements were achieved by using a partial stemmer (there is no good stemmer / morphological analyzer for Hindi), and focusing on adjective/adverbs (45 + 75 seed words given to the system); their final accuracy was 79.0% for the product review dataset. Mukherjee et al. (2012) presented the inclusion of discourse markers in a bag-of-words model and how it improved the sentiment classification accuracy by 2-4%. Mittal et al. (2013) incorporate hand-coded rules dealing with negation and discourse relations and extend the HSWN lexicon with more opinion words. Their algorithm achieves 80.2% accuracy on classification of movie reviews on a separate dataset.

[14] is a work The model has developed into More details in 3.3

Chapter 3

Background

The algorithms and data structures used in this thesis have been introduced and discussed below.

3.1 Word-Vectors by using Skip-Gram

Mikolov et al. (2013b) proposed two neural network models for building word vectors from large unlabelled corpora; Continuous Bag of Words(CBOW) and Skip-Gram. In the CBOW model, the context is the input, and one tries to learn a vector for the central word; in Skip grams, the input is the target word and one tries to guess the set of contexts. The Skip gram was found to perform better on smaller corpora, and here we have focused on this model for building our word vectors. The model uses each current word as an input to a log-linear classifier with continuous projection layer, and predict words within a certain range before and after the current word. The objective is to maximize the probability of the context given a word within a language model:

$$p(c|w;\theta) = \frac{\exp^{v_c \cdot v_w}}{\sum_{c' \in C} \exp^{v_c \cdot v_w}}$$

where v_c and $v_w \in R^d$ are vector representations for context c and word w respectively. C is the set of all available contexts. The parameters θ are $v_c i$, $v_w i$ for $w \in V$,

$c \in C$, $i \in 1, ..., d$ (a total of $|C| \times |V| \times d$ parameters).

The weights between the input layer and the output layer can be represented by a $V \times N$ matrix **W**. Each row of **W** is the N-dimension vector representation v_w of the associated word of the input layer.Given a word, assuming $x_k = 1$ and $x_{k'} = 0$ for $k' \neq k$, then

$$h = x^T W = W_{(k,.)} := v_{w_I}$$

which is essentially copying the k-th row of **W** to **h**. v_{w_I} is the vector representation of the input word w_I.

From the hidden layer to the output layer, there is a different weight matrix **W'**=$\{w'_{ij}\}$ which is a $N \times V$ matrix. Then we can use soft-max, a log-linear classification model, to obtain the posterior distribution of words, which is a multinomial distribution.
On the output layer, instead of outputing one multinomial distribution, we are outputing C multinomial distributions. Each output is computed using the same hidden→output matrix.

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{exp(u_{c,j})}{\sum_{j'=1}^{V} exp(u_{j'})}$$

where $w_{c,j}$ is the j-th word on the c-th panel of the output layer; $w_{O,c}$ is the actual c-th word in the output context words; w_I is the only input word; $y_{c,j}$ is the output of the j-th node on the c-th panel of the output layer; $u_{c,j}$ is the net input of the j-th node on the c-th panel of the output layer. Because the output layer panels share the same weights, thus

$$u_{c,j} = u_j = v{`}^{T}_{w_j}.h, \text{ for } c = 1, 2..., C$$

where v'_{w_j} is the output vector of the j-th word in the vocabulary, w_j, and also v'_{w_j} is taken from a column of the hidden→output weight matrix, **W'** .

The loss function is

$$\text{E} = \text{-log}\, p(w_{O,1}, w_{O,2}, ..., w_{O,C}|w_I)$$

$$=\text{-log}\prod_{c=1}^{C}\frac{exp(u_{c,j_c^*})}{\sum_{j'=1}^{V}exp(u_{j'})}$$

$$=\text{-}\sum_{c=1}^{C}u_{j_c^*}+C.\log\sum_{j'=1}^{V}exp(u_{j'})$$

where j_c^* is the index of the actual c-th output context word in the vocabulary.

After taking the necessary derivatives, the update equation for the hidden→output weight matrix, **W'**,

$$w'ij^{(new)} = w'ij^{(old)} - \eta.EI_j.h_i$$

or,

$$v'w_j{}^{(new)} = v'w_j{}^{(old)} - \eta.EI_j.\mathbf{h} \text{ for } j = 1, 2, ..., V$$

The update equation for the input→hidden weight matrix, **W**,

$$vw_I{}^{(new)} = vw_I{}^{(old)} - \eta.EH$$

where EH is a N-dimensional vector. Its each component is defined as

$$EH_i = \sum_{j=1}^{V}EI_j.w'_{ij}$$

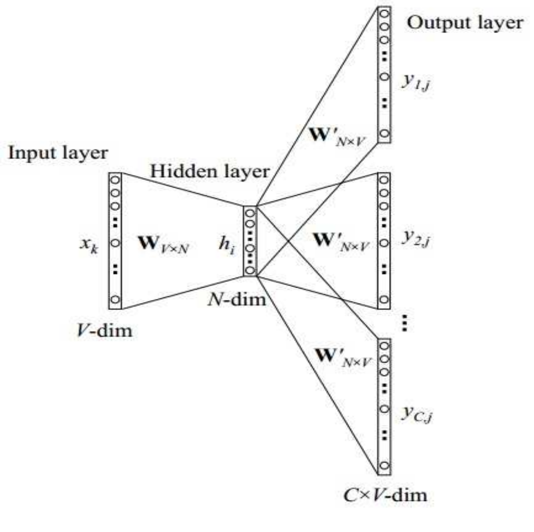


Figure 3.1: Skip Gram Model(Figure from Rong (2014))

3.1.1 tf-idf

Let $D = d_1, d_2, d_3....d_N$ be N documents under study and $T = t_1, t_2, t_3, ...t_M$ be the M unique terms in these documents, then each document can be represented as a M-dimensional vector:

$t_d = \{tf_1, tf_2, tf_3, ...tf_M\}$

$tf - idf$ weights discounts the frequency of terms based on their importance to a particular document in the entire document set collection under consideration. This is done as follows:

$$tfidf(d,t) = tf(d,t) \times \log(\frac{|D|}{df(t)})$$

Here $df(t)$ is the number of documents in which term t appears.

