Sentiment Analysis and Deep Learning: A Survey

Prerana Singhal and **Pushpak Bhattacharyya**
Dept. of Computer Science and Engineering
Indian Institute of Technology, Powai
Mumbai, Maharashtra, India
{singhal.prerana,pushpakbh}@gmail.com

Abstract

Deep learning has an edge over the traditional machine learning algorithms, like SVM and Naïve Bayes, for sentiment analysis because of its potential to overcome the challenges faced by sentiment analysis and handle the diversities involved, without the expensive demand for manual feature engineering. Deep learning models promise one thing - given sufficient amount of data and sufficient amount of training time, they can perform the task of sentiment classification on any text genre with minimal restrictions and no task-specific or data-specific manual feature engineering. In this survey, we describe some of the different approaches used in sentiment analysis research.

1 Introduction

Previous research work has shown that basic machine learning techniques produce effective results in performing several natural language processing tasks like topic categorization of documents. However the same techniques cannot be *naively* used for sentiment classification. The non-trivial nature of the latter demands extra effort to contribute effectively towards opinion classification. Opinions need more *understanding* (Pang et al., 2002) for them to be analyzed properly.

We discuss some techniques from the two machine learning paradigms: *traditional models*, which have proved useful for sentiment analysis since over the past few decades, and *deep learning models*, which have emerged as a powerful tool for natural language processing in recent years.

2 Traditional Paradigm

Extensive research has been done over the past years to exploit popular machine learning algorithms for the task of sentiment classification (Pang et al., 2002). Depending on the problem statement in opinion mining, these classifiers have shown good performance accuracy, provided proper feature engineering and pre-processing steps are carried out prior to the classification process.

2.1 Features Used

To perform classification, traditional machine learning algorithms require hand-crafted features as input. Hence, given an input piece of text, the first step is to decide upon the features which can be useful for sentiment classification. Researchers have experimented with varied schemes of feature extraction (González-Ibánez et al., 2011; Buschmeier et al., 2014), some of which are enumerated as follows:

- **Lexical features:** The occurrence of *n-grams* (generally *unigrams* and *bigrams* in labeled documents, either as count or as boolean representation, can be used as features. An alternative to using *all n-grams* (occurring in documents) can be to use only some fixed number of most frequently occurring n-grams in the dataset for classification purposes.

- **Sentiment lexicon-based features:** The *SentiWordNet* (Esuli and Sebastiani, 2006) is a lexical resource that adds sentiment-related information to the WordNet. It tags the synsets with three scores - positive, negative and objective score – between 0.0 and 1.0, which can be used as features for classification. Also a set of *predefined positive and negative sentiment lexicons* such as that

formed by Hu and Liu (Hu and Liu, 2004) can be used. Given a piece of text, the occurrence of one or more of these words can be used in the form of a feature vector.

- **Parts of Speech:** Appending each word with parts of speech tag helps in sense disambiguation of the word to some extent which in turn can help in sentiment classification. For example, the word *love* expresses positive sentiment when used as a verb (*People love comedy movies*) but is neutral in sentiment when used as a noun (*He writes love stories*). Hence POS tags can serve as useful features for detecting sentiments.

- **Adjectives and Adverbs:** Adjectives (and adverbs to certain extent) have been the focus of many researchers for sentiment detection as it is argued that they carry most of the information regarding sentiment of a document. Hence, using these as features can produce good classification results (Pang et al., 2002).

- **Interjections and Question marks:** Expressions like '*haha*' or '*wow*' contain strong sentiment information and hence can be used as features. Also, question marks can change the sentiment orientation of a text. For example, the statement '*It's a great weather.*' is a positive opinion whereas '*It's a great weather?*' clearly reflects negative notion.

2.2 Traditional Models

Naïve Bayes Classifier is the simplest and the most widely used probabilistic classification algorithm (Russell et al., 2003; McCallum et al., 1998). It is based on Bayes' Theorem. It basically calculates the posterior probabilities of events and assigns the label with the maximum posterior probability to the event.

A major assumption made by the Naïve Bayes Classifier is that the features are conditionally independent, given the sentiment class of the document (Pang et al., 2002), which is not true in real-life situations. Furthermore, another problem with this technique is that, if some feature value, which was not encountered in the training data, is seen in the input data, its corresponding probability will be set to 0. Bayes classifier fails in this case. To remove this undesirable effect, smoothing techniques are applied (Chen and Rosenfeld, 2000).

Maximum Entropy classifier is another model which performs probabilistic classification, making use of the exponential model. It is based on the *Principle of Maximum Entropy* (Jaynes, 1957) which states that subject to the prior data which has been precisely stated, the probability distribution which describes this data with the current knowledge in the best possible manner is the one with the largest possible entropy value. This technique has been proven to be effective in many NLP classification tasks (Berger et al., 1996) including sentiment analysis.

