Every Word has Different Meaning

Pranjal Singh
Dept. of Computer Science & Engg.
IIT Kanpur
spranjal@iitk.ac.in

Amitabha Mukerjee
Dept. of Computer Science & Engg.
IIT Kanpur
amit@cse.iitk.ac.in

Abstract

In recent years, distributional semantics or vector models for words and documents have been proposed to capture both the syntactic and semantic similarities. Since these are language free models and can be obtained in an unsupervised manner, they are of interest for under-resourced languages such as Hindi as well and many more languages. Language free models are generic and, in recent years, have developed the capability of performing various NLP tasks efficiently. Their capability of being invariant for different languages has attracted the attention of various researchers. We have developed such a language free model using distributional semantics and traditional language representation methods and achieved better results than language models. We test the efficacy of such an approach for many things and especially 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 many English review datasets including IMDB and two small Hindi databases from earlier work and our own built dataset of Hindi movie reviews. We have gone a step ahead with document vectors and built new features merged with various models to achieve state-of-the art results in sentiment classification on classical IMDB movie review dataset achieving an accuracy of 94.19%(improvement of 1.61% over previous best). We also implement an ensemble model which boosts our accuracy by about 0.5% using recursive neural network. We demonstrate the language free aspects by developing a huge increment on results for Hindi. We also propose that dimensionality reduction techniques such as ANOVA-F and PCA are of great help to reduce noise and boost accuracy of our models. In order to handle larger strings from the word vectors, several methods - additive, multiplicative, or tensor neural models, have been proposed. Here we propose weighted additive average technique, which results in an impressive accuracy gain on state of the art by 12% (from 80%) for two review datasets. The results suggest that it may be worthwhile to explore such methods further for Indian languages.

1 Introduction

Since a very long time back whenever researchers have tried to model languages, they fell into broadly two categories: ones which are language dependent models(e.g.,(**?**)) and ones which are language independent models(e.g., LDA, BOW, SkipGram, NLM, etc.). We have tried to present a work which is language independent and which is enriched with vital properties of other models. This ensures that our model is effective as well as accurate in various languages. The world class results in English clearly indicate the efficacy of our approach and improvements in Hindi depict the deficiency in other models which were used earlier.

Language independent models such as LDA and BOW have been quite effective since long time. Variants of BOW such as tf-idf had changed the perception of researchers towards these models when they were proved effective in various NLP tasks. LDA was able to model inter and intra documental statistical and relational structure quite well overcoming the drawbacks of BOW. But the semantic and syntactical dependencies were still ignored. After the introduction of neural language

vector models, NLP saw a huge diversion in representation of words and documents.

Vector models for individual words are obtained via distributional learning, the mechanisms for which varies from document-term matrix factorization (**?**), various forms of deep learning (**?**; **?**; **?**), optimizing models to explain co-occurrence constraints (**?**; **?**),etc. Once the word vectors have been assigned, similarity between words can be captured via cosine distances. The same models have been extended((**?**)) with new variables to build vector models for sentences and documents. These models include the essence of individual words as well as their relative order in terms of sentence vector which was earlier absent in word vectors.

There has been another problem which has attracted the attention of Indian grammarians for over a period of nearly a millennium. They 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 (**?**). 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. (**?**)). The latter line has resulted in strong improvements in several NLP tasks using word vectors (**?**; **?**; **?**; **?**). Recently, the introduction of sentence/document vectors have achieved very good results in various NLP task (**?**). The advantage of these approaches is that they can capture both the syntactic and the semantic similarity between words/documents 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 (**?**).

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 (**?**).

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 (**?**), but multiplicative models (based on a reducing the tensor products of the vectors) appears to correlate better with human judgements (**?**; **?**). 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 (**?**). The results reported here, are based on applying the Skip-Gram model (**?**) to Hindi.