3.1.2 Vector Averaging for phrases

As an output of the word vector learning, we now have a n-dimensional vector representation for each word in the Hindi corpus. Now we need to assign features for sentences and paragraphs taken from the sentiment dataset (training and test). Mikolov et al. (2013b) and Levy et al. (2014) show that many relational similarities can be recovered by means of vector arithmetic in the embedded space. Thus, additive models are useful, though others have claimed that multiplicative models correlate better with human judgments [18, 23]. In this work, we have retained teh simplicity of vector averaging to model larger chunks of discourse. This models the sentence/document in the same high dimensional space.

A preprocessing step involved removing some words that appear at very high or very low frequencies in the corpus. Our model was trained on the Hindi Wikipedia dump to create vector representations for words. The previous two vectors were concatenated to create another feature set for training purpose.

Algorithm

1. Input the Hindi text corpus

2. Train skip-gram model to obtain word vector representation

3. Given a sentiment training set, obtain average vector data for each sentence/document
4. Obtain tf-idf vector for each sentence/document in the corpus
5. Concatenate vectors of step 3 and step 4 to obtain a feature set for a training instance
6. Train linear SVM with m-fold cross validation to create a classifier (here m=20)

3.2 Document Vectors

This distributed representation of sentences and documents [10] modifies wors2vec (Skip-Gram) algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. The algorithm represents each document by a dense vector which is later trained and tuned to predict words in the sentencedocument.
In Paragraph Vector framework, every paragraph is mapped to a unique vector and id, represented by a matrix D, which is a column matrix. Every word is mapped to a unique vector and word vectors are conactenated or averaged to predict the context, i.e., the next word.
The change in this framework is that the h(in Skip-Gram model's equation) is now constructed in a different way. It is now constructed using both W and D.
The contexts are fixed-length and sampled from a sliding window over the paragraph. The paragraph vector is shared across all contexts generated from the same paragraph but not across paragraphs. The word vector matrix W, however, is shared across paragraphs. i.e., the vector for "good" is the same for all paragraphs.

The paragraph vectors and word vectors are trained using stochastic gradient descent and the gradient is obtained via backpropagation. At every step of stochastic gradient descent, one can sample a fixed-length context from a random paragraph, compute the error gradient from the network in Figure 3.3 and use the gradient to update the parameters in our model. At prediction time, one needs to perform an inference step to compute the paragraph vector for a new paragraph. This is also obtained by gradient descent. In this step, the parameters for the rest of the model, the word vectors W and the softmax weights, are fixed.

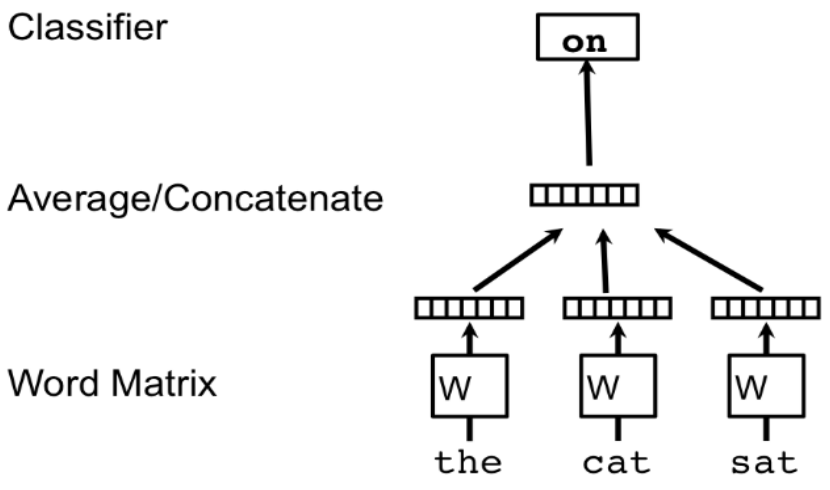


Figure 3.2: Framework for learning word vectors(Figure from Le (2014)).

In Figure 3.2, context of three words (”the”, ”cat” and ”sat”) is used to predict the fourth word (”on”). The input words are mapped to columns of the matrix W to predict the output word(Figure from Le (2014)).

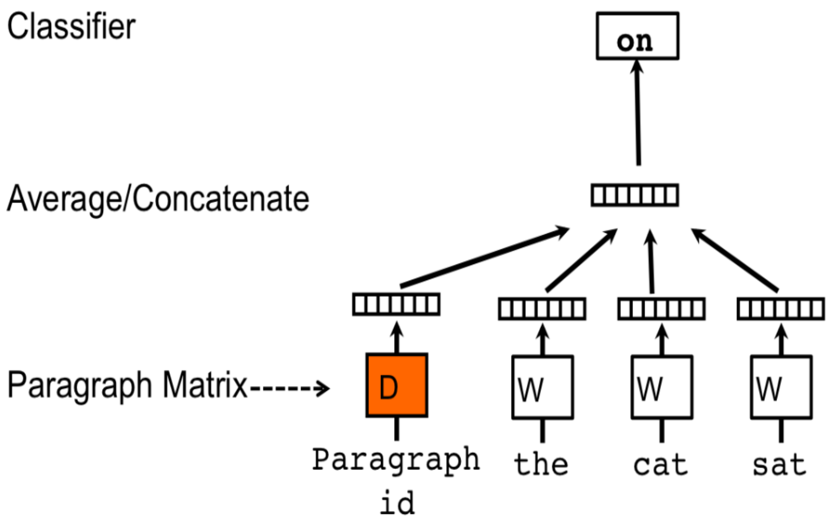


Figure 3.3: Framework for learning paragraph vectors(Figure from Le (2014)).

In Figure 3.3, the only change is the additional paragraph token that is mapped to a vector via matrix D. In this model, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph.

The advantage of using paragraph vectors is that they inherit the propery of word vectors, i.e., the semantics of the words. In addition, they also take into consideration a small context around each word which is in close resemblance to the n-gram model with a large n. This property is crucial because the n-gram model preserves a lot of information of the sentenceparagraph, which includes the word order also. This model also performs better than the Bag-of-Words model which would create a very high-dimensional representation that has very poor generalization.