Max entropy classifier is seen to outperform the Naïve Bayes in many cases (Pang et al., 2002) (though not always). One major advantage of this classifier is that it makes no conditional independence assumption on the features of the documents to be classified, given a sentiment class. Hence, it is applicable to real-life scenarios, unlike in case of Naïve Bayes.

Support Vector Machines (SVMs) (Cortes and Vapnik, 1995) have proved to be highly effective for the categorization of documents (Joachims, 1998) based on similar topics. As opposed to the probabilistic classifiers like the previous two (Rennie et al., 2003), this method aims to find *large margin* between the different classes. It is a supervised learning model which analyzes data and learns patterns which can be used to classify the data.

Support vector machines attempt to find a *hyperplane* (in case of 2-class classification problem) which not only separates data points based on the category they belong to, but also tries to maximize this separation gap between the two classes, *i.e.*, this is a constrained optimization problem.

One major advantage of this classifier is that it makes no assumption on the documents to be classified and it endeavors to find the best classification margin for the data at hand instead of relying on probability values. It is one of the widely used machine learning algorithms, which yields very good results for the task of sentiment analysis (Pang et al., 2002).

2.3 Possible Limitations

Most of the classical machine learning algorithms for text classification are either rule-based or

corpus-based. Their efficiency depends on the quality of the annotated corpora as well as the feature engineering task involved prior to the classification. The features need to be manually handcrafted as well as they differ from domain to domain and document to document, which makes it less generic and more text-specific.

The accuracy of these systems depends on how the features were chosen, which makes the system liable. Furthermore, it is very difficult, and many a times not feasible, to adapt a system designed for a particular problem to new problems or in different language for the same problem. And for texts like tweets, which do not follow any rules or grammar as such, these approaches tend to perform very badly.

Hence, extensive pre-processing and feature engineering need to be done specific to the text genre, language and the problem statement using other NLP tools Since these tools are not 100% accurate, the loss in accuracy in the pre-processing steps will in turn affect the overall accuracy of the sentiment analysis task. Hence, pre-processing steps, especially feature extraction steps, need to be carefully managed. Although good accuracy values have been reported in literature and these algorithms seem to work well for a long time, there is a large scope of improvement, which cannot be overlooked.

3 Introduction to Deep Learning Paradigm

Neural networks recently have become a very popular topic of research in the field for natural language processing, including sentiment analysis. Neural networks are proving useful in solving almost any machine learning classification problem. The only adjustment required is defining its architecture — number of hidden layers to be used, number of hidden units to be present in each layer, activation function for each node, error threshold for the data, the type of inter-connections, *etc.*

Once a suitable neural network architecture is designed for the problem at hand, a solution to the classification problem can be obtained using deep learning models. The only demand for deep learning models is enough training data and enough time and resources to train the network for classification.

> *Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using model architectures, with complex structures or otherwise, composed of multiple non-linear transformations.*
>
> *– defined by Deng and Yu* (Deng and Yu, 2014)

Clearly, a traditional machine learning algorithm can be designed using deep learning but not necessarily vice-versa. This is because neural networks are capable of capturing very complex characteristics of data without any significant involvement of manual labour as opposed to the machine learning systems. Deep learning uses deep neural networks to learn good *representations* of the input data, which can then be used to perform specific tasks.

3.1 Advantages of Deep Learning

Although the traditional machine learning algorithms like SVM have shown good performance in various NLP tasks for the past few decades, they have some shortcomings and deep learning models have the potential to overcome these limitations to a large extent. Some of the advantages of deep neural networks are:

- A strength of the deep learning models is *no demand for carefully optimized hand-crafted features*. Instead of the features, they take *word embeddings* as input which contain context information, and the intermediate layers of the neural network learn the features during the training phase itself. This means that a necessity, which is at the base of the traditional classification models, is no longer required for the functioning of deep learning models.

- Deep learning allows good *representation learning*. While feature representations of input text can be learned automatically from the training data for a particular task, the representations of the words, containing context information, can be learned from raw corpus in an unsupervised manner. This disregards any need for manual construction of appropriate features or word information.

- As discussed before, sentiment analysis consists of varied problem statements. The ability to adapt to the task variations with very small changes in the system itself adds a feather in the cap of the deep learning paradigm.

3.2 Basic Neural Networks

Neural Networks play an important role in machine learning and cognitive science. These have been widely used in the field of image processing and pattern recognition. Recently, they are becoming popular for solving Natural Language Processing problems. A neural network can be used to learn the word embeddings as well as in turn use them as input for NLP tasks like sentiment classification. The basic structure of a fully-connected neural network, which uses one hidden layer, is shown in Fig 1.

