Sentiment Analysis of generalized text through LSTM Neural Networks

Prem Shah
School of Engineering & Applied Science
Ahmedabad University
Email: prem.s.btechi13@ahduni.edu.in

Rohit Mathur
Exponentia Datalabs Pvt. Ltd.
Mumbai, India
Email: rohit.mathur@exponentiadata.com

***Abstract*—Every single day, people post millions of comments and reviews regarding various things online. The automation of evaluation of people's opinions can make a big difference to companies in order to improve their products and services. Sentiment Analysis and specifically aspect-based sentiment analysis is an important research topic nowadays. This report presents an approach to build a sentence level implementation of aspect based sentiment analysis using phrase chunking and Long Short Term Memory Recurrent Neural Networks. The report outlines and explains the various approaches taken to improve the model and make it more robust. Stanford Movie Review Dataset has been used to train the model. Additionally, internet service provider(ISP) reviews were sourced from a public review website. Furthermore, it explains the current model and algorithm. The report also explains the results and challenges encountered during implementation of the various approaches.**

***Keywords:* Sentiment Analysis, Neural Networks, RNN, LSTM-RNN, Aspect Extraction**

I. Introduction

With the use of social media and the internet growing leaps and bounds in the last few decades, there has been an exponential growth in the availability of information. The internet has become a medium to communicate and share thoughts and opinions on everything. Mediums such as Twitter, Facebook etc. act as voices of people when it comes to products and services. In such an era, information from such sources acts as an important resource of public opinion towrds anything and everything. An important aspect of opinion is sentiment which indicates the polarity of a review or a document being positive, negative or neutral.

Sentiment analysis has several approaches. There are majorly three types of sentiment analysis : document level, sentence level or aspect level. Document level & sentence level give the overall sentiment of a sentence or a paragraph but they do not identify for what aspect is the sentiment expressed. Aspect level Sentiment Analysis focuses on what the sentiment is as well as for whom or what is the sentiment for [6]. It may be easy to extract sentiment from a simple sentence formation but if the sentence contains two or more entities and different sentiments for each entity, it becomes much more difficult to figure out each individual sentiment.

The aim of the project was to build a simple sentence level sentiment analysis model that was to be integrated into a company product and improve it further by inculcating various approaches to make it more robust and accurate. The first part of the implementation was to build a sentiment analysis model on a single domain using Logistic Regression/ Naive Bayes algorithms. The second part consists of an implementation through a special type of recurrent neural networks known as Long Short Term Memory [7]. We study the performance of all the three algorithms namely Naive Bayes, Logistic Regression & Long Short Term Memory Recurrent Neural Networks (LSTM-RNN) [7] to perform sentiment analysis. Subsequently,to improve the model further, an aspect level implementation has also been tried through two ways part of speech tagging and phrase chunking. This implementation tries to extract the subject verb and predicate through noun and verb phrase extraction out of each sentence which helps in finding out what the user/review is referring to.

This report outlines how the implementation started and how it was improved due to several drawbacks in each of the implementations. It is divided into three major parts. The first part introduces the two main types of sentiment analysis first and then explains how both of them were combined in the implementation. The second part discusses the results of both the model and the algorithm for phrase chunking. The third part explains the challenges and future prospects of this implementation before concluding the report.

II. Knowledge(Lexicon) Based vs. Learning Based Sentiment Analysis

There are two main approaches one can take to extract sentiment out of a statement. One is the knowledge based approach which uses a dictionary of scored words which are either annotated as positive/negative or have a score within a defined range assigned to each of them. It calculates the overall sentiment based on the number of negative to positive words or as a sum of scores of individual words. For this, there are many dictionaries publicly available for use in research as well as commercial. VADER [1], SentiWordNet [8], Bing Liu's Opinion Lexicon [12] are some of the well-known lexicons in use today for sentiment analysis research. VADER [1] is specifically built for social media posts and comments and it considers a lot of slang and colloquial words along with emoticons and it assigns human-rated scores between -4 to +4 for each word. SentiWordNet [8] assigns a label to each word based on its polarity.

The second method is learning based classification wherein datasets are trained using machine learning algorithms and then the model is used to predict sentiment of previously unseen data. The most commonly used learning classifiers for sentiment analysis are Naive Bayes, Support Vector Machine and Logistic Regression. Usually, in learning based methods, text is preprocessed to remove unnecessary stopwords and punctuation marks. After that, feature selection is performed using methods like tf-idf weighting, word-vector representation etc. Following this, the selected features are input into the classifier to train the model and then it is tested on unseen data. AYLIEN [15] is one such text analysis API which is gaining traction for Aspect based sentiment analysis.