3.3 Recurrent Neural Network

The primary feature of a Recurrent Network is that it contains atleast one feedback connection which captures dynamic temporal behavior and allows it to learn sequences. They have application in tasks such as prediction of a word given the context, perform sequence recognition/reproduction.
Recurrent Neural Networks have many different forms. One of them is a Fully Recurrent Network, a network of neuron-like units, each with a directed connection to every other unit.

The architecture of a simple RNN consists of an input layer x, a hidden layer s(also known as context layer or network state) and an output layer y. Since the training is time dependent, we will denote input x as $x(t)$, output y as $y(t)$ and state s as $s(t)$. Input vector $x(t)$ is a concatentaion of vector w representing current word, and output from neurons in context layer s at time $t-1$(We can also include output from context layer before $t-1$ as well). The equations of each layer are described

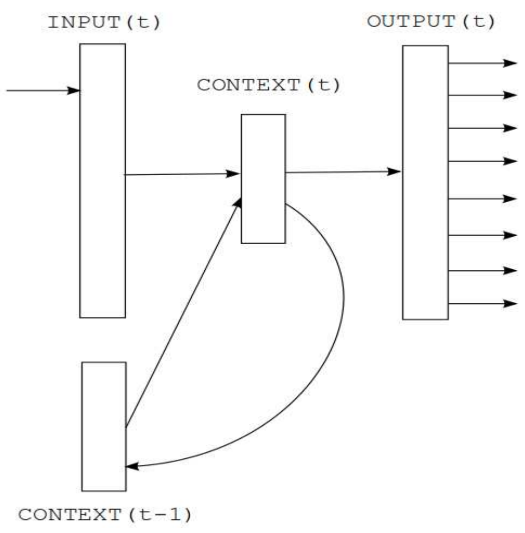


Figure 3.4: Simple Recurrent Neural Network(Figure from [14]).

below:

$$x(t) = w(t) + s(t-1) \tag{3.1}$$

$$s_j(t) = f\left(\sum_i x_i(t)u_{ji}\right) \tag{3.2}$$

$$y_k(t) = g\left(\sum_j s_j(t)v_{kj}\right) \tag{3.3}$$

where $f(z)$ is sigmoid function for activation and $g(z)$ is a softmax function for output prediction:

$$f(z) = \frac{1}{1+\mathrm{e}^{-z}} \tag{3.4}$$

$$g(z_m) = \frac{\mathrm{e}^{z_m}}{\sum_k \mathrm{e}^{z_k}} \tag{3.5}$$

[14] claim that size of hidden layer reflects amount of training data with smaller size leading to less number of layers. Weights are initialized to random small values and updated using gradient descent. Output layer $y(t)$ represents probability distribution of next word given previous word $w(t)$ and context $s(t-1)$. The objective function is:

$$error(t) = actual(t) - y(t) \tag{3.6}$$

where *actual*(t) is a 1-hot vector representing the word that should have been predicted given the context. Table 3.1 representes results of [14] obtained in WSJ

Model	DEV WER	EVAL WER
Lattice 1 best	12.9	18.4
Baseline-KN5 (37M)	12.2	17.2
Discriminative LM (37M)	11.5	16.9
Joint LM (70M) (37M)	-	16.7
Static 3xRNN + KN5 (37M)	11.0	15.5
Dynamic 3xRNN + KN5 (37M)	10.7	16.3

Table 3.1: Comparison of WSJ results obtained with various models(RNN is trained on just 6.4M words)

experiments using RNN.

[15] present several modifications of the original recurrent neural network language model (RNN LM). The present approaches that lead to more than 15 times speedup for both training and testing phases. They also show the importance of backpropagation through time(BPTT) which is an extension to backpropagation algorithm for recurrent networks. With truncated BPTT, the error is propagated through recurrent connections back in time for a specific number of time steps. Thus, the network learns to remember information for several time steps in the hidden layer when it is learned by the BPTT.

The speedup is obtained by assuming that words can be mapped to classes. Thus if we assume thate ach word belongs to exactly one class, we can first estimate the probability distribution over each class using RNN and then calculate the probability of a particular word from the desired class assuming unigram distribution of words within the class. Thus, now we are reducing the connections between hidden and output layer from V to C which is a significant improvement.

They are often very sensitive to small changes in its parameters which changes the gradient by a large amount. Few others are, Bi-directional RNN and Hierarchical RNN.

3.4 Semantic Composition

The Principle of Compositionality is that meaning of a complex expression is determined by the meaning of its parts or constituents and the rules which guide this combination. It is also known as *Frege's Principle.* In our case, the constituents are word vectors and the expression in hand is the sentence/document vector. For example,

The movie is funny and the screenplay is good

In the above sentence, consider the word vectors are represented by $w(x)$ and the sentence vector as $S(x)$. Hence,

$$S(x) = c_1 w_1(x) \Theta c_2 w_2(x) \Theta c_3 w_3(x) \Theta c_4 w_4(x) \ldots \Theta c_k w_k(x) \tag{3.7}$$

where Θ can be any operation(e.g., addition, multiplication) and c_is are constants.

Composition	Accuracy
Average	88.42
Weighted Average	88.41
Multiplication	50.30

Table 3.2: Results of Vector Composition with different Operations

Analyzing the results from Table 3.2, we observed that when we deal with large number of features, there is a presence of large number of *zeros* and presence of a single zero in a feature will make that features contribution zero in the final vector, which happens in our case and thus multiplicative composition fails.

We, therefore, adopt both simple and weighted average methods in our work. The advantage with addition is that, it doesnot increase the dimensionof the vector and captures high level semantics with ease. In fact, [26] have used simple average to construct phrase vectors which they have later used to find phrase level similarity using cosine distance.

[17] showed that relations between words are reflected to a large extent in the offsets between their vector embeddings. They also use additive composition to reflect

semantic dependencies.

$$\textit{queen - king} \approx \textit{woman - man}$$

[3] clearly show that vectors of Neural Language Model and Distributed Model when used with additive composition outperform those with multiplicative composition in Paraphrase Classification task. DM vectors outperform by nearly giving accuracy difference of 6%. They also perfrom very well on Phrase simliarity tasks.
[23] also present yet another model for semantic composition but that uses a sentiment treebank which is a very costly structure to build and it is task dependent. For under-resourced languages such as Hindi, it would take years to build (for English, task done through Amazaon Mechanical Turk). This leads to appreciation of models such as Additive Composition.