(**?**) give a detailed overview of various vector space models and their composition. A surprising event in Information Theory has higher information content than an expected event (Shanon, 1948). The same happens when we give weights to word vectors. We give more weights to events which occur by surprise and less weight to events which are expected. The most popular weighting concept in this domain is the idea of tf-idf which is explained later. (**?**) defined a large family of tf-idf weighting functions and evaluated them on information retrieval tasks, demonstrating that tf-idf weighting can yield significant improvements over raw frequency. Term weighting is also significant in reducing the weights of highly correlated words such as *hostage* and *hostages* when they occur together (Church, 1995). (**?**) have found use of term weighting in sentiment classification in Thai language.

A single learning algorithm in any domain cannot always induce the most accurate learner and hence we include in this work, the concept of ensemble learning though it is not ensemble in absolute sense. We require that our classifier should have properties of both a discriminative model and a generative classifier and hence merged the predictions of both achieving better results(majority voting). Since our main focus is not ensemble, so we donot look into other ensemble methods in detail.

1.1 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 रफ़तार 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 both Hindi and English (see section **??**). There have been heuristic based and machine larning based models used in this domain (**?**). Heuristic based methods, in general, classify text sentiments on the basis of total number of derived positive or negative sentiment oriented features. For example, Hatzivassiloglou and McKeown (**?**) described that adjectives are more predictive of sentiment classification and they predicted the sentiment of adjectives by inspecting them in conjunction with "and", "or", "but", "either/or" and "neither/nor". However the problem with this approach is that even if adjectives are important in many languages but this model underestimates some predictive words of other parts-of-speech which may be important in some languages. Also these models rely heavily on human engineered features which, in general, is a domain and language dependent task.

In this work, we first learn a distributional word vector model as well as sentence vector model based on the wikipedia 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%). With weighted additive composition, the gain is more than 2% beating the state-of-the-art in Hindi((**?**)). 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). (**?**) give a summary of work done in Hindi in the field of opinion mining.

We then go a step ahead to build a model using document vectors along with tfidf and ensemble techniques to achieve state-of-the-art result on IMDB movie review dataset achieving a significant improvement over the previous best. We have got an accuracy of 94.19% by using an ensemble method merged with recursive neural network to show the excellency of generative models merged with discriminative ones. Our model uses concatenation of average word vectors, sentence vectors and tf-idf to build informative 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).
For Hindi, we have experimented with tf-idf and weighted average word vectors 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.

2 BAckground and Related Work

Sentiment analysis is a well-known research area in NLP today (see reviews in (**?**) and Pang et al. (2008), and also challenge in SemEval-2014). Early work on movie review sentiments achieved

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

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.

Neelakantan et al.(2015)(**?**) took word vector models to next level where they proposed multiple embeddings per word. Nearly all the work before this assumes a single vector per word typeignoring polysemy and thus jeopardizing their usefulness for downstream tasks. The authors present an extension to the Skip-gram model that efficiently learns multiple embeddings per word type. It differs from recent related work by jointly performing word sense discrimination and embedding learning, by non-parametrically estimating the number of senses per word type, and by its efficiency and scalability. They achieve state-of-the-art results in the word similarity in context task and demonstrate its scalability by training with one machine on a corpus of nearly 1 billion tokens in less than 6 hours.

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 been criticized 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 (**?**; **?**; **?**; **?**).

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% (**?**).

There has been limited work on sentiment analysis in Hindi – see review in (**?**), 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%.

(**?**) traversed the WordNet ontology to antonyms and synonyms to identify polarity 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.

(**?**) presents new recurrent neural network based langauge model which has found vital application in various NLP tasks as well. They provide ample empirical evidence to suggest that connectionist language models are superior to standard n-gram techniques, except their high computational (training) complexity. The model has also been extended to speedup training time by few optimizations.(More details in 3.4).

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

3 Word-Vectors by using Skip-Gram

Mikolov et al. (2013b) proposed two neural network models for building word vectors from large unlabeled 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:

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

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 outputting one multinomial distribution, we are outputting 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

$$\begin{aligned} \text{E} &= -\log p(w_{O,1}, w_{O,2}, ..., w_{O,C}|w_I) \\ &= -\log \prod_{c=1}^{C} \frac{exp(u_{c,j_c^* \Psi})}{\sum_{j'=1}^{V} exp(u_{j'})} \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C. \log \sum_{j'=1}^{V} exp(u_{j'}) \end{aligned}$$