Majority of the sentiment analysis methods and lexicons today are domain specific i.e. they usually perform the best when the data is from a particular domain like retail, product or social media posts etc.[1][11]

The model presented in this approach tries to combine the advantages of both the methods the approaches while at the same time trying to mitigate the disadvantages of the same.

III. Approaches taken

The project began with a small implementation of learning based sentiment analysis as described in Section A. As new problems and complexities occured, the model was refined and modified to overcome them. The approaches are described below in order of their implementations. The subsequent section describes the current approach.

A. Model - based Sentiment Analysis

The first implementation of sentiment analysis was based on a simple model where data is preprocessed and converted into numerical features, eventually being trained on two different algorithms, Naive Bayes and Logistic Regression.

1) Preprocessing: Text preprocessing is one of the most important steps when it comes to text classification. There are many unnecessary symbols and words which do not signify any meaning and are not needed as features in our training dataset. In this project preprocessing was carried out to remove stopwords, remove punctuations and also to remove email salutations and other unnecessary text. Also removed were twitter mentions and tags in order to make the data more robust because the model was going to be used on social media data of clients.

2) Feature Selection: After the data is preprocessed, feature selection is carried out by two methods. First the dataset is converted into a matrix of token counts where the rows are the datapoints whereas the columns are unique words.

Subsequently the term frequency and inverse document frequency (tf-idf weight) of each word is calculated. The inverse document frequency signifies how important a word is in a sentence. It is calculated by dividing the total number of documents by the number of documents containing that word and taking its logarithmic value. This tends to give less importance to common words like a, at, the etc. while giving preference to more significant words. [4]

3) Classification: In the next step, the feature selected data was passed through the classifier to train the model and test it

4) Drawbacks: This implementation had several drawbacks. The most significant drawback was that it considered all the words as independent. 'Not good' was treated as postive because good might have had a better inverse document frequency as compared to not and hence given more weight. Also, if the training data is not complete meaning if the testing data encounters something significantly different from what it has been trained on, the model fails to predict that correctly.

B. Negative n-gram Dictionary + Model

In order to curb the above mentioned drawback, a combination of lexicon and learning based methods was implemented. A dictionary of highly negative context specific bigrams and trigrams is created through human effort. Then instead of passing all the testing data through the model, each data point is searched for the occurence of those trigrams or bigrams. If an exact match was found, the data point is classified as negative and after the whole testing set passes through the dictionary, the remaining unclassified datapoints are passed through the trained model to be evaluated. Table I shows some example n-grams from the dictionary.

TABLE I
Negative Trigram Examples from the defined dictionary

Negative n-gram dictionary
bad customer care
bad customer experience
no internet access
never resolve issues
pathetic screenplay
horrible screenplay
bad acting
connection goes off
very low standard
annoy the customers
harass the customers
internet doesnt work
horrible customer service
bad support
horrible movie

This approach significantly increased the number of true negatives being classified by the model and also improved the time taken to run the model but the accuracy remained the same. Unfortunately, it was increasing the false negatives at the same time in turn keeping the accuracy similar to what the previous model achieved. Also, since the dictionary has to be domain specific, it becomes too burdensome to maintain such a knowledge base for each domain that this model was going to be applied to.

C. LSTM-RNN

' Long Short Term Memory[7] are a special type of Recurrent Neural Networks which specifically try to mitigate the vanishing gradient problem which occurs in Recurrent Neural Networks. Recurrent Neural Networks were selected because they have loops which theoretically allow information to pass through their states thus maintaining sequences. But as

the gap in sequence grows, learning long term dependencies becomes difficult for RNNs through gradient descent [17]. Hence, LSTM-RNNs are a special type of networks which specifically handle the long term dependency problem. The main difference between LSTM-RNNs and Naive Bayes or Logistic Regression is that it is able to learn dependenceies between words and depending on the number of layers, it can extract more features.

1) LSTM module: The below figure shows the structure of a repeating module in an LSTM layer. It has three gates to facilitate storage and transfer of information namely input gate, forget gate and output gate. This gating mechanism differentiates an LSTM-RNN from a simple Recurrent Neural Network. The input gate decides what values have to be updated. Simultaneously a tanh layer creates a vector of candidate values that could be added to the state. After this, the forget gate is multiplied to the old state to forget previous values and the input gate values multiplied to the candidates generated and is added to the cell state. The input gate values scale how much each candidate value has to be updated. Subsequently, the output layer is passed through a sigmoid function which decides what parts of the data we are going to output and then it is multiplied to the candidate set which itself is run through a tanh layer to restrict outputs between 1 & -1. [10].