Figure 3.5 depicts the approach of recursive neural network. When an n-gram

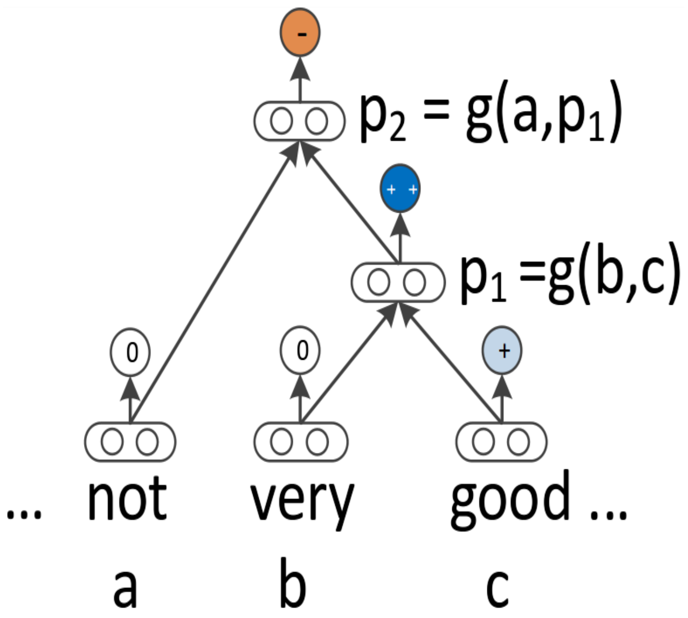


Figure 3.5: Approach of Recursive Neural Network(Figure from [23]).

is given to the compositional models, it is parsed into a binary tree and each leaf node, corresponding to a word, is represented as a vector. Recursive neural models will then compute parent vectors in a bottom up fashion using different types of compositionality functions g. The parent vectors are again given as features to a classifier.

Chapter 4

Data Acquisition

This section describes the corpus used in our experiments.

4.1 Hindi Corpus

4.1.1 Product and Movie Review Corpus

We experimented on two Hindi review datasets. One is the Product Review dataset (LTG, IIIT Hyderabad) containing 350 Positive reviews and 350 Negative reviews. The other is a Movie Review dataset (CFILT, IIT Bombay) containing 127 Positive reviews and 125 Negative reviews.
Each review is around 1-2 sentences long and the sentences are mainly focused on sentiment, either positive or negative.

4.1.2 Movie Review Corpus(Our Contribution)

We collected Hindi movie reviews from websites such as *Dainik Jagran* and *Navbharat Times*. The movie reviews are longer than the previous corpus and contains subjects other than sentiment. There are in total 697 movie reviews from both the websites. The statistics compiled is described below.

Dainik Jagran	
Positive Reviews	210
Negative Reviews	226
Total Reviews	436
29.7 sentences per document	
427.1 words per document	

Table 4.1: Statistics of Dainik Jagran Movie Reviews

Navbharat Times	
Positive Reviews	146
Negative Reviews	115
Total Reviews	261
29.7 sentences per document	
607.2 words per document	

Table 4.2: Statistics of Navbharat Times Movie Reviews

Overall	
Positive Reviews	356
Negative Reviews	341
Total Reviews	697
29.7 sentences per document	
494.6 words per document	

Table 4.3: Statistics of Movie Reviews from Jagran and Navbharat

4.1.3 Wikipedia

We also trained our skip-gram model on Hindi Wikipedia text dump (approx. 290MB) containing around 24M words with 724K words in the vocabulary. This provided us with good embeddings due to larger size and contents from almost all domains.

4.2 English Corpus

4.2.1 IMDB Movie Review

We trained on IMDB movie review dataset (Maas et al.(2013)) which consists of 25,000 positive and 25,000 negative reviews. The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). It also contains an additional 50,000 unlabeled documents for unsupervised learning.

In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. Further, the train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets. In the unsupervised set, reviews of any rating are included and there are an even number of reviews > 5 and <= 5. For example,

- **Postive**: After reading previews for this movie I thought it would be a let down, however after I got my region 1 dvd (the dvd was available before the film hit the uk cinemas) I was pleasantly surprised, strong performances from all cast members make this a very enjoyable movie. The fact that the script is quite weak means that you dont get bogged down in story and therefore the repeat viewing factor is greater. I recommend this movie to one and all

- **Negative**: This is by far the worst non-English horror movie I've ever seen. The acting is wooden, the dialogues are simply stupid and the story is totally braindead. It's not even scary. 2 out of 10 from me.

4.2.2 Trip Advisor Review

It contains around 240K reviews(206MB) from hotel domain. Reviews with overall rating >=3 were annotated as positive and those with overall rating <3 were annotated as negative.
There were total 27315 words in the vocabulary after removing those with count <10. Overall, there were 154448 words in the corpus.
The dataset was split into 80-20 ratio for training and testing purpose.
The Meta data includes: Author, Content, Date, Number of Reader, Number of Helpful Judgment, Overall rating, Value aspect rating, Rooms aspect rating, Location aspect rating, Cleanliness aspect rating, Check in/front desk aspect rating, Service aspect rating and Business Service aspect rating. Ratings ranges from 0 to 5 stars, and -1 indicates this aspect rating is missing in the orginal html file.

4.2.3 Amazon Review

Reviews with overall rating >=3 were annotated as positive and those with overall rating <3 were annotated as negative. The dataset was split into 80-20 ratio for training and testing purpose.
There were 3 review datasets: Watches, Electronics and MP3 each of size 30.8MB, 728.4MB and 27.7MB respectively.
Electronics dataset consists of 1,241,778 reviews, Watches Dataset consists of 68,356 reviews and MP3 Dataset consists of 31,000 reviews.

4.2.4 Wikipedia

We also trained our skip-gram model on Hindi Wikipedia text dump (approx. 20.3GB) containing around 3.5B words with 7.8M words in the vocabulary. This provided us with good embeddings due to larger size and contents from almost all domains.

Chapter 5

Experiment

5.1 Context Size

[16] describe in their work how context size affects the type and quality of word vectors. One of the observations is that a smaller context size tend to capture syntactic similarity to a much better extent. As we increase the context size, vectors tend to capture more of semantic similarity.