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

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

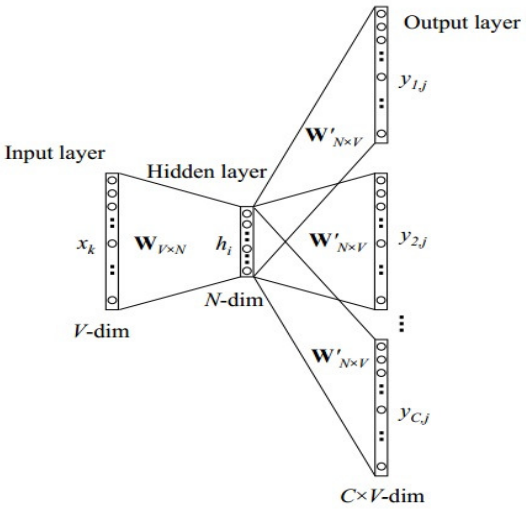

Figure 1: Skip Gram Model(Figure from Rong (2014))

The main advantage of using skip-gram is that it is computationally less expensive than other neural language models with a complexity of O(log V) instead of O(V). They use hierarchical soft-max((**?**)) to achieve such computational efficiency.

3.1 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.

3.2 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 (**?**; **?**). 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)

3.3 Document Vectors

This distributed representation of sentences and documents (**?**) modifies word2vec (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 concatenated 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 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.

In Figure 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)).

In Figure 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

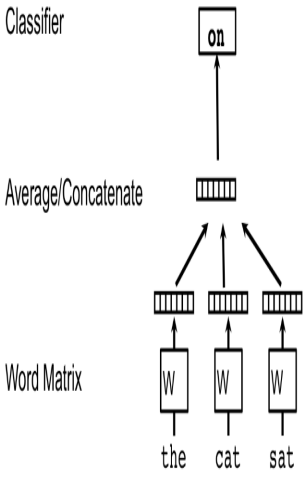

Figure 2: Framework for learning word vectors(Figure from Le (2014)).

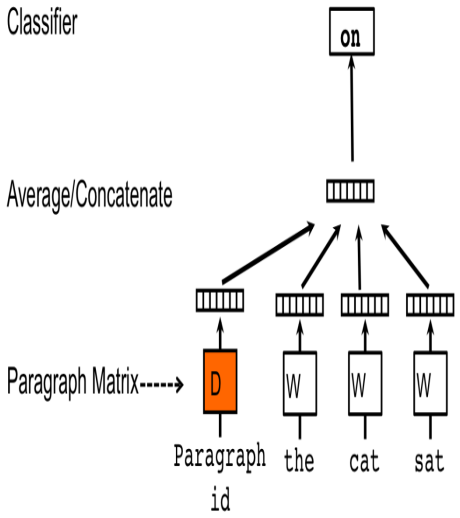

Figure 3: Framework for learning paragraph vectors(Figure from Le (2014)).

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 property 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 sentence/paragraph, 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.

3.4 Recurrent Neural Network

The primary feature of a Recurrent Network is that it contains atleast one feed-back 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.

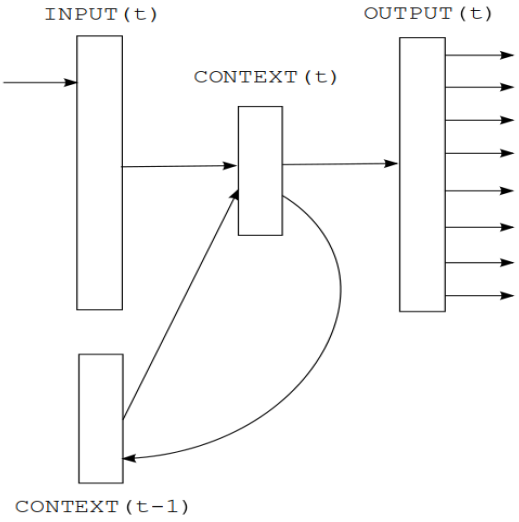

Figure 4: Simple Recurrent Neural Network(Figure from (**?**)).