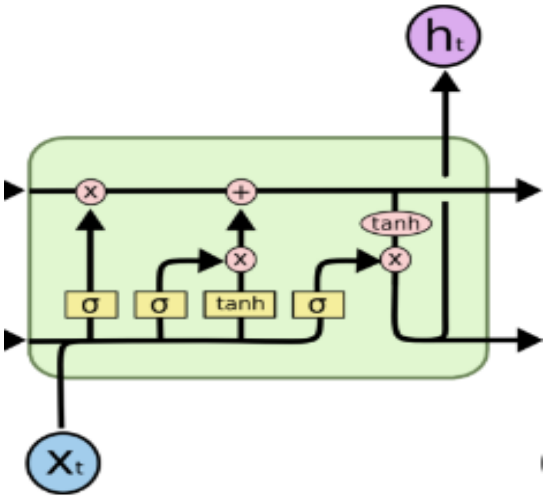

Fig. 1. Structure of a module in an LSTM layer [10]

2) Our Architecture: Firstly, the data is preprocessed and converted into sequences of numbers where the word of rank *i* has index *i*. All the reviews are converted into number sequences through this method. Also, the data is restricted to 40000 words. We do not include the first 10 words from that because usually the most common words like a, at, the etc. are not relevant features for classification. Here, even though not is a very common word it is kept in the training corpus since it depicts contradictory sentiment in many cases.

Fig. 2 shows the architecture used for the sentiment analysis model. The length of the sentences is set to a maximum of 170 which is the length of the longest review in the dataset. All the other shorter reviews are padded with zeros to maintain the same input length. The first layer is an embedding layer which converts word sequences into dense vectors of size 128. There are two LSTM-RNN layers with number of cells set to 128 & 56 respectively, and a fully connected Dense layer. All the three layers use dropout [16], a method to prevent the overfitting of data. Dropout uses probability values and according to the probability value (p) defined, the dropout function drops a few neurons from the layer. A sigmoid activation function is used on the dense layer to map the output values between 0 & 1 and have kept the threshold for classifcation at 0.5. Hence any output value below 0.5 will be classified as Negative and any value above that as Positive.

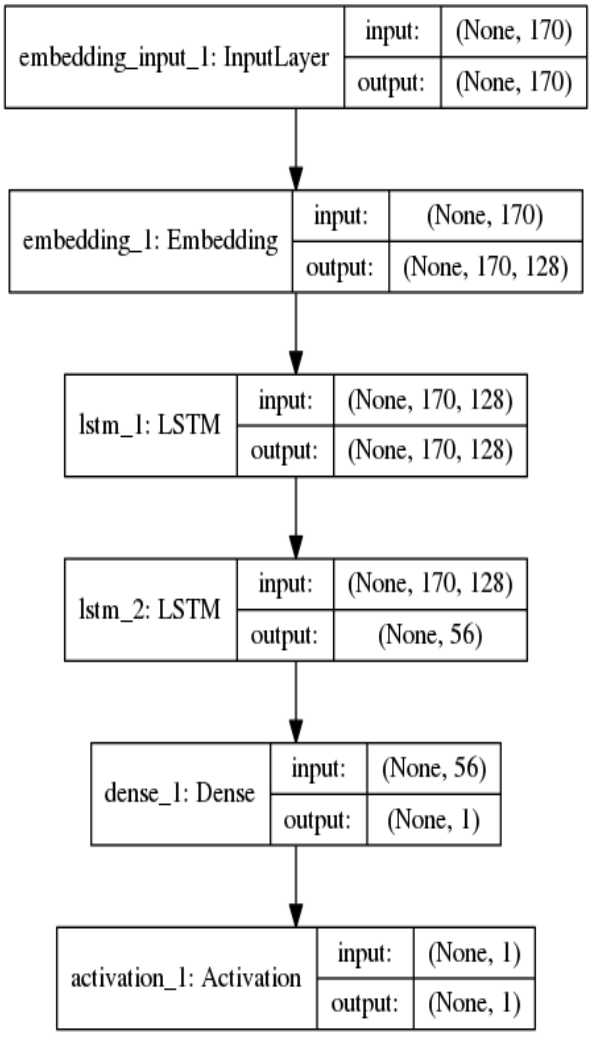

Fig. 2. LSTM-RNN architecture used in model

LSTM-RNNs are chosen to be trained on a large dataset in order to figure out if it can perform well on data from other domains. While it outperforms Logistic Regression and Naive Bayes on a single domain dataset, it performs poorly when tested on data from a different domain. The results are as shown in Section 5.

