UNIVERSITY OF DHAKA

Sentiment Analysis in Bengali with Word Embeddings

by

Exam Roll No: Curzon-208

Registration No: 2012-41-2017 Session: 2012-13

Exam Roll No: Curzon-204

Registration No: 2012-41-2008 Session: 2012-13

4th year(Honors)

Session: 2012-13

A thesis submitted in partial fulfillment for the
degree of Bachelor of Science

in the

Faculty of Engineering and Technology
Department of Computer Science and Engineering

March 2019

"You look at these past predictions like there's only a market in the world for five computers [as allegedly said by IBM founder Thomas Watson] and you realize it's not a good idea to predict too far into the future."

Geoffrey Everest Hinton

Abstract

Automatic sentiment analysis is the process of categorizing a piece of text by their expressed opinions such as positive, negative or neutral. Lack of usable dataset for sentiment analysis in Bengali language has prevented researchers from analysis in Bengali. In this paper we publish a 10000 sentence text taken from online news portal comments annotated as positive, negative or neutral. We also provide the performance of popular word2vec model on Bengali corpus taken from a news portal and a blog. We then measure the performance of LSTM Neural Networks combined with word2vec model on the sentiment dataset.

Contents

Abstract ii

List of Figures v

Abbreviations vi

1 Introduction 1
- 1.1 Inspiration . . . 1
- 1.2 Current State of Sentiment Analysis . . . 2
  - 1.2.1 Bengali Sentiment Analysis . . . 3
  - 1.2.2 English Sentiment Analysis . . . 3
- 1.3 Research Objective . . . 4
- 1.4 Task Overview . . . 4
- 1.5 Contributions . . . 6
- 1.6 Thesis Outline . . . 6

2 Literature Review 7
- 2.1 Sentiment Analysis in English . . . 7
- 2.2 Sentiment Analysis in Bengali . . . 8
- 2.3 Data annotation . . . 9
- 2.4 Word Embeddings . . . 11
- 2.5 LSTM Network . . . 11

3 Dataset preparation 13
- 3.1 Collection of Text Articles . . . 13
- 3.2 Collection of Comments . . . 14
- 3.3 Preprocessing . . . 15
- 3.4 Annotation of Comments . . . 16
  - 3.4.1 Building The Website . . . 16
  - 3.4.2 Validating the Annotation . . . 23
- 3.5 Crowdsourcing . . . 23

4 Dataset Analysis 25
- 4.1 Word2Vec Analysis . . . 25
  - 4.1.1 Word2Vec basics . . . 25
  - 4.1.2 Word2Vec Experiment . . . 28
- 4.2 LSTM Sentiment Analysis . . . 33

4.2.1 LSTM 33
4.2.2 Sentiment Analysis with LSTM 34

5 Discussions of the Results **37**
5.1 Discussion on Errors 37
5.2 Further Research 38

6 Conclusion **40**

List of Figures

3.1 Positive Sentences 17
3.2 Negative Sentences 18
3.3 Neutral Comments 19
3.4 Ambiguous Comments 20
3.5 Annotation 22
3.6 Profile 22

4.1 Classic Neural Network Language model by Bengio 26
4.2 Architecture of Word2Vec 27
4.3 Word2Vec running on Sachalayatan Corpus 30
4.4 Word2Vec running on Prothomalo Corpus 31
4.5 Word2Vec running on Combined Corpus 32
4.6 Similar words by Word2Vec model 33
4.7 LSTM Unit 34
4.8 Step Size and Epoch vs Accuracy 36

5.1 Examples of wrongly predicted comments 38
5.2 Sentence prediction at word level 38

Abbreviations

GPU	**G**raphics **P**rocessing **U**nit
LSTM	**L**ong **S**hort **T**erm **M**emory
NLP	**N**atural **L**anguage **P**rocessing
NN	**N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
ReLU	**R**ectified **L**inear **U**nit
SVD	**S**ingular **V**alue **D**ecomposition

Chapter 1

Introduction

1.1 Inspiration

Sentiment Analysis is the automatic classification of a given text into positive, negative or neutral category using Natural language Processing,text classification and computational linguistic tools. Sentiment analysis is used to understand the reaction of people in various social platforms. In general it tries to understand or determine attitude of a writer on a certain topic. Sentiment analyzer is a system that can predict human sentiment, Given a sentence the system will be able to determine the sentiment in it.

One of the reasons Sentiment Analysis has gained such popularity is because of its large area of applications which include social media analysis, product review, stock price prediction and numerous other fields. With posts and chats from social media about a company we can predict at what direction the stock of that company will be going, From millions of reviews of the product we can quickly find the product with good sentiments and show them to potential customers. With Sentiment Analysis tools in place it is now possible to process thousands of tweets, comments, blog posts within a fraction of a second. Also as new data comes in, it is possible to process the data and make decisions without manual intervention which saves time and money.