We try to look how context size affects the accuracy on two amazon datasets, Watches and MP3.
Figure 5.1 depicts the accuracies obtained by varying the context size from 5 to 10. In both the cases, we find a peak at context size 7. There is an increase from 5 to 7 which denotes increase of context strength. As a result, we see increase in accuracies. Once the context size goes beyond a threshold, the semantic counterpart dominates which attains maximum at context size 10.
We also see that larger data size or training data affects the accuracy on test set. In this case, *Watches* dataset is larger than *MP3* dataset.

Figure **??** shows how the high dimensional representation changes when we change the context size. The embeddings were formed with context sizes 5 and 10 by training on Latest Wikipedia dump.

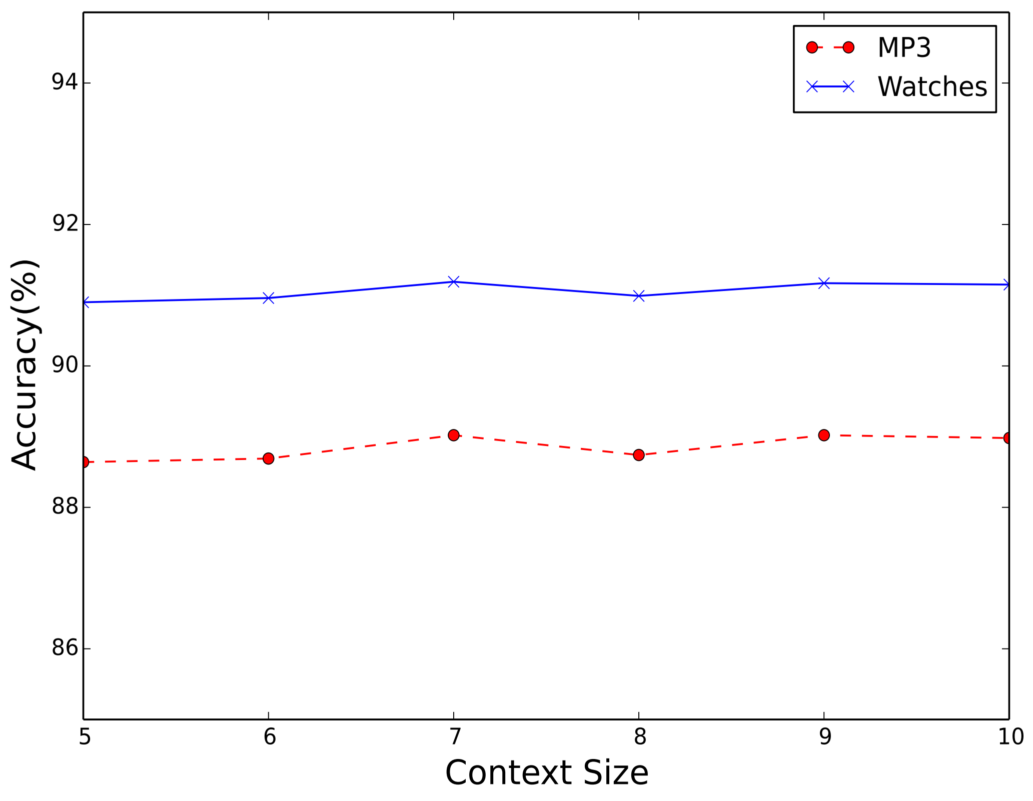


Figure 5.1: Variation of Accuracy with Different Context Size on Watches and MP3 Datasets.

5.2 SkipGram or CBOW

SkipGram model tends to predict a context given a word whereas CBOW model predicts a word given a context. It seems intuitive and also from observation [13] that SkipGram will perform better on semantic tasks and CBOW on syntactic tasks. We now try to evaluate how they differ on classification accuracies on the two datasets: *Watches* and *MP3*. Figure 5.2 show that skipgram outperforms CBOW on sentiment classification task. It can be justified by the fact that sentiment inclination of a document is more oriented towards semantics of that document rather than just syntax and our results clearly demonstrate this fact.

A workflow defined as a graphic summary of the following has been depicted in Figure 5.4. We also trained our skip-gram model on Hindi Wikipedia text dump (approx. 290MB) containing around 24M words with 724K words in the vocabulary. This provided us with good embeddings due to larger size and contents from almost

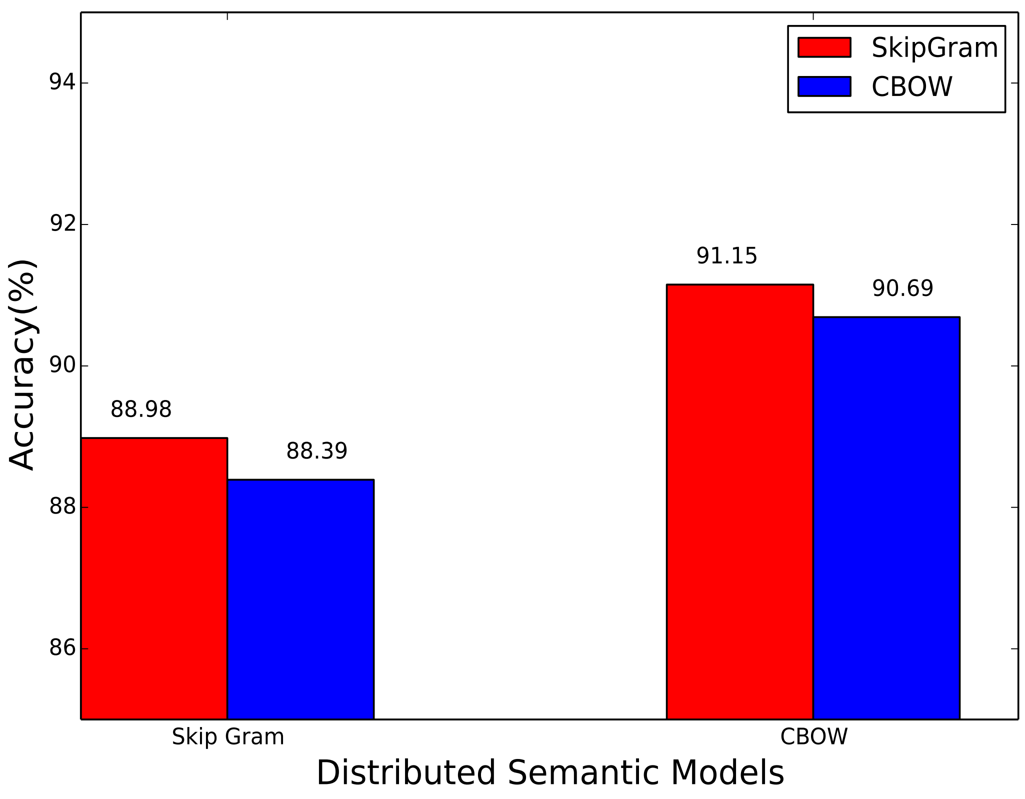


Figure 5.2: Variation of Accuracy with Different Context Size on Watches and MP3 Datasets.

all domains.