D. Aspect Extraction + LSTM-RNN

Even though the overall accuracy of the model improved significantly, the above model had one major flaw. If a sentence had two different sentiments, the model was unable to correctly predict both since the model was trained with reviews having a single sentiment.

For example, *'The movie was good but the songs were bad'*, there are two different sentiments for two different aspects. movie : positive & songs : negative. But the model classified this review as negative which is incorrect. Hence, the model was further improved by integrating an algorithm which tags the sentences and separates them based on two criterias. We use SENNA [15] tagger for POS tagging.

1) Separate the review based on full stops excluding exceptions such as etc., et al. and treat each sentence as having a different sentiment.
2) If a co-ordinnating conjunction is present which signifies opposite meaning (but,though)then extract two sentences before and after the conjunction. As we can see in the sentence in italics above, this method gave out two sentences *The songs were good, the movie was bad*. In this way, the model was able to extract multiple entities and also give their own individual sentiments.

This algorithm performs well on simple noun aspect extraction but failed in a number of complex cases. For example *'quality of service'* is a noun phrase but this method fails to link quality and service by the injunction since it extracts only nouns. Also, *'customer service at Kotak Mahindra'* is a noun phrase. But since this algorithm extracts only individual nouns, it performs badly when noun phrases are encountered where relations are important.

IV. Current Approach - Phrase Chunking + LSTM-RNN

The previous method of aspect extraction in which we were tagging the words and extracting nouns proves to be very inaccurate when a combination of nouns is the subject. For example, *the employees of Exponentia Datalabs*. In this example, the previous method will extract both employees as well as Exponentia Datalabs, but will not be able to correlate both as explained previously.

Phrase chunking, a natural language process separates a meaningful sentence into parts such as verb, adjecteive, noun etc. We use phrase chunking in our model to extract noun phrases, verb phrases as well as adjectives which complement the former. Su-Nam Kim et al. in their paper [3] have described an approach for extracting noun phrases. Extending the regular expressions defined in the paper to extract noun phrases, we developed a set of regular expressions to try and extract verb phrases, adjectives and whole phrases by creating a dependency tree of the sentence.

Algorithm 1 describes the whole process of extracting phrases and noun phrases from a review. Step 1 separates the sentences by fullstops, exclusions being etc., et al. and others. Then it checks for contradicting co-ordinating conjunctions in the sentences. If present, the grammar rules will extract two phrases from the sentence. Further, the subjects, verb and objects are extracted from the sentence and then the sentence is passed through a trained classifier to extract sentiment from that.

(**Rule1**) NBAR = (NN*|JJ*)*(NN*)
e.g. *complexity, effective algorithm, distributed web-service discovery architecture*
(**Rule2**) NBAR IN NBAR
e.g. *quality of service, sensitivity of VOIP traffic, simplified instantiation of zebroid*

Fig. 3. Noun Phrase rules defined in the paper by Su Nam Kim et al. [3]

Algorithm 1 Algorithm to extract noun, verb and injunction phrases from sentences

Input: Sentence
Output: Noun & Verb Phrases (Subject, Verb, Predicate)

```
input review
for each sentence in review do
    if (injunction or conjunction present) then
        extract phrases
        pass phrases
    else
        pass sentence
    end if
    for each phrase or sentence do
        parsed_sentence = sentence.parse(rules)
    end for
end for
return Noun & Verb Phrases (Subject, Verb & Predicate)
```

The below code fragment grammar rules which are used to parse sentences into trees based on the rules. Fig. 3-8 in the Data & Results section show some example review excerpts parsed by these grammar rules.

Grammar rules defined as Regular Expressions for parsing sentences to generate Dependency Trees

```
ADJ:
{
<JJ.*>*<CC>*<IN>*<JJ.*> |
<DT>*<JJ.*><JJ.*>*
}

NBAR:
{
<DT><NN.*> |
<ADJ>*<NN.*> |
<NN.*>*<NN.*>
}
```

```
NP:
{
<NBAR><RB>*<IN>*<NBAR>* |
<NBAR><VB.*><JJ>
}

VP:
{
<VB.*>| <DT><VB.*>
}

VERBTO:
{
<VP><ADJ>*<TO><VP>
}

PHRASE:
{
<PRP.*>*<NP>*<VP><IN>*<NP> |
<NP><NBAR>*<VP><ADJ><NBAR>* |
<PRP.*><WDT>*<RB><WRB>*<RB>*...
<NP>*<VP>*<ADJ>*<RB>* |
<PRP.*>*<NP>*<NBAR>*<VP><VP>*<NP>*...
<ADJ>*<NBAR>*<NP>*<VP>*|
<PRP.*><RB><RB>*<NP>*<VP>*<ADJ>*<RB>*|
<PRP.*><RB.*>*<VERBTO><PRP.*>*<TO>*...
<NBAR>*<NP>* |
<PRP.*>*<NP>*<NBAR>*<TO>*<VP>*<PRP.*>...
<NP>*<RB>*<ADJ>*<NBAR>*<TO>*<NP>*}
```