For example we can say that Obama administration, in 2012 used sentiment analysis to understand the public sentiment on campaign messages and announced policy during the

presidential election [1]. We can also understand the emotion of the people on a change of beds in a hotel. We could certify the changes as positive if people have positive review on hotel beds. Also by analyzing the ups and downs of prices of a company on a stock market, we can make decisions about investing on a certain company or not.

A considerable amount of work is done in English, Chinese, Japanese and some other languages[2] [3] [4] [5] [6]. As Bengali is spoken mostly in Bangladesh and West Bengal. Because of the diversity of the people the colloquial form of the language changes from one place to another. So some words are spoken in different tone in different parts of our country. Both lack of valid data-set and highly iflective nature of Bengali language has made it challenging to build a Sentiment Analysis model on this language. An inflection is the change of a word depending on tense, mood, person, number, case, and gender. An example of inflection is write, wrote, written. In Bengali verbs may have as high as 160 inflectional form[7]. This makes working with Bengali language especially difficult.

To alleviate the problem we are publishing a text corpus of articles and sentiment dataset of around 10000 sentences in Bengali language. We also demonstrate the performance of recently developed Neural Networks models for English NLP. First we apply the famous word2vec model from [8] on Bengali news corpus. Then we apply LSTM(Long Short Term Memory) [9] neural network on about 10000 annotated comments below news articles.

1.2 Current State of Sentiment Analysis

Sentiment analysis begun from the early 2000s when different social media started coming to light. We could collect vast amount of data from social media for analysis. These data collected from the social media were the fuel to sentiment analysis. So in this section we will discuss on the recent studies on sentiment analysis.

1.2.1 Bengali Sentiment Analysis

Current Sentiment analysis task for Bengali is based on the sentiments of the lexicons of a sentence. Das et al.[10] describes a way to analyze the sentiments based of subjectivity words. Dictionary based approaches were used in this process. The authors translated the words in SentiWordNet[11] directly and used the words as a basis for the analysis.

Currently the main challenge behind Bengali sentiment analysis and NLP(Natural Language Processing) is that a lack of valid dataset. So we collected a considerable amount of data, and then we annotated and validated a portion of these data.

1.2.2 English Sentiment Analysis

SemEval, which stands for semantic evaluation[12] is yearly series of tasks on sentiment analysis which evaluates computational semantic evaluation systems. Almost every year it introduces different types of subtasks or variations to analyze.

SemEval-2015 [13] added some new subtasks. These subtasks include topic based, trend detection and determination of strength of association. The first two subtasks need topic with message, and the final one determines how close a sentence/phrase towards positive. It gives a value from 0-1. Given a score of 1 determines a positive sentence.

In SemEval-2016 [14] few variants are proposed. First variant is, on a given tweet it estimates sentiment on a 5 point scale. It ranges from Highly Positive to Highly Negative. Another variant showed in this paper is the replacement of classification with quantification. It estimates the distribution of sentiment on a certain topic rather than a certain persons sentiment on a topic.

Sentence level annotation was proposed in [15]. Six basic emotion tags such as: anger, disgust, fear, happy, sad and surprise are used to annotate a sentence.

1.3 Research Objective

Most of our data is collected from political and sports articles. So when there are sudden change or some new decision has been made in the field of politics or sports, we need to find public sentiment on these matter. There can also be some instability in the country. With the help of social media we can check all the comments on a certain article. But going through such a huge number of comments is also tiresome. In this case our system is a big help. It collects and analyzes the data(comments) and can predict the public sentiment on this topic. So when a public decision has been made our system can say whether the public like the decision or not.

To the best of our knowledge only [16] worked with Sentiment Analysis using LSTM networks. The authors used 6698 Bengali entries and 2639 Romanized Bengali entries. The dataset was available in encoded form and we were not able to obtain and verify the original dataset.

In our work we provide a new dataset consisting mostly Bengali comments. We discarded the comments which contained more than 10% of Roman characters. We also provide two corpus. Prothomalo[1] and sachalayatan[2] corpus of articles. We demonstrate the result of the word2vec algorithm and show that it works well in case of Bengali. We then proceed and build a Sentiment analyzer using LSTM.

1.4 Task Overview

Our research work consisted 5 steps

1. Scrape articles and comments from prothomalo and sachalayatan website.
2. Clean the comments from the two websites.
3. Annotate the comments
4. Build word vectors from the whole corpus of articles and comments.