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 concatenation 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 below:

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

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right) \tag{2}$$

$$y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right) \tag{3}$$

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

$$f(z) = \frac{1}{1 + \mathrm{e}^{-z}} \tag{4}$$

$$g(z_m) = \frac{\mathrm{e}^{z_m}}{\sum_k \mathrm{e}^{z_k}} \tag{5}$$

(**?**) 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:

$$error(t) = actual(t) - y(t) \tag{6}$$

where $actual(t)$ is a 1-hot vector representing the word that should have been predicted given the context. Table 1 represents results of (**?**) obtained

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

in WSJ experiments using RNN.

(**?**) 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 that each 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.

3.5 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,

$$S(x) = c_1 w_1(x) \Theta c_2 w_2(x) \Theta c_3 w_3(x) \Theta c_4 w_4(x) \ldots \Theta c_k w_k(x) \tag{7}$$

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 2: Results of Vector Composition with different Operations

Analyzing the results from Table 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 dimension of the vector and captures high level semantics with ease. In fact, (**?**) have used simple average to construct phrase vectors which they have later used to find phrase level similarity using cosine

distance.

(**?**) 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.

$$queen - king \approx woman - man$$

(**?**) 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 perform very well on Phrase similarity tasks.

(**?**) 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 Amazon Mechanical Turk). This leads to appreciation of models such as Additive Composition.

Figure 5 depicts the approach of recursive neural

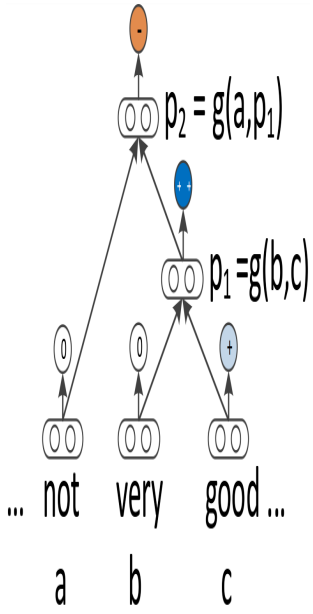

Figure 5: Approach of Recursive Neural Network(Figure from (**?**)).

network. When an n-gram 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.

3.6 Dimensionality Reduction

Dimensionality Reduction is the process of reducing the number of random variables in such a way that the remaining variables effectively reproduce most of the variability of the dataset. The reason for using such techniques is because of the *curse of dimensionality* which is a phenomena that occurs in high-dimension but doesn't occur in low-dimension.

In machine learning problems that involve learning a "state-of-nature"(maybe an infinite distribution) from a finite number of data samples in a high-dimensional feature space with each feature having a number of possible values, an enormous amount of training data are required to ensure that there are several samples with each combination of values. With a fixed number of training samples, the predictive power reduces as the dimensionality increases, and this is known as the Hughes effect[3] or Hughes phenomenon (named after Gordon F. Hughes)[1]

It happens, in general, that the intrinsic dimension is small but the data is represented by a large number of features. e.g. there may be some features across which the variance is very low and hence we can safely ignore such features reducing the dimension.

Dimensionality Reduction helps in visualization of high-dimensional data in 2D or 3D, data compression for efficient retrieval and storage and noise removal. In machine learning, it is often used for feature selection and feature extraction. In this work, we have successfully used dimensionality reduction for feature selection i.e., we have selected an optimal subset of features from the given data to achieve a large improvement in accuracy.

Next we will discuss two of the techniques which we have used in our work and present the results obtained after applying these techniques.

3.6.1 Principal Component Analysis(PCA)

Principal Component Analysis(PCA) is a very famous technique for dimensionality reduction in machine learning. When given a data with n-dimensions/features, PCA aims to find a linear subspace of dimension d lower than n such that this reduced space contains most of the data points and it maintains most of the variability of the data. It can be easily proved that first principal compo-

[1] Wikipedia

nent is given by the normalized eigenvector with the largest associated eigenvalue of the sample covariance matrix S. A similar argument can show that the d dominant eigenvectors of covariance matrix S determine the first d principal components. Also, the projection onto the principal subspace minimizes the squared reconstruction error, $\sum_{i=1}^{t} \|x_i - \widehat{x_i}\|^2$.