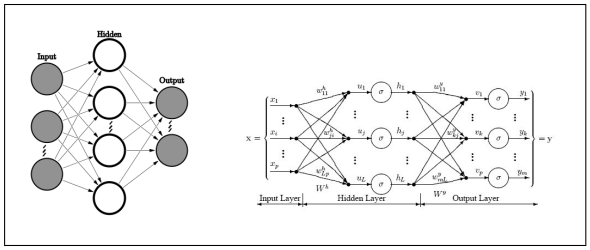

Figure 1: A Fully-Connected Simple Neural Network

The weights on edges are learned by means of back-propagation of errors through the layers of the neural network based on the inter-connections and the non-linear functions. The labeled data is fed to the network a number of times (called *epochs*) for the network to learn the weight parameters until the *error* becomes negligible (in the ideal case). Generally, for experiments, training is done for a fixed number of times to reach a minimum error value when the network does not converge any further.

4 Word Embeddings

Neural networks in NLP, unlike other traditional algorithms, do not take raw words as input, since the networks can understand only numbers and functions. Hence words need to be transformed into *feature vectors*, or in other words *word embeddings* (Mikolov et al., 2013), which capture the characteristics and semantics of the words if extracted properly. The word vectors can be learned by feeding large raw corpus into a network and training it for sufficient amount of time.

Bengio *et al.* (Bengio et al., 2003) presented the first large-scale deep learning model for natural language processing to learn the *distributed representation of words* by using language modeling (Figure 2. The network is trained on a raw corpus, which is expressed as sequence of words. The idea is (a) to associate each word in the vocabulary with a real-valued *word feature vector* of m dimensions, (b) to express the joint probability function of word sequences in terms of the feature vectors of the words occurring in the sequence, and (c) to learn the word feature vectors and the parameters of the probability function simultaneously.

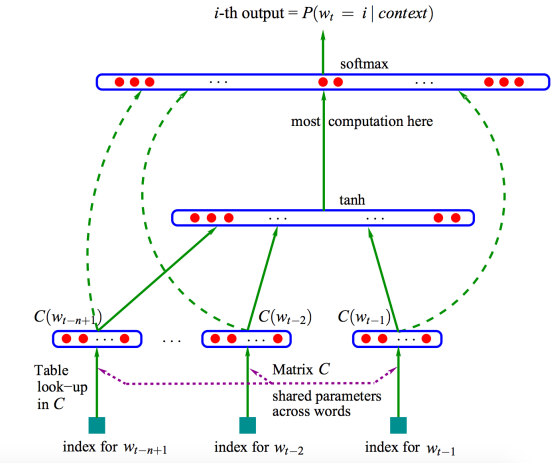

Figure 2: Bengio's Neural Network Model (Bengio et al., 2003)

Each word embedding may be of any dimensionality as the user wishes. Higher dimensionality implies more information captured but on the other hand incurs higher computational expense. Hence, a trade-off is to be chosen to balance both. Google has released pre-trained vectors trained on part of *Google News* dataset (about 100 billion words)[1], which can be used by researchers. The model contains 300-dimensional vectors for 3 million words and phrases. These can then be used as inputs into the neural networks for any NLP tasks.

The quality of the word vectors is defined by how the vectors distinguish between dissimilar words and are close for similar ones. The closeness of word vectors is generally determined by *cosine distance* (Mikolov et al., 2013). For exam-

[1]available at `https://code.google.com/archive/p/word2vec/`

ple, embeddings of words *cat* and *dog* are closer to that of *horse* as compared to that of *play*. In addition, certain directions in this induced vector space are observed to specialize towards certain semantic relationships (unlike one-hot encoding) like gender, verb tense and even country-capital relationships between words (Mikolov et al., 2013). For example, following relationships are captured by the pre-trained Google vectors:

(<*word*> implies corresponding word vector)

<*king*> + <*queen*> = <*boy*> + <*girl*>

<*playing*> + <*played*> = <*trying*> + <*tried*>

<*Japan*> + <*Tokyo*> = <*Berlin*> + <*Germany*>

Hence, this substantiates the utility of these vectors, *i.e.*, word embeddings as features for several canonical NLP prediction tasks like part-of-speech tagging, named entity recognition and sentiment classification (Collobert et al., 2011).

5 Neural Network Architectures for NLP

Generally, for NLP tasks, we tend to use the *Window Approach* (Collobert et al., 2011). This method assumes that the tag to be assigned to a word in a sentence depends upon its neighbouring words. Hence, a fixed window size (additional hyper-parameter) is chosen and this amount of words is fed into the network to tag the middle word (Fig. 3). The feature *window* is not defined for border words (start/end) and hence padding is done in the sentences.