[1] http://www.prothom-alo.com/
[2] http://www.sachalayatan.com/

5. Use the word vectors in an LSTM network and see the initial performance.

First of all we obtained a corpus of 22682 **articles** from the public bitbucket account of Md. Iftekhar Tanveer[3]. Then we build a **scraper** and scraped 39694 articles from **prothomalo** website. Then we prepared a new dataset of about 90000 comments from the section below prothomalo news articles and around 500000 comments from sachalayatan blog. We then annotated random 13000 comments among all the prothomalo comments. Around 3000 of the comments were annotated by students of the department and around 13000 comments were annotated by the two authors. We then took the positive and negative comments only and we were able to collect around 10000 annotated comments.

During the analysis of our dataset we started by converting each word in the corpus of text into a **100** dimensional vector. The idea of representing words by vectors appeared in as early as 1986[17]. It is also known as the Distributed representation of words,Word embeddings etc. The methods aim to quantify the distributional property of words in a large body of text corpora. The advantage of representing words as a vector is that it effectively reduces the amount of space required to represent the vocabulary than representing each word of the vocabulary by one-hot vector. In our research we used the Continuous Bag of Words model from[8] for the Distributed representation of words. The CBOW model basically comprises of 1-layer shallow neural network where we try to map similar words in close proximity. We applied the model on the corpora of consisted of all the articles and comments scraped from prothomalo and sachalayatan website. The whole text consisted of 33 million tokens. We observed similar type of word get mapped in close proximity in that 100-dimensional vector space.

After obtaining the vectors for each word in the vocabulary we proceed to analyze the Sentiment from our 10000 annotated comments. We used two layer LSTM neural network where the first layer consists of 100 LSTM units and the second layer consists of 50 LSTM units. Then we had a fully connected layer of 100 neurons. We converted each comments as a series of vectors and feed it in the LSTM network. We were able to get around 74% accuracy on this model.

[3] https://bitbucket.org/itanveer/sachalayatan_corpus

1.5 Contributions

- We collected a huge amount of data, mainly comments and atricles from Prothomalo[4] and sachalayatan[5].
- We cleaned and annotated the data as POSITIVE, NEGATIVE, NEUTRAL or AMBIGUOUS.
- We constructed a website to annotate and keep track of annotation.
- It was also used as a helpful medium for crowdsourcing.
- We used LSTM which is a variant of RNN to analyze Bengali text.
- Finally we reached an accuracy level of 74.4% .

1.6 Thesis Outline

- In chapter 2 we describe the literature review. We describe what the current state of Sentiment Analysis and Bengali Sentiment Analysis. We also describe the work done with word vectors and LSTM.
- In chapter 3 we describe how we collected comments and articles from Prothomalo[6] and sachalayatan[7]. We then describe what preprocessing we applied and how we annotated the sentences.
- In Chapter 4 we describe our experimental setup and what technologies we used.
- In chapter 5 we discuss the results of our experiment and potential areas of further improvement.
- We discussed our learning's and observations of the proposed system in summary.

[4] http://www.prothom-alo.com/
[5] http://www.sachalayatan.com/
[6] http://www.prothom-alo.com/
[7] http://www.sachalayatan.com/

Chapter 2

Literature Review

A huge amount of work is done in many languages on sentiment analysis. But unfortunately only a few amount of work is done in Bengali language. We only found one paper[16] that was almost the same as ours. The authors also used LSTM. But we failed to get their dataset.

In this chapter we discuss the works that have been done on Sentiment Analysis in English and Bengali. Then we discuss how data annotations are performed. After that we discuss word embeddings and LSTM.

2.1 Sentiment Analysis in English

SemEval-2015 [13] proposed a few new subtasks. Topic based, trend based and trend detection and determination of strengths of association. They collected yearly data from twitter. The data for evaluation was collected few months prior to publication. They also preprocessed the data before training. The data they collected were from the previous and current years tweets for SemEval. They made sure that a tweet was annotated by at-least 5 Turkers. Appearing a word 3 out of 5 times was considered subjective. Most of the teams improved over the previous work, but for the newly introduced subtask trend detection, the result was rather different. Only three teams seemed to improve significantly.

Few variants such as prediction of sentiment, prediction of sentiment over a given topic, estimate sentiment on a five point scale, distribution of tweets to positive and negative, and distribution of tweets on a five points scale were proposed in SemEval-2016 [14]. One is that given a tweet it estimates the sentiment on the 5 point scale rather than 2 point scale. It also has an interesting variant, that it estimates sentiment on a topic but not opinion of someone on some topic.

It introduced ordinal classifications as HIGHLY POSITIVE, POSITIVE, NEUTRAL, NEGATIVE, HIGHLY NEGATIVE. The data they collected were also from previous years SemEval tweet data with new training and test data. They also annotated the data using Amazons mechanical Turk with the addition that they also used CrowdFlower. It provides much better results in some dimensions as excludes inconsistent annotators on the basis of some hidden tests. It was found that the most success were found from the team that used deep learning, convolutional NNs, recurrent NNs, and word embeddings.