V. Data & Results

Data has been obtained from two sources. One is the Stanford Movie Review Dataset [9] and the ISP reviews have been sourced from a public review website known as Mouthshut.com [13]. For Logistic Regression and Naive Bayes, 40000 reviews were used to train the model whereas for LSTM-RNN, training set was 32000 reviews and 8000 were used for validation of the training set to adjust the weights. All the three models were tested on the same testing set of 10000 reviews.

TABLE II
Example Reviews from the Datasets

Domain	Review	Polarity
Movie	If you sometimes like to go to the movies to have fun , wasabi is a good place to start.	Positive
	An absolutely atrocious adaptation of the wonderful children's book.	Negative
ISP	Speed is always good with symmetric download and upload. Also the renewal process is brilliant with mobile application.	Positive
	No way to contact their customer service	Negative

A. Phrase Chunking Results

Figures 4-9 show the result of creating dependency trees through phrase chunking.

Fig. 4 & 5 depict the results of phrase chunking for two sentences whose dependency trees are shown in Fig. 7 & 9 respectively. It extracts probable subjects, objects and predicates from the sentences which help in extracting the aspect.

Fig. 6 has an injunction 'IF' present in the sentence and hence the parser correctly extracts two meaningful phrases 'you sometimes like to go to the movies' & 'Wasabi is a good place to start'. Also, it successfully extracts Wasabi, a good place and movies as nouns.

In Fig. 7, the first noun phrase 'no way' is the subject, 'contact' is the verb phrase and the object is 'customer service'. The object and subject are denoted as noun phrases. Figure 6 shows an example of the excerpt of a movie review. Figure 7 & 8 show the results of extracting probable subject/objects and predicates from the precious results.

```
(S
  (PHRASE
    (NP (NBAR no/DT way/NN))
    to/TO
    (VP contact/VB)
    their/PRP$
    (NP (NBAR customer/NN) (NBAR service./NN))))

Probable Subject/Object(s):

no way
customer service.

Probable Predicate(s):

contact
```

Fig. 4. Results for phrase chunking example - 2

```
>>> runfile('D:/Prem/npchunking.py', wdir='D:/Prem')
(S
  (PHRASE
    I/PRP
    also/RB
    (VP believe/VBP that/IN)
    (NP (NBAR Dark/NNP) (NBAR Angel/NNP))
    (VP will/MD become/VB)
    (NP (NBAR a/DT cult/NN) (NBAR favorite./NN))))

Probable Subject/Object(s):

Dark Angel
cult favorite.

Probable Predicate(s):

believe that
will become
>>>
```

Fig. 5. Results for phrase chunking example - 4

In Fig. 8, we can see the algorithm extracts two different phrases because it contains two different sentences. In the first sentence we can see two noun phrases 'speed' and 'download and upload'. But in the next phrase, due to an incorrect POS tag, mobile is tagged as an adjective instead of a noun and hence 'mobile application' which is theoretically a noun phrase is not detected. Hence, incorrect POS tagging is a

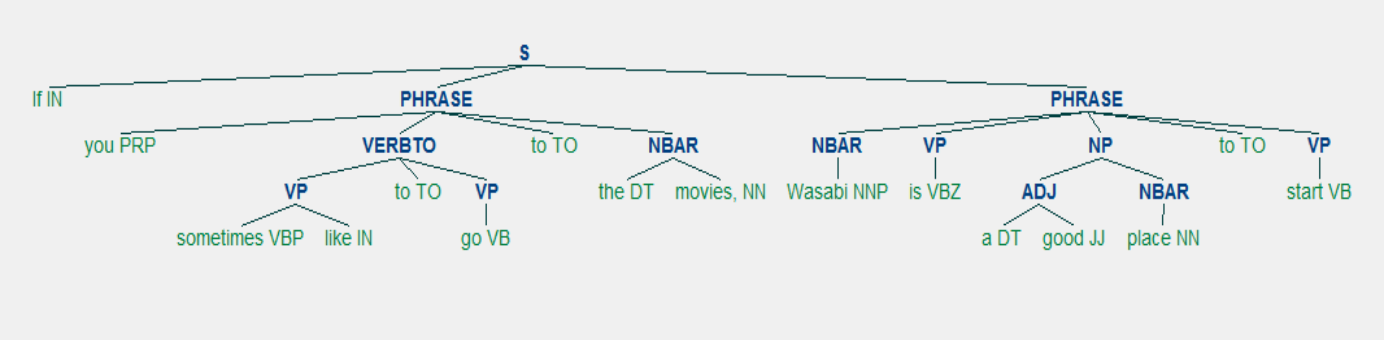

Fig. 6. Phrase chunking example - 1 (with injunction)