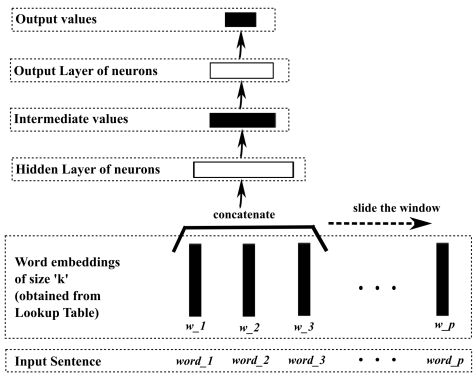

Figure 3: Window-level Neural Network Architecture

The *window-level approach* cannot be applied to macro-text level tasks like sentiment classification because sentiment tag requires the whole sentence to be taken into consideration, whereas in window level approach, only a portion of sentence is considered at a time. Also, for other NLP tasks as well, one word may depend on some word which does not fall in the pre-decided window. Hence, *sentence-level approach* is a viable alternative, which takes the feature vectors of *all* the words in the input text as input (Collobert et al., 2011).

Figure 4 shows the overall structure of such a network which takes the whole sentence as input. The sequence layer can have different structures to handle the *sequence of words* in the text. For sentence classification, since sentences can be of variable size, there is a *pooling layer* after the *sequence layer*, which is a feature map of a fixed size. This is then fed into fully connected layers of neurons to produce a single tag for the whole sentence.

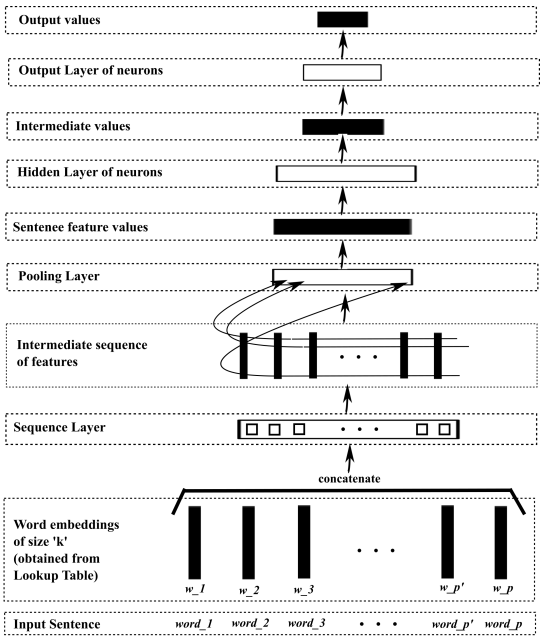

Figure 4: Sentence-level Neural Network Architecture

The number of layers, the number of units in each layer, the structure of the sequence layer, the dimensionality of the word vectors, the interconnections and the activation functions are some of the *hyper-parameters* of the neural network model, which need to be tuned to achieve the best

performance for a particular task.

5.1 Convolutional Neural Networks (CNN)

Collobert et al. proposed a unified neural network architecture (Collobert et al., 2011) which can be applied to numerous Natural Language Processing tasks like Part-Of-Speech Tagging, Parsing, Chunking, Semantic Role Labeling and Named Entity Recognition. The architecture, known as CNN (Convolutional Neural Network), takes concatenated word vectors of the text as input and involves *convolutional* and *max-pooling* layers prior to the general neural network framework.

Convolution Layer: It is a sort of generalization of window approach where a window of fixed size is moved over the sentence and the weight matrix is same for each sequence. One feature vector is obtained by *convoluting* over each sequence. This layer is meant to extract local features from sequence of words in the sentence. The network can have a number of window sizes and a number of weight matrices, each forming one channel.

Max-Pooling Layer: The length of the output of convolution layer depends on length of the input sentence and the number of channels used. To establish uniformity in the size of the *sentence vector*, max-pooling layer is used to select the maximum value for each feature across all windows. This is preferred over simple averaging because for the classification, all words do not contribute equally; their relative significance is captured by the max-pooling layer. Now the *global feature vector* size for the sentence is proportional to that of individual words and to the number of channels used, *i.e.*, its length is constant for all sentences of varying length.

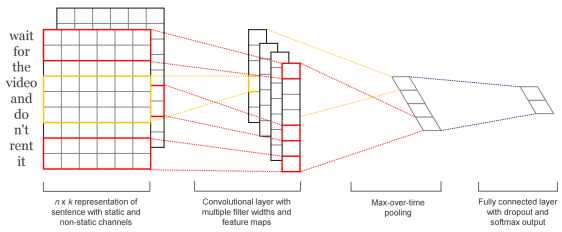

Figure 5: Model architecture with two channels in CNN (Kim, 2014)

The "*sentence vector*" is then fed into a fully connected neural network with 0/1/2/.. hidden layers and activation functions like softmax or sigmoid to ultimately reach the output layer whose size is equal to the number of labels. A model architecture of CNN is shown in Figure 5 (Kim, 2014).