The quality of word vectors can be evaluated by comparing them with words which are closer to them semantically and syntactically. This is usually done via cosine similarity. Another evaluation can be done through tSNE [25] which helps in visualization which maps each high-dimensional data point to a two or three-dimensional map. In our experiment, we took 5K words and plotted them with tSNE (fig. 5.3).

Figure 5.3 gives a closer look into few clusters which depicts the relation between words in high dimensional space. Figure 5.3a shows that words such as are closer to each other but farther from words such as

5.2.1 Skip-Gram and *tf-idf* based Word Vectors

In this experiment, we first generated 300-dimensional word vectors by training skip-gram model on both review corpus. The context size was taken as 5. We then averaged-out word vectors for each document to create document vectors. This now acts as a feature set for that particular document. We also created *tf-idf* vectors for each document. This can be seen as a vector representation of that particular document. We then concatenated these document vectors with document vectors obtained after averaging-out word vector of each document. In this case, the dimension of each word vector obtained from skip-gram training was 500.

5.2.2 Skip-Gram and *tf-idf* based Word Vectors without stop-words

In this experiment, we filtered out stop-words on the basis of their frequency in the corpus. Words which had very high or very low frequency were pruned as they had negligible contribution to the sentiment polarity of a document. This is a noise-reduction step and gives better results.

5.2.3 Tools/Libraries

- Eclipse: An Integrated Development Environment helped speed up the process of coding and its subsequent debugging.
- Python: Being a very common and widely used programming language, loads of documentation and third party libraries are available.
- Scikit: Open source library built on NumPy, SciPy and matplotlib which contains simple and efficient tools for data mining and data analysis.

- Gensim: It is an open-source vector space modeling and topic modeling toolkit, implemented in the Python programming language, using NumPy, SciPy and optionally Cython for performance. It is specifically intended for handling large text collections, using efficient online algorithms.

5.2.4 System Requirements

The code and tests have been successfully run on the following configuration. Any system with a configuration equal or higher than this should be able to do the job faster.

- RAM: 16GB; fails to run on 8GB, due to the high amount of in memory data.

- CPU: Intel Core i7, 4th Generation; have used all 8 virtual cores with hyper-threading, with lower CPU, processes would take longer to complete.

- HDD: 280GB dedicated; Million Song Dataset is the only component using considerable persistent memory. Storing user history and codes consume very minimal data storage (in MBs).

5.2.5 Optimizations

- Parallelization: Since, gathering data for every user is fairly independent of each other, these tasks have been implemented in parallel. Also, during the recommendation, once the set of similar users have been collected, suggesting songs per user can again be done in parallel. 16 threads has been found to be the optimal for speed on a CPU with 8 virtual cores (2 hyperthreads per core).

- On-Demand: Most songs from the dataset would not be required for recommending, and might consume some valuable runtime. The songs are loaded in the memory at the first missed access. This not only ensures faster runtime but also a lower comsumption of physical memory.

- Load Minimal Data Efficiently: Only song data like *loundess*, *tempo* and *popularity* that is required for recommending is loaded into memory. This data is stored in a serialized JSON format. This lightens the overhead of high-level data structures.

- Prevent File Access: Each song's information in the million song dataset is stored away in a separate file for each. This heightens the overhead of several files being opened and closed at runtime. Also, there is a limit imposed by the OS to the number of files that can remain open at any given point of time. Thus, only the required information has been extracted, compressed into a serialized format and loaded in memory on an on-demand basis.

5.2.6 Complications

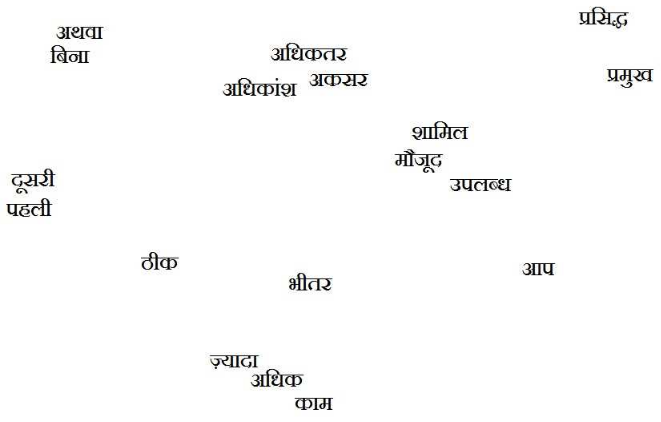
- Last.FM API: The API is not very robust when multiple calls are made in a short span of time. Several of the calls tend to get failed resulting in the obtained history being corrupted. This restricts the use of multi-threading to fetch user data, which would have improved the runtime significantly.

5.2.7 Improvements

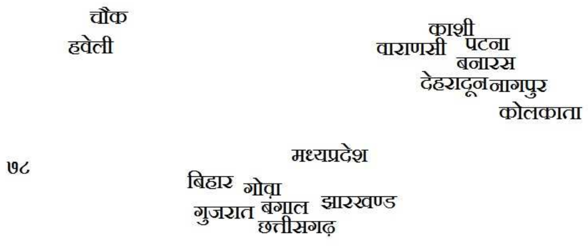
- Distributed Systems: It gives advantages in terms of more number of CPUs and thus more threads at work, enabling to reduce the runtime considerably. Also, the Last.FM API finds itself in a bit of fix when several calls are being made at the same time from the same computer; this also then can be relieved. This would also enable load balancing the recommendation requests.