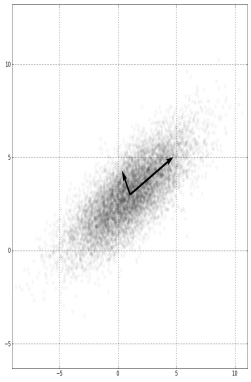

Figure 6: PCA of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3 in roughly the (0.878, 0.478) direction and of 1 in the orthogonal direction. The vectors shown are the eigenvectors of the covariance matrix scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean(Figure from (**?**)).

3.6.2 ANOVA: F-value

Analysis of variance(ANOVA) is a set of statistical models used to analyze the differences between the group means and variation among and between groups. It was developed by R.A. Fisher. It is also a measure of statistical significance of the data point with reference to other data points present in the sample.
In our experiment, we have used F-test to reduce the number of features and it has led to a considerable increase in classification accuracy on our new movie dataset(Refer 4.1.2).
The F-statistic is calculated as:

$$F = \frac{\text{variance between groups}}{\text{variance within groups}}$$

The variance between groups is given as:

$$\frac{\sum_i n_i(\bar{Y_i} - \bar{Y})^2}{K-1} \tag{8}$$

where $\bar{Y_i}$ denotes the sample mean in the i^{th} group, n_i is the number of observations in the i^{th} group, $\bar{Y}$ denotes the overall mean of the data, and K denotes the number of groups.
The variance within groups is given as:

$$\frac{\sum_{ij}(Y_{ij} - \bar{Y_i})^2}{N-K} \tag{9}$$

where Y_{ij} is the j^{th} observation in the i^{th} out of K groups and N is the overall sample size.

3.6.3 Result

Method	Feature Selection	Accuracy
Document Vector	None	74.57
	PCA(n=50)	76.33
	ANOVA-F	88.07
Weighted Average Word Vector	None	76.43
	ANOVA-F	90.37
	PCA(n=50)	78.61

Table 3: Accuracies on 700-Movie Review Dataset

Table 3 summarizes how feature selection has improved classification accuracy on the 700 Movie review dataset. With ANOVA-F, we selected around 4k features but with PCA, this number was just 50. So, the low accuracy with PCA can be attributed to the fact that we may have lost some important features in low dimension. Also, PCA cannot work with size of dimension $d >$*size of learning set*.
This sharp decrease in accuracy in both cases: Document vector and Weighted Average word vector happens because ANOVA-F selects features with larger variance across group and thus reduces noise to a larger extent whereas PCA reduces angular variance which is not effective in this case due to the distribution of data points in high-dimensional space.

Corpus	Feature Selection	Accuracy
Movie Reviews-IITB	None	79.17
	ANOVA-F	89.17
Product Reviews-IIIT	None	86.86
	ANOVA-F	90.83

Table 4: Accuracies on IITB Movie Review and IIIT Product Review Dataset using Document Vectors

Again from Table 4, we observe that ANOVA-F performs much better than PCA. Reducing dimension to 4k preserves important features whereas

we are losing information once that number is reduced to 50.
So we easily infer that with these datasets, ANOVA-F is the feature selection method to adopt.

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.

4.1.2 700-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 5: Statistics of Dainik Jagran Movie Reviews

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.

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

Table 6: 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 7: Statistics of Movie Reviews from Jagran and Navbharat

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,

- **Positive**: 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 original 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.

5 Models and Experiments

5.1 Context Size

(**?**) 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.

Figure 9 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. We see that 5.1(a), has words {*high,higher,highest*} very close to each other which is in accordance to our argument that smaller context size promotes syntactical similarity whereas these words are farther when we increase the context size. From 5.1(b), we can see that there is a larger semantic difference between two clusters of words:{*wikiproject,wikipedia*} and {*facebook,html,id*}. This again justifies our argument that larger context size favors more of semantic similarity than syntactic similarity.

We try to look how context size affects the accuracy on two amazon datasets, Watches and MP3. Figure 5.2 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.