5.2 Recursive Neural Tensor Networks (RNTN)

A recursive neural tensor network (RNTN) (Socher et al., 2013) is a kind of deep learning model in which the same set of weights is applied recursively over a structure (*e.g.* tree), to produce a structured or a scalar prediction over variable-length input, by traversing the given structure in topological order. The RNTN model takes as input, the word vectors and the *parse tree* of the text, and then computes vectors for the nodes in the tree using a single tensor-based composition function. This model is a modification over the *recursive neural networks* which uses a simple weight matrix shared across all the nodes.

The input sentence is first converted into a constituent parse tree. The leaves of the tree are represented by the corresponding word vectors. The vector representations of the parent nodes are computed in a *bottom-up fashion* using a tensor-based compositionality function g, which uses the vectors of the children nodes as features for a classifier at the parent node. This function, which is shared over the whole network, varies for different models based on the task to be performed. An example of the structure of a recursive neural tensor network for sentiment classification is shown in Figure 6.

Recursive neural tensor networks are a powerful model to capture the *meaning of longer phrases in a principled way* (Socher et al., 2013) and understand the compositionality in complex tasks such as sentiment detection. It is able to accurately capture the effects of several phenomena like negation and their scope at various tree levels. The only extra pre-processing step which this model requires is construction of a parse tree for the input sentence and in cases like tweets which do not follow any syntax or grammatical rules, the efficiency of the model is affected adversely due to the demand for correct parse tree construction.

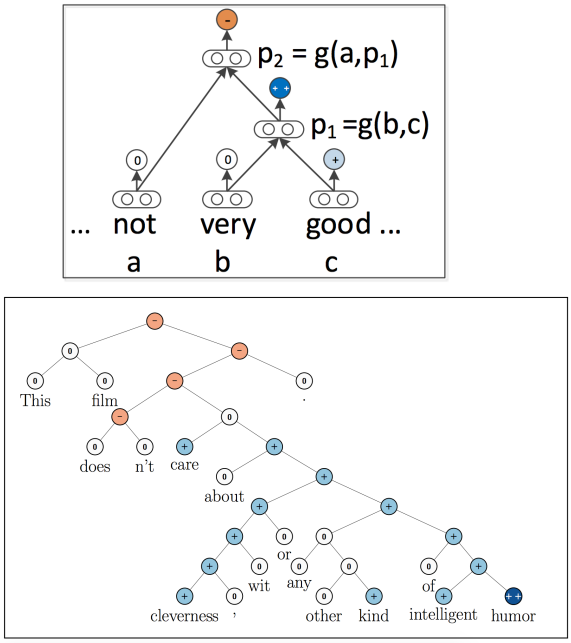

Figure 6: Example of Recursive Neural Tensor Network (Socher et al., 2013)

5.3 Recurrent Neural Networks (RNN)

Recurrent Neural Network Models (Mikolov et al., 2010) are a form of neural networks which do not depend on the window size to work for Natural Language Processing tasks. RNN is capable of conditioning the network on all the previously seen inputs (words in case of a sentence). In addition to dependency on the current input, the value of each hidden layer unit also depends on its previous state, thereby propagating the effects of words over the sentence (Fig 7).

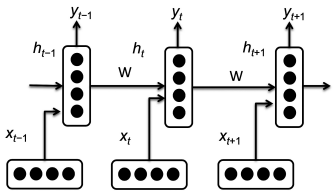

Figure 7: A Recurrent Neural Network with three time steps

The word vectors for each word is fed into the network one by one and the effect of each word is carried on till the end of the sentence, thereby ensuring that the dependency of each word on all other words is captured through activations of neurons and back-propagation on weight matrices. The goal of an RNN implementation is to allow propagation of context information through far-away time-steps.

RNN model works by propagating weight matrices over the time-steps. However, this creates anomalies which are not acceptable in practice. Intuitively, one should be able to predict a word more accurately given more context (i.e., more number of words preceding this) as compared to lesser context. However, RNN tends to perform the opposite due to the problems of *Vanishing Gradient* and *Gradient Explosion* problems. The gradient signal can result in being multiplied a large number of times (as per the number of time-steps) by the weight matrix associated with the interconnections between the neurons of the recurrent hidden layer. This implies that, the magnitude of weights can have a strong impact on the learning process.