- NFS: Using an NFS over a local network could shared the dataset and user histories across all the involved machines.

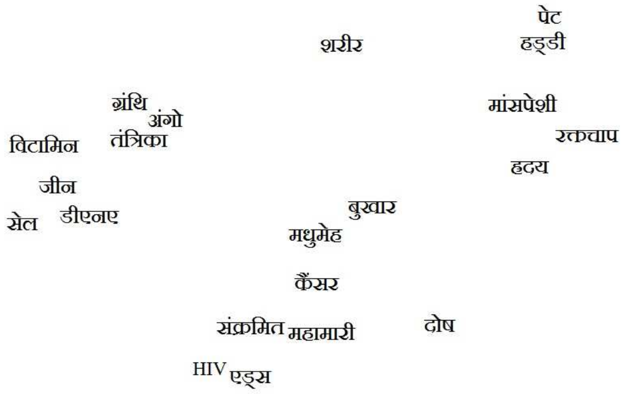
5.3 Work Flow Summary



(a)



(b)



(c)

Figure 5.3: A closer look at two clusters in the visualization showing a) quantity relations, b) locations and c) diseases.

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Figure 5.4: Work Flow

Chapter 6

Results

6.1 Results on English Datasets

Features	**Accuracy**
Dredze et al.(2008)	85.90
Max Entropy	83.79
WordVector Averaging (Our Method)	**89.23**

Table 6.1: Results on Amazon Electronics Review Dataset

Features	**Accuracy**
Maas et al.(2011)	88.89
Paragraph Vector (Le and Mikolov(2014))	92.58
WordVector Averaging+Wiki (Our Method)	88.60

Table 6.2: Results on IMDB Movie Review Dataset

6.2 Results on Hindi Datasets

Features	**Accuracy(1)**	**Accuracy(2)**
WordVector Averaging	78.0	79.62
WordVector+tf-idf	90.73	89.52
WordVector+tf-idf without stop words	**91.14**	**89.97**

Table 6.3: Accuracies for Product Review and Movie Review Datasets.

Table 3 represents the results using three different techniques for feature set

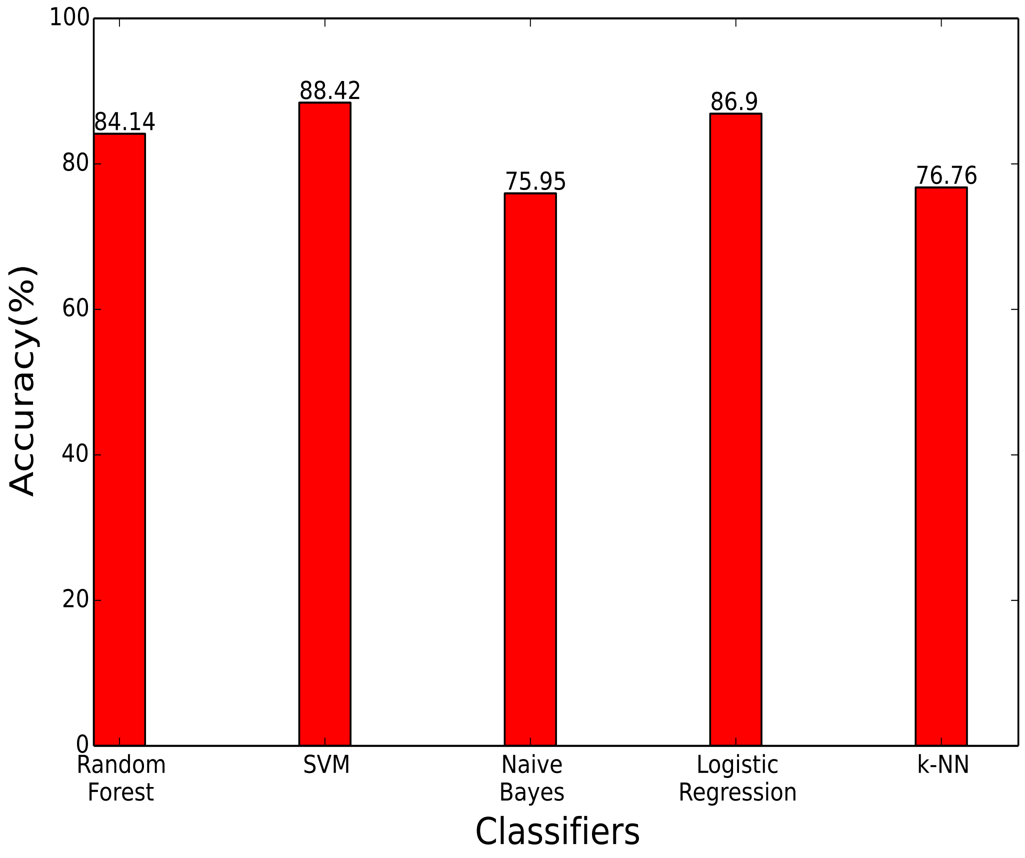


Figure 6.1: Accuracies of Different Classifiers with Average Word Vectors(IMDB).

construction. We see that there is a slight improvement in accuracy on both datasets once we remove stop-words.

Experiment	**Features**	**Accuracy**
Word Vector with SVM (Our method)	tf-idf with word vector	**91.14**
Subjective Lexicon (Bakliwal et al.(2012))	Simple Scoring	79.03
Hindi-SWN Baseline (Arora et al.(2013))	Adjective and Adverb presence	69.30

Table 6.4: Comparison of Approaches: Product Review Dataset

Table 4 and 5 compares our best method with various other methods which have performed well using techniques such as *tf-idf*, subjective lexicon, etc.

Table 5 shows the top few similar words for certain words from the corpus with cosine similarity as a distance metric. The words which have higher cosine similarity tend to be semantically and syntactically related.