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 (**?**) 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 11 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

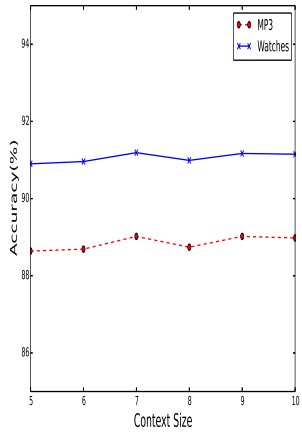

Figure 10: Variation of Accuracy with Different Context Size on Watches and MP3 Datasets.

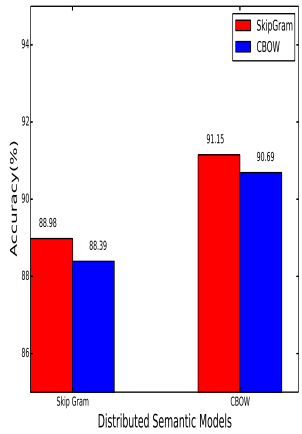

Figure 11: Variation of Accuracy with Different Context Size on Watches and MP3 Datasets.

Figure 7:

Figure 8:

that document rather than just syntax and our results clearly demonstrate this fact.

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

In this experiment, we first generated n-dimensional word vectors by training skip-gram model on product and movie review corpus in Hindi and IMDB corpus in English. 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 another experiment, we filtered out stopwords 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.

5.4 Weighted Average Word Vectors

Algorithm **??** presents a new approach to build document vectors using distributional semantics.Till date, everyone has ignored how to effectively use vector composition techniques and as a result, this area has seen very less attention. But we have successfully used *idf* values to give weights to word vectors and hence obtain much better sentence/document vectors. Refer to 7 to see how this new technique has improved sentiment classification on Hindi reviews.

The advantage of this model is that once we obtain *idf* values from training corpus, we can directly use it with test corpus without any additional computation.

5.5 Word and Document Vectors with tf-idf: Enhanced Document Vectors

This experiment redefined document representation in NLP used for sentiment classification. It has the property of including both syntactic and semantic properties of a piece of text. The limitations of skip-gram word vectors have been fulfilled by document vectors and hence we achieve state-of-the-art results on IMDB movie review dataset.

5.6 Ensemble of RNNLM and Enhanced Document Vectors

This experiment was done on IMDB movie review dataset. Here, we first trained a *recursive neural network* and then obtained predictions on test reviews in terms of probability. We trained another classier(Linear SVM) using Enhanced Document Vectors and then obtained predictions on test reviews. We then merged these two predictions using a simple heuristic, described below, to obtain final classification. This led to a tremendous increase in classification accuracy(Refer section **??**). Let $y*$ be actual output and y is the predicted output. The heuristic used is:

$$((RNNLM_{pred} - 1) * 7 + (0.5 - SVM_{pred})).y < 0 \text{ if } y* = 1 \tag{10}$$

$$((RNNLM_{pred} - 1) * 7 + (0.5 - SVM_{pred})).y < 0 \text{ otherwise} \tag{11}$$

5.7 Embeddings

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.

$$similarity = cos(\theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B} \tag{12}$$

Another evaluation can be done through tSNE (**?**) 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. 12).

Figure **??** gives a closer look into few clusters which depicts the relation between words in high dimensional space. Figure 7 shows that words such as मौजूद and उपलब्ध are closer to each other but farther from words such as ज्यादा and अधिक.

5.8 Work Flow

This section summarizes our approach in brief with graphics. This is a general classification system and includes building of enhanced document vectors.

6 Results on English Datasets

Table 8 summarizes the results obtained by others and by us on the IMDB movie review dataset. We have gone above the previous best((**?**)) by a margin of 1.33% using enhanced document vector.

The main contributor for improvement in results is our enhanced document vector which overcomes the weaknesses of BOW and document vectors taken separately.

Table 9 is a further improvement in results once we incorporate predictions of RNNLM and enhanced document vector model together(voting ensemble).

Table 10 gives an overview of the results obtained with various experiments that we conducted on IMDB dataset. We have tried various vector composition models and feature composition models to obtain good results. Even a simple feature composition of word and document vector yield an improvement over the existing state-of-the-art suggesting that feature composition is a vital area when it comes to NLP tasks such as sentiment analysis.