Vanishing Gradient Problem: While back tracking a few steps, the gradient may go on getting smaller and smaller due to small weight values (less than 1.0) and a time may come when it is close to zero. This means the effect between the far away words in a sequence becomes negligible or zero.

Gradient Explosion Problem: The gradient value may explode over the time steps, if the weight values are large (majority being greater than 1.0). This means the weight matrices overpower the word vector values in the learning process, which is again not desirable.

Hence, RNN model, due to its unavoidable limitations, is inappropriate for natural language processing tasks. Though it has its merits, the unreliability of the back-propagation of information makes it less favourite among the researchers. The necessity is some modification in the network so as to preserve the pros while mitigate the cons. Some of the improved versions are *Gated Recurrent Units (GRU)* and *Long Short Term Memory Models (LSTM)*.

5.4 Long Short Term Memory (LSTM) models

Long Short Term Memory networks (Hochreiter and Schmidhuber, 1997) are a modified version of the recurrent neural networks but with much more complicated activation units. The key element of an LSTM model is a *memory cell*. Here, information is stored in two ways (hence the name Long Short Term Memory):

Short-term Memory as activations of the neurons which capture the recent history,
Long-term Memory as weights which are modified based on back propagation

This model allows retention of information over a much longer period (more than the usual 10-12 steps as in case of RNNs) through the use of the memory cell and hence produces appreciable results when applied to NLP tasks.

The internal units of an LSTM model are shown in Fig 8. The network architecture is very complex and its structure can be broken down into certain stages [2]:

1. ***Input Gate*** : It uses the input word and the past hidden state to determine whether or not the current input is worth preserving.

2. ***New Memory Generation*** : It uses the input word and the past hidden state to generate a new memory which includes aspects of the new word.

3. ***Forget Gate*** : It uses the input word and the past hidden state to make an assessment on whether the past memory cell is useful for computation of the current memory cell.

4. ***Final Memory Generation*** : It first takes the advice of the forget gate and accordingly forgets the past memory; it then takes the advice of the input gate and accordingly gates the new memory and lastly it sums these two results to produce the final memory.

5. ***Output/Exposure Gate*** : It makes the assessment regarding what parts of the memory needs to be exposed/present in the hidden state.

So the LSTM model (Gers et al., 2000) consists of memory cells to maintain information over long periods of time, and the gating units to decide what information to store and when to apply that information. This division of responsibilities enables the network model to remember salient events over arbitrarily long period of time.

The task of sentiment analysis is generally assigning a label to the whole sentence and not to the individual words. Hence, the outputs produced at each time-step, i.e, with each input word, need to be reconciled to finally get a single label for whole sentence. One way is to have a number of *LSTM cells* for each of which the outputs are passed through a *mean-pooling layer* before going through logistic regression.

[2] http://cs224d.stanford.edu/lecture_notes/LectureNotes4.pdf

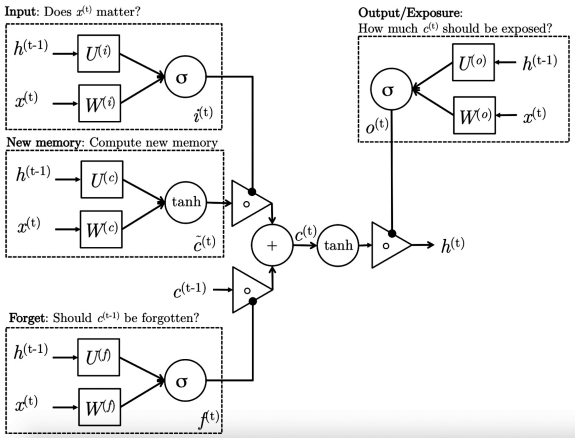

Figure 8: Detailed Internals of an LSTM

Hence, in spite of the fact that the LSTM network is very complicated and has its own disadvantages like huge computational complexity, the network offers promising results since it is capable of taking into account the whole sentence as context to generate results.

6 Experimental Results

We discussed some basic machine learning techniques from the two paradigms so far. However, for performing classification tasks efficiently, researchers are frequently coming up with *modified systems based on these architectures* and the experimental results reported in literature reflect the viability of the different techniques. Although extensive research has been done using traditional models (as the term *traditional* suggests), a good amount of work has been done using deep learning models too in recent years. In fact, the latter is seen to outperform the traditional systems in most cases, thereby establishing its utility in the field of natural language processing, including sentiment analysis.