Experiment	**Features**	**Accuracy**
WordVector Averaging	word vector	78.0
Word Vector with SVM (Our method)	tf-idf; word vector	**89.97**
In language using SVM (Joshi et al.(2010))	tf-idf	78.14
MT Based using SVM (Joshi et al.(2010))	tf-idf	65.96
Improved Hindi-SWN (Bakliwal et al.(2012))	Adj. and Adv. presence	79.0

Table 6.5: Comparison of Approaches: Movie Review Dataset

अच्छा	**खराब**	**भयानक**
बहुत	निरासाजनक	भयन्कर
सुपर	कम्ज़ोर	भीषण
केवल	नाज़ुक	भयावह
इतना	बदतर	अवसाद

Table 6.6: Some sentiment words and their neighbors

Chapter 7

Conclusion and Future Work

7.1 Inference

In this work we present an early experiment on the possibilities of distributional semantic models (word vectors) for low-resource, highly inflected languages such as Hindi. What is interesting is that our word vector averaging method along with tf-idf results in improvements of accuracy compared to existing state-of-the art methods for sentiment analysis in Hindi (from 80.2% to 89.9%). Distributional semantics approaches remain relatively under-explored for Indian languages, and our results suggest that there may be substantial benefits to exploring these approaches for Indian languages. While this work has focussed on sentiment classification, it may also improve a range of tasks from verbal analogy tests to ontology learning, as has been reported for other languages. In our future work, we seek to explore various compositional models - a) weighted average - where weights are determined based on cosine distances in vector space; b) multiplicational models. Another aspect we are considering is to incorporate multiple word vectors for the same surface token in cases of polysemy - this would directly be useful for word sense disambiguation. Identifying morphological variants would be another direction to explore for better accuracy. With regard to sentiment analysis, the idea of aspect-based models (or part-based sentiment analysis), which looks into constituents in a document and classify their sentiment polarity separately, remains to be explored in Hindi. Another point to note is that we are re-computing the word vectors for the two review corpora, which are extremely small. We may expect better performance with a larger sentiment corpus.

We also observe that pruning high-frequency stop words improves the accuracy by around 0.45%. This is most likely because such words tend to occur in most of the documents and don't contribute to sentiment. Similarly, words with very low frequency are noisy and can be pruned. For example, the word फिल्म occurs in 139/252 documents in Movie Review Dataset(55.16%) and has little effect on sentiment. Similarly words such as सिद्धार्थ occur in 2/252 documents in Movie Review Dataset(0.79%). These words don't provide much information.

Before concludiong, we return to the unexpectedly high improvement in accuracy achieved. One possibility we considered is that when the skip-grams are learned from the entire review corpus, it incorporates some knowledge of the test data. But this seems unlikely since the difference in including this vs not including it, is not too significant. The best explanation may be that the earlier methods, which were all in some sense based on a sentiWordnet, and at that one that was initially translated from English, were essentially very weak. This is also clear in an analysis from [1], which shows intern-annotator agreement on sentiment words are very poor (70%) - i.e. about 30% of these words have poor human agreement. Compared to this, the word vector model provides considerable power, especially as amplified by the tf-idf process. Thus, this also seems to underline the claim that distributional semantics is a topic worth exploring for Indian languages.

7.2 Improvements

This recommendation system has been built more as a *proof of concept* than a competitive tool. A lot of things may enhance the performance the results of the recommendation and to make it even more accurate. Some of them have been discussed below.

- Million Song Dataset: The global dictionary of songs has been restricted to 1,000,000 and recommendations are made from this dataset only. Adding more songs, will improve the quality of recommendations.

- User History: There is a certain amount of loss while fetching the user history and cleaning it up. It corrupts the determined “mood” of the user. Using a cleaner method and a reliable source of history should be able to tackle this matter.

- Users: Currently, the set of users being used for collaborative filtering is not only limited but also very random and does not encompass the different moods and interests of music listeners.

- MFCC: There are several other parameters as obtained from the MFCC analysis of a song. These included appropriately in the computation would significantly bring out better results.

- Feedback: If a user skips a certain song, the recommendation engine should learn not to present her with the song anytime soon again. The weights for each parameters can be dynamically changed as per the user feedback using machine learning techniques.

7.3 Summary

- Corpus: This plays a very significant role in the process of sentiment prediction and has not been very widely tapped. Some corpus tend to be more focused towards sentiment while some contain a lot of information which is redundant as far as their sentiment score is concerned.

- Features: This parameter has been part of what we call in machine learing as *Feature Engineering* and has been very popular whenever a new algorithm appears in the domain of Machine Learning or Natural Language Processing.

Bibliography

[1] Akshat Bakliwal, Piyush Arora, and Vasudeva Varma. Hindi subjective lexicon: A lexical resource for hindi adjective polarity classification. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA), 2012.

[2] Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology*, 9, 2014.

[3] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics, 2012.

[4] Nivet Chirawichitchai. Emotion classification of thai text based using term weighting and machine learning techniques. In *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on*, pages 91–96. IEEE, 2014.

[5] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[6] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving Word Representations via Global Context and Multiple Word Pro-

totypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

[7] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

[8] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

[9] Thomas K Landauer, Darrell Laham, and Peter W Foltz. Automated scoring and annotation of essays with the intelligent essay assessor. *Automated essay scoring: A cross-disciplinary perspective*, pages 87–112, 2003.

[10] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[11] B. K. Matilal. *The Word and The World India's contribution to the study of language*. Oxford Paperback, Delhi, 2001.

[12] Nishantha Medagoda, Subana Shanmuganathan, and Jacqueline Whalley. A comparative analysis of opinion mining and sentiment classification in non-english languages. In *Advances in ICT for Emerging Regions (ICTer), 2013 International Conference on*, pages 144–148. IEEE, 2013.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[14] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[15] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan H Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[17] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[18] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *In Proceedings of ACL-08: HLT*, pages 236–244, 2008.

[19] Namita Mittal, Basant Agarwal, Garvit Chouhan, Nitin Bania, and Prateek Pareek. Sentiment analysis of hindi review based on negation and discourse relation. In *proceedings of International Joint Conference on Natural Language Processing*, pages 45–50, 2013.

[20] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014.

[22] Richa Sharma, Shweta Nigam, and Rekha Jain. Opinion mining in hindi language: A survey. *arXiv preprint arXiv:1404.4935*, 2014.

[23] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the*

conference on empirical methods in natural language processing (EMNLP), volume 1631, page 1642. Citeseer, 2013.

[24] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[25] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. 2008.

[26] Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398, 2013.