Figure 14 gives an overview of accuracies of various classifiers using word vector averaging

method on IMDB dataset. This clearly gives a green signal to us for using Linear SVM as our classier in all experiments.

Table 11 presents result of another experiment conducted on famous Amazon electronics review dataset(refer 4.2.3). Our vector averaging method alone has beaten previous best by 3.3% which was based on maximum entropy method.

7 Results on Hindi Datasets

Table 12 represents the results using five different techniques for feature set construction. We see that there is a slight improvement in accuracy on both datasets once we remove stop-words but the major breakthrough occurs once we used weighted averaging technique for construction of document vectors from word vectors.

Table 13 and 14 compares our best method with various other methods which have performed well using techniques such as *tf-idf*, subjective lexicon, etc.

Table 15 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.

8 Odd One Out

We trained Skip-Gram model on latest English and Hindi Wikipedia dump and tried to analyze the quality of embeddings by finding the odd word amongst a list of words. For this task, we took found cosine similarity between each pair and found the word with lowest cosine similarity with every other. The results are highlighted below-

9 Similar Words

The trained skip-gram model was used for this task as well. We found the top few words which were closer in terms of cosine distance to the give word. The results are highlighted below-

10 Conclusion

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 90.3% on IITB Movie Review Dataset).

We 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.

We also observe that giving weights to word vectors while adding has resulted in a significant increase of accuracy. This seems much in accordance with expectation as pure addition ignores the importance of word which is directly related to its ratio of frequency in whole corpus and its frequency in a particular document. We have successfully tapped this area which most of the works have ignored.

We also see that when number of features accumulate to a large number than there are few redundant features present also. This brings a noise in the representation of the text in high dimension. We tried to reduce this noise by using feature variance technique called ANOVA-F and PCA and have succeeded as well. There is a large increase in accuracy(around 11%).

Before concluding, 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 (**?**), 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.

Our experiments on new dataset and existing datasets show that our method is competitive with existing methods including state-of-the-art. This concept of Enhanced Document Vectors can overcome the weaknesses of existing models which were either deficient in capturing syntactic or semantic properties of text. The ensemble of RNNLM and Enhanced Document Vector has beaten state-of-the-art by a significant margin and has opened this area for future research. These models have the advantage that they don't require parsing at any step neither do they require a lot of heavy pre-processing. These tasks require a lot of extra effort and they slow the progress a lot. We have made the code available at `http://github.com/pranjals16/thesis` for anyone to reproduce the result as well as extend this work further.
The challenge of obtaining a better corpus still remains with Hindi even though we have released a much larger corpus of movie reviews for sentiment analysis and for other NLP tasks as well.

11 Future Work

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 focused 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. For future work, we can explore various compositional models - a) weighted average - where weights are determined based on cosine distances in vector space; b) weighted multiplicative 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.

In English, our enhanced document vectors has led open a new area to look at where there can be many possible ensembles which may improve our work. Also, we could incorporate multiple word vectors here as well to distinguish between polysemous words. Another interesting and open area is to look at *Region of Importance* in NLP where we filter out sentiment oriented sentences and phrases from a unfocused corpus which contains text from various domains.

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503–512.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Figure 12: t-SNE visualization of the top 5000 Hindi Words in high dimensional space. (Magnify to see details).

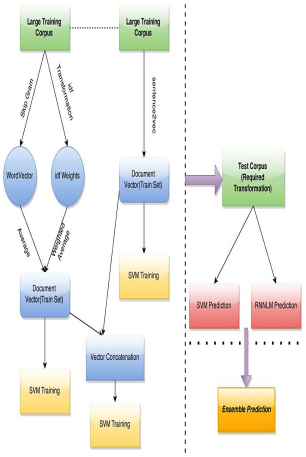

Figure 13: Work Flow Summary.