We tabulate and compare experimental results, reported in literature, on some datasets, for some of the following models:

Traditional Paradigm ::

1. Naïve Bayes (Pang *et al.*, 2002) : With unigram and unigram+bigram as features (Pang et al., 2002)

2. SVM (Pang *et al.*, 2002) : With unigram and unigram+bigram as features (Pang et al., 2002)
3. NBSVM (Wang and Manning, 2013) : SVM with Naïve Bayes unigram and unigram+bigram features (Wang and Manning, 2012)
4. Tree-CRF (Nakagawa et al., 2010) : Dependency tree-based sentiment classification using CRFs with hidden variables (Nakagawa et al., 2010)

Deep Learning Paradigm ::

5. CNN-rand, CNN-static, CNN-nonstatic, and CNN-multichannel (Yoon Kim, 2014) : Four variants of the Convolutional Neural Network based on the variations in the usage of word vectors (Kim, 2014)
6. DCNN : Dynamic Convolutional Neural Network with k-max pooling, as given in (Kalchbrenner et al., 2014) (Kalchbrenner *et al.*, 2014)
7. LSTM, Bidirectional LSTM, 2-layer LSTM, 2-layer Bidirectional LSTM, Tree-LSTM : Long Short Term Memory Models and their variants, as given in (Tai et al., 2015) (Socher *et al.*, 2015)
8. RNN : Recurrent Neural Networks, as given in (Socher et al., 2013) (Socher *et al.*, 2013)
9. MV-RNN : Matrix-Vector Recursive Neural Network with parse trees, as given in (Socher et al., 2013) (Socher *et al.*, 2013)
10. RNTN : Recursive Neural Tensor Network with tensor-based feature function and parse trees, as given in (Socher et al., 2013) (Socher *et al.*, 2013)
11. RAE : Recursive Autoencoders, as given in (Socher et al., 2011) (Socher *et al.*, 2011)
12. RAE-pretrain : Recursive Autoencoders with pre-trained word vectors from Wikipedia, as given in (Socher et al., 2011) (Socher *et al.*, 2011)
13. DRNN : Deep Recursive Neural Networks, as given in (Irsoy and Cardie, 2014) (Irsoy and Cardie, 2014)
14. VecAvg : A model that averages neural word vectors and ignores word order, as given in (Socher et al., 2013) (Socher *et al.*, 2013)
15. Paragraph-Vec : Logistic regression on top of paragraph vectors, as given in (Le and Mikolov, 2014) (Le and Mikolov, 2014)

6.1 Performance of Models on Stanford Sentiment Treebank Dataset

The sentiment treebank dataset[3] introduced by Stanford (Socher et al., 2013) contains *11,855 sentences* (movie reviews along with their parse trees giving rise to *215,154 phrases*. These are annotated with sentiment scores ranging between 0 and 1 where 0 means most negative and 1 means very positive. Experiments are performed on this dataset (using the standard train/development/test splits) for two subtasks of classification:

Fine grained sentiments: The dataset is categorized under 5 labels as – *very negative* $(0 - 0.2)$, *negative* $(0.2-0.4)$, *neutral* $(0.4-0.6)$, *positive* $(0.6 - 0.8)$ and *very positive* $(0.8 - 1)$.
The train/development/test splits of 8544 / 1101 / 2210 are used here.

Binary sentiments: The dataset is categorized under 2 labels as – *negative* $(0 - 0.4)$ and *positive* $(0.6-1)$ with neutral sentences removed.
The train/development/test splits of 6920 / 872 / 1821 are used here.

We show a comparison of the performance of the three basic traditional models : Naïve Bayes with unigram features, Naïve Bayes with unigram+bigram features and SVM with unigram features[4] (Pang et al., 2002) with the accuracy values for some of the deep learning models in Table 1 (the results are taken from papers cited alongside the classification models in the table).

6.2 Performance of Models on Benchmark Datasets

We show experimental results on the following benchmark datasets:

1. *MR*[5] : Movie reviews with one sentence per review; classification involves detecting positive/negative reviews (Pang and Lee, 2005)

[3] available at `http://nlp.stanford.edu/sentiment/index.html`
[4] Experimental results are taken from (Socher et al., 2013)
[5] available at `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

Models	Fine-grained (5-class)	Binary (2-class)
Traditional Paradigm		
Naïve Bayes (unigrams)	41.0	81.8
Naïve Bayes (uni-bigrams)	41.9	83.1
SVM (unigrams)	40.7	79.4
Deep Learning Paradigm		
CNN-rand (Kim, 2014)	45.0	82.7
CNN-static (Kim, 2014)	45.5	86.8
CNN-nonstatic (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DCNN (Kalchbrenner et al., 2014)	48.5	86.8
LSTM (Tai et al., 2015)	45.8	86.7
Bidirectional LSTM (Tai et al., 2015)	49.1	86.8
2-layer LSTM (Tai et al., 2015)	47.5	85.5
2-layer Bidirectional LSTM (Tai et al., 2015)	46.2	84.8
Constituency Tree LSTM (Tai et al., 2015)	**50.6**	86.9
RNN (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
RAE-pretrain (Socher et al., 2011)	43.2	82.4
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
VecAvg (Socher et al., 2013)	32.7	80.1
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8