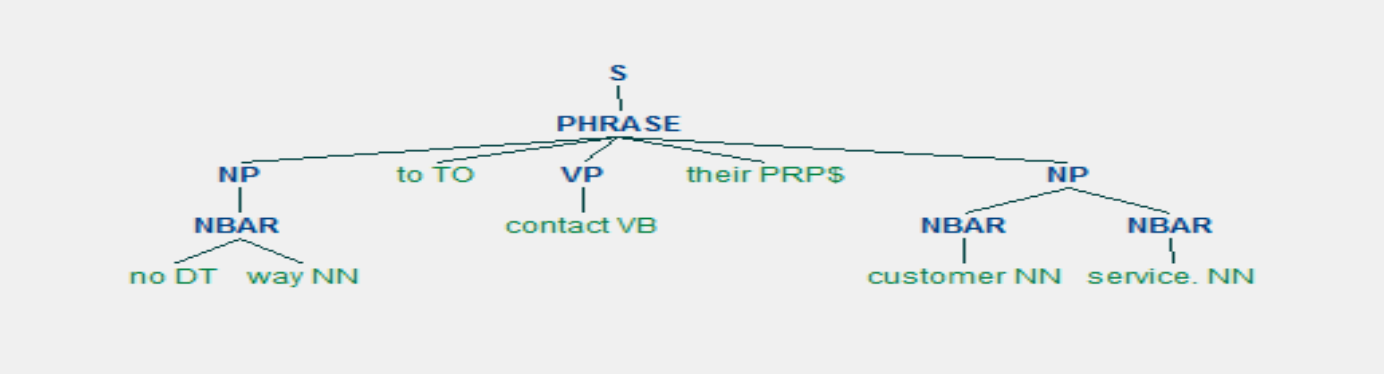

Fig. 7. Phrase chunking example - 2

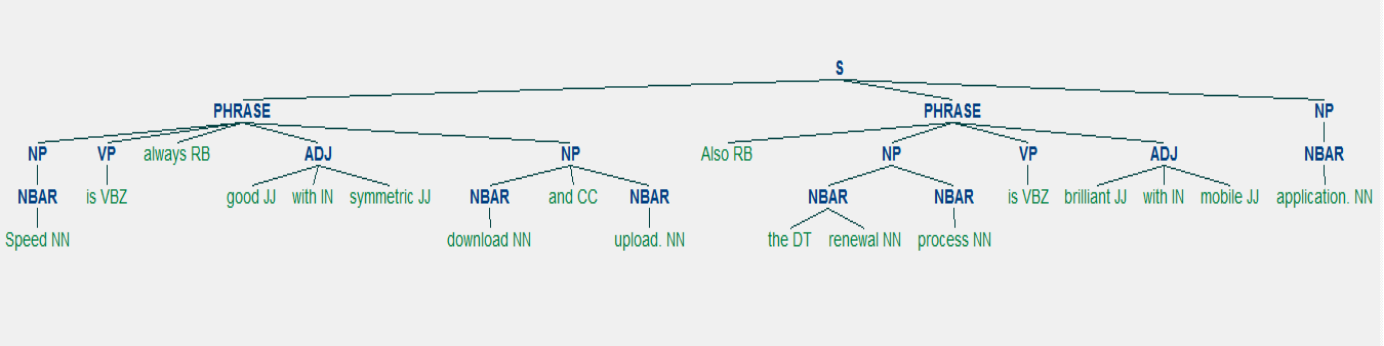

Fig. 8. Phrase chunking example - 3 (with multiple sentences)