Method	Accuracy
Maas et al.(2011)	88.89
NBSVM-bi (Wang & Manning, 2012)	91.22
NBSVM-uni (Wang & Manning, 2012)	88.29
SVM-uni (Wang & Manning, 2012)	89.16
Paragraph Vector (Le and Mikolov(2014))	92.58
WordVector+Wiki(Our Method)	88.60
WordVector+TfIdf(Our Method)	89.03
WordVector Averaging+TfIdf+Document Vector	**93.91**

Table 8: Results on IMDB Movie Review Dataset

Method	Accuracy
WordVector Averaging+TfIdf+Document Vector	93.91
WordVector+TfIdf+Document Vector+RNNLM(Our Method)	**94.19**

Table 9: Results on IMDB Movie Review Dataset

Method	Accuracy
WordVector Averaging	88.42
Weighted WordVector Average	88.41
WordVector Averaging+Wiki	88.60
WordVector Averaging+TfIdf	89.03
WordVector Averaging+Document Vector	93.24
WordVector Averaging+Wiki+Document Vector	93.18
WordVector Averaging+Document Vector+RNNLM	93.70
WordVector Averaging+Wiki+Document Vector+RNNLM	93.57
WordVector Averaging+TfIdf+Document Vector	93.91
WordVector Averaging+Wiki+Document Vector+TfIdf	93.55
WordVector Averaging+TfIdf+Document Vector+RNNLM	**94.19**

Table 10: Comparison of results on IMDB Movie Review Dataset with Various Features

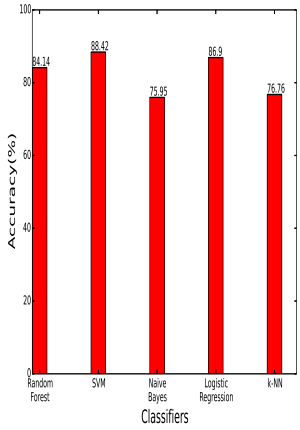

Figure 14: Accuracies of Different Classifiers with Average Word Vectors(IMDB).

Features	Accuracy
Dredze et al.(2008)	85.90
Max Entropy	83.79
WordVector Averaging (Our Method)	89.41
WordVector Averaging+Document Vector(Our Method)	**92.17**

Table 11: Results on Amazon Electronics Review Dataset

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
Weighted WordVector	89.71	85.90
Weighted WordVector+tf-idf	**92.89**	**90.30**

Table 12: Accuracies for Product Review and Movie Review Datasets.

Experiment	Features
Subjective Lexicon (Bakliwal et al.(2012))	Simple Scoring
Hindi-SWN Baseline (Arora et al.(2013))	Adjective and Adverb pres
Word Vector with SVM (Our method)	tf-idf with word vector
Weighted Word Vector with SVM (Our method)	tf-idf+weighted word vec

Table 13: Comparison of Approaches: Product Review Dataset

Experiment	Features	Accuracy
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
WordVector Averaging	word vector	78.0
Word Vector with SVM (Our method)	tf-idf; word vector	89.97
Weighted Word Vector with SVM (Our method)	tf-idf+weighted word vector	**90.30**

Table 14: Comparison of Approaches: Movie Review Dataset

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

Table 15: Some sentiment words and their neighbors

breakfast	**cereal**	lunch	dinner
eight	seven	**owe**	nine
shopping	math	reading	science

Table 16: Odd One Out in English

भारत	**मुम्बई**	रूस	चीन
लड़की	बेहन	**मर्द**	महिला
उद्योग	नेता	मंत्री	सरकार

Table 17: Odd One Out in Hindi

Father	France	XBOX	scratched	megabits
grandfather	Germany	XBLA	scraped	gigabits
uncle	French	Xbox360	rubbed	kilobits
mother	Greece	SmartGlass	bruised	megabit
father-in-law	Netherlands	360/PS3	cracked	terabits
brother	Scotland	XBLA	discarded	MB/s
-	-	Qubed	shoved	Tbit/s
-	-	Kinect	tripped	-

Table 18: Top Few Similar words in English

भारत	व्यापार	ओबामा
प्रदेश	व्यवसाय	क्लिंटन
तिब्बत	पुनर्बीमा	बराक
देश	वाणिज्य	सीनेटर
आँध्रप्रदेश	बैंकिंग	राष्ट्रपति
लद्दाख	उद्योग	उम्मीदवार

Table 19: Top Few Similar words in Hindi