Table 1: Results (Accuracy values in %) of different machine learning approaches for sentiment classification on Stanford Sentiment Treebank dataset

2. *MPQA*[6] : Opinion polarity detection subtask of the MPQA dataset (Wiebe *et al.*, 2005)

3. *CR*[7] : Customer reviews of various products (cameras, MP3s etc.); task is to predict positive/negative reviews (Hu and Liu, 2004)

4. *Subj*[8] : Subjectivity dataset where the task is to classify a sentence as being subjective or objective (Pang and Lee, 2004)

The summary statistics of these datasets are given in Table. The comparison of the traditional models with some of the deep learning models are shown in Table 2 which reports the average accuracy values of 10-fold cross-validation experiments performed on the datasets (the results are taken from papers cited alongside the classification models in the table).

[6]available at `http://www.cs.pitt.edu/mpqa/`

[7]available at `http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`

[8]available at `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

7 Conclusion

The classification performance of the different models on the different datasets gives a rough idea of the utility of the different models in different scenarios for sentiment classification. On most of the datasets, the performance of the deep learning models, especially CNN and LSTM variants, is over-whelming. Convolution Neural Networks (Kim, 2014) give the best accuracy for binary sentiment classification on Stanford Sentiment Treebank dataset (*CNN-multichannel* : **88.1%**), Movie Review (MR) dataset (*CNN-nonstatic* : **81.5%**), MPQA dataset (*CNN-static* : **89.6%**) and Customer Review (CR) dataset (*CNN-multichannel* : **85.0%**) while *Constituency Tree LSTM* model (Tai et al., 2015) performs the best for fine-grained sentiment classification of Stanford Sentiment Tree-

Models	MR	MPQA	CR	Subj
Traditional Paradigm				
Naïve Bayes (unigrams) (Wang and Manning, 2012)	77.9	85.3	79.8	92.6
Naïve Bayes (uni-bigrams) (Wang and Manning, 2012)	79.0	86.3	80.0	**93.6**
SVM (unigrams) (Wang and Manning, 2012)	76.2	86.1	79.0	90.8
SVM (uni-bigrams) (Wang and Manning, 2012)	77.7	86.7	80.8	91.7
NBSVM (unigrams) (Wang and Manning, 2012)	78.1	85.3	80.5	92.4
NBSVM (uni-bigrams) (Wang and Manning, 2012)	79.4	86.3	81.8	93.2
Tree-CRF (Wang and Manning, 2012)	77.3	86.1	81.4	-
Deep Learning Paradigm				
CNN-rand (Kim, 2014)	76.1	83.4	79.8	89.6
CNN-static (Kim, 2014)	81.0	**89.6**	84.7	93.0
CNN-nonstatic (Kim, 2014)	**81.5**	89.5	84.3	93.4
CNN-multichannel (Kim, 2014)	81.1	89.4	**85.0**	93.2
RAE (Socher et al., 2011)	76.8	85.7	-	-
RAE-pretrain (Socher et al., 2011)	77.7	86.4	-	-

Table 2: Results (Accuracy values in %) of different machine learning approaches on MR, MPQA, CR and Subj Datasets

bank dataset (**50.6%**) – more than 8% performance gain over the traditional models. This clearly exhibits the efficacy of deep learning models for sentiment analysis.

Neural networks are versatile, owing to the fact that they avoid task-specific engineering, thereby disregarding any sort of prior knowledge related to the data. Hence, man-made features, which have to be carefully optimized to produce proper results in case of traditional models, are not at all necessary while using deep learning models. Although the widely popular traditional models like Naïve Bayes and SVM have proved useful for so long, the potential of the deep learning paradigm cannot be overlooked. In fact, the latter promises to perform much better than the former, with minimal constraints on the task or data for sentiment analysis.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. 2014. An impact analysis of features in a classification approach to irony detection in product reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49.

Stanley F Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for me models. *IEEE transactions on Speech and Audio Processing*, 8(1):37–50.

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

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Li Deng and Dong Yu. 2014. Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational*

Linguistics: Human Language Technologies: short papers-Volume 2, pages 581–586. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Edwin T Jaynes. 1957. Information theory and statistical mechanics. *Physical review*, 106(4):620.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on EMNLP*, page 17461751.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. 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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86.

Jason D Rennie, Lawrence Shih, Jaime Teevan, David R Karger, et al. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC).

Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. 2003. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. 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, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.