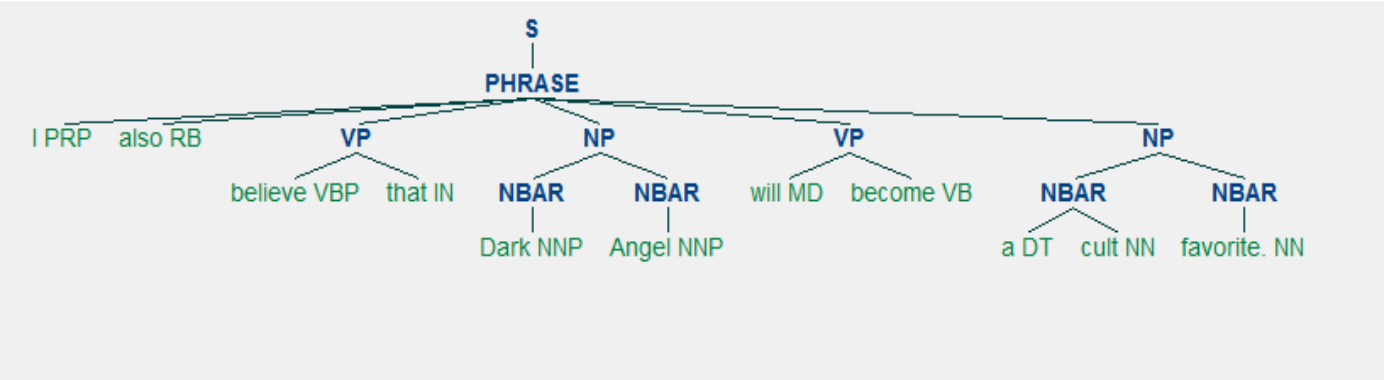

Fig. 9. Phrase chunking example - 4 (with multiple verb & noun phrases)

ptoential drawback of this method. Fig. 9 is another example of phrase chunking where there are multiple verb and noun phrases.

B. Classification Results

TABLE III
EVALUATION OF DATA (SINGLE DOMAIN)

Model	Accuracy	Precision	Recall
Logistic Regression	84.94%	**93.28**%	79.99%
Naive Bayes	83.71%	87.97%	81.03%
LSTM-RNN	**85.46**%	93.25%	**83.56**%

The LSTM - RNN models were trained and tested on a Microsoft Azure Server having an Intel Xeon e5-2667 v3 Haswell 4-core processor and 14 gigabytes of RAM. Training the neural network took approximately 200 seconds per epoch for a total of 25 epochs. Table 2 below shows the results achieved on a testing dataset of 10000 movie reviews. The results of the training & validation accuracy & loss with respect to number of epochs are shown in Fig. 10 & 11 respectively.

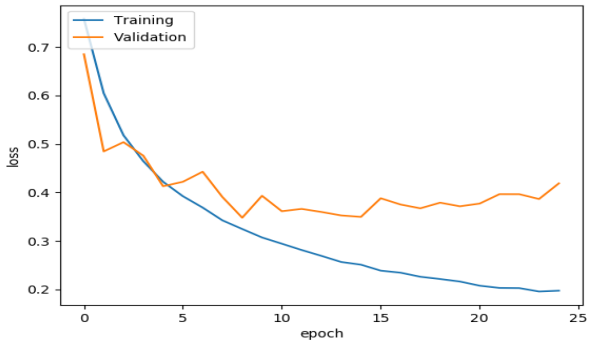

Fig. 10. Training and Validation Loss for LSTM-RNN

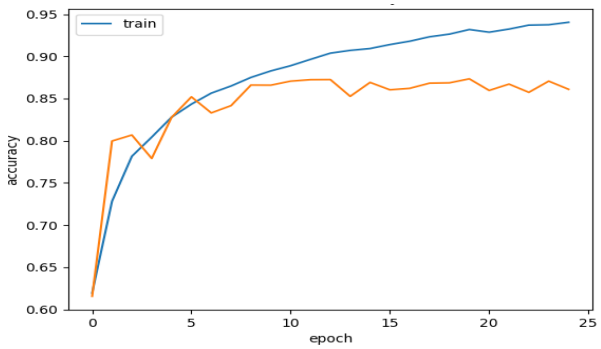

Fig. 11. Training and Validation Accuracy for LSTM-RNN

Table 4 shows an example of a movie review that was misclassified by Logistic Regression & Naive Bayes but was correctly classified by the LSTM-RNN model. We can see that only the LSTM-RNN model is able to predict the actual value of the phrase 'not bad' whereas the other two models are unsuccessful in doing that. This shows that even though the accuracy of the LSTM-RNN is only marginally better than Naive Bayes or Logistic Regression if trained more, it can perform and converge much better.

TABLE IV
CLASSIFIED REVIEW

Review	Classifier	True Value	Predicted Value
The movie was not bad. In fact, it is just opposite of what people portray it to be.	Logistic Regression	Positive	Negative
The movie was not bad. In fact, it is just opposite of what people portray it to be.	Naive Bayes	Positive	Negative
The movie was not bad. In fact, it is just opposite of what people portray it to be.	LSTM-RNN	Positive	Positive

Additionally, the trained model was tested on a dataset of 1000 ISP Reviews whilst being trained on the movie review dataset. Since the context and words used in a different domain are different, it performs poorly giving about 50-55% accuracy since it is not trained on such data.

VI. CHALLENGES

All the implementations had their own drawbacks and came with many challenges like:

1) Spelling mistakes make it very difficult to extract phrases and also, some spelling mistakes results in incorrect part of speech tagging which in turn may lead to an incorrect output.
2) The current model performs poorly when it comes to data from a different domain (Accuracy approx. 50 %). If the model is not trained on a particular type of data, it performs very poorly. Hence, it is not possible to test generalized text on such a model.
3) Incorrect POS tagging as seen in the above aspect extraction results, may lead to incorrect phrase chunking. This in turn leads to incorrect entity extraction which might hide the actual results.
4) Complex grammatical phrases and sentences are still a challenge to the parser. It is very difficult to break down complex sentences with multiple conjunctions/in-junctions into separate phrases.
5) The LSTM-RNN could not perform better due to lack of more balanced training data. More data would have helped it learn more features and prevent overfitting.

VII. FUTURE DIRECTIONS

A. Phrase Chunking Algorithm

The phrase extraction can be further improved by figuring out relations between sentences in a review. For example, 'DVois provides really good internet plans. It also has a great customer support center.' In the previous review, 'it' refers to DVois. Such relations in reviews can be very helpful in extracting a main entity and its attributes Grammar rules

Additionally, an algorithm to figure out the voice of the sentence can be undertaken. This can help in accurately predicting the subject, object and it's predicate.

B. Model

The classification model can be further improved by increasing the training data in order to facilitate better testing. Also, an approach to combine various different domain-specific datasets into one big training dataset can be undertaken and then tested on multiple domains to evaluate multi-domain classification by LSTM-RNN.

VIII. Conclusion

We conclude that the LSTM-RNN model gives a superior performance when compared to the Naive Bayes and Logistic Regression models. Also, it is able to capture some word dependencies as seen in the figures which is essential for sentiment analysis. We plan to train the model on more data and fine tune it in the future. Also, we plan to capture the voice of the sentences (active, passive) in order to accurately extract and subject and the objects of sentences.

References

[1] Hutto, C. J., & Gilbert, E., *VADER: A parsimonious rule-based model for sentiment analysis of social media text*, Eighth International AAAI Conference on Weblogs and Social Media.. Association for Computational Linguistics., 2014.

[2] Bowe A.. *Demonstration of extracting key phrases with NLTK in Python* https://gist.github.com/alexbowe/879414, 2011

[3] Kim, S. N., Baldwin, T., & Kan, M. Y., *Evaluating N-gram based evaluation metrics for automatic keyphrase extraction*, Proceedings of the 23rd international conference on computational linguistics (pp. 572-580). Association for Computational Linguistics., 1999.

[4] Ramos, J. *Using tf-idf to determine word relevance in document queries.* Proceedings of the First Instructional Conference on Machine Learning, 2003.

[5] Songbo, T. & Zhang J., *An empirical study of sentiment analysis for chinese documents*, Expert Systems with Applications 34.4, 2008

[6] Liu, B., *Sentiment analysis and opinion mining.*, Synthesis lectures on Human Language Technologies 5.1, 2012

[7] Hochreiter, S. & Schmidhuber, J., *Long Short-Term Memory*, Neural Computation 9.8, 1997

[8] Esuli, A. & Sebastiani F., *SENTIWORDNET: A high-coverage lexical resource for opinion mining*, Evaluation, 2007.

[9] Maas, A. L., et al., *Learning word vectors for sentiment analysis* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Association for Computational Linguistics, 2011

[10] Olah C. *Understanding Long Short Term Memory (LSTM) Networks: Blog Post* http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015

[11] Melville P., Gryc W., & Lawrence R., *Sentiment analysis of blogs by combining lexical knowledge with text classification*, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.

[12] Minqing H., & Liu B., *Mining and summarizing customer reviews* Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.

[13] *Moutshut ISP Reviews*, http://www.mouthshut.com/internet-service-providers/D-VoiS-reviews-925740132, 2017

[14] Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K. & P. Kuksa., *Natural Language Processing (Almost) from Scratch*, Journal of Machine Learning Research (JMLR), 2011.

[15] *AYLIEN Text Analysis API*, AYLIEN Inc. n.p. n. d., 2014.

[16] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., & Salakhutdinov R., *Dropout: a simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research 15, no. 1, 1929-1958, 2014. .

[17] Yoshua B., Simard P., & Frasconi P., *Learning long-term dependencies with gradient descent is difficult*, IEEE transactions on neural networks 5.2, 157-166. , 1994.