2.2 Sentiment Analysis in Bengali

To the best of our knowledge the only work on sentiment analysis using LSTM networks is [16]. Authors used Bengali words and Romanized Bengali to analyze. The authors had a dataset of 9337, which were preprocessed. Out of those there were 6698 Bengali entries and the rest 2639 were English entries. They copy pasted their data from different social media like facebook twitter, youtube and several blog-sites. It was ensured that each comment is annotated at-least twice by two different annotators who were unaware of each other. They used deep recurrent model on the corpus and finally train the dataset. The authors conducted 32 different experiment on the same model but different datasets and loss function. They had a an accuracy of 78%.

Another form of Sentiment Analysis is to use the number of positive and negative Sentiment words in a sentence. SentiWordNet is a publicly available resource for opinion mining. Three sentiment scores: positivity, negativity, objectivity are assigned to each SynSet of WordNet. The database that contains English lexicons are called WordNet.

SynSet is a set of English words where each set is a set of synonyms. It contains information about relations among these sets of synonyms.

Das and Bandyopadhyay[10] proposed multiple techniques to generate SentiWordNet. It is done in three languages which include Bengali also. The authors used a gaming facility involving the Internet population. Automatic, semi-automatic and manual validations and evaluation are used to ensure credibility of the SentiWordNet. Here they analyzed on the basis of subjectivity of the words. In the process of bilingual dictionary based approach an English to Bengali dictionary was produced and positive/negative scores were also copied from their correspondent English words. In case of WordNet based approach Bengali WordNet is developed by AWN which only contains 1775 words. For antonym generation they used some hand crafted rules, which produced about 8% antonym for Bengali words.

2.3 Data annotation

Snow et al.[2] explored the use of Amazons mechanical Turk System. They found high similarity between non-expert and gold standard annotators. The authors used a biased correction technique which helps helps improving annotations of non-expert labels. The interesting thing they did about biased correction is that they re-calibrated non-expert annotations more closer to gold standards. They found that adding non-expert annotations to gold standard has a significant improve to the agreement, which concludes that non-expert annotations can also improve gold levels. Finally they compared the the machine learning classifiers between the two type of annotations.

Sheng et al. [3] also proposed a method to outsource labeling using AMT or Rent-A-Coder. They proposed that repeated labeling can sometimes improve annotation quality, but not always. The authors showed that correct annotation can significantly improve learning rate. In case of imperfect labeling they used selective acquisition of multiple label. The authors explored when to use repeated labeling in a system given the cost setting. Authors concluded that it should be considered when there are noisy labels

and they can be repeated. Also they found that selective-labeling has a huge impact in improving the data substantially.

Hsueh et al.[3] investigated about the quality of data from crowdsourcing. They considered three selection criteria for high standard annotation. There are noise level, sentiment ambiguity, and lexical uncertainty. They considered the newly issue that came to light is that the workers might not be trained, and annotations may be noisy by nature. As they are analyzing the sentiment on a political blog snippets, they find out that the sentiment is highly non-uniform. The authors used the workers to find out the snippets of a specific person and figure whether the snippet is to support or oppose the named candidate. Finally they concluded that noisy annotations can also be useful for modeling.

German Product Reviews[4] does sentence-wise classification and not taking context into account. They considered each sentence to be one of the four categories. They are positive, negative, mixed and neutral. The annotators are asked to notify when multiple products are reviewed in a single sentence and whether background knowledge is required for the sentence to annotate. Authors also considered the fact that if there is an imbalance between positive and negative reviews the machine might not be suitable for training and evaluation. So they balanced the number of positive and negative reviews in a certain product. There were 17853 positive and 21881 negative sentences. The other were considered as mixed or neutral. The rest on which no decision could be made were considered as controversial sentences.

Bosco et al. [18] brought an interesting feature about annotation, irony. They collected twitter data and find the polarity. They collected their data through web scrapping and crawling. Data were annotated multiple annotators in different phases. At first there were five annotators and during the later phase there were two annotators. In order to correct the annotations that were not agreed, a third annotation was done on 25% of the data. Human can naturally identify irony which stands as a ground for studying and developing a corpora which can be used to detect irony. They found that irony can be found in seemingly positive statements which then refers to negative.

2.4 Word Embeddings

Word Embeddings are set of models and algorithms in NLP where words from the vocabulary of a text corpus get mapped into real vector space. [19] summarizes the different ways of creating word vectors and their applications. One way of representing words is to use the index of the vocabulary. But this type of representation fails to capture the semantic relation between words. To capture the semantic relation authors in [20] used a window based co-occurrence matrix where for each window we update the count for each pair of words in the matrix. Finally we get a row and a column for each words in the vocabulary and we concatenate them to form our high dimensional vector. Although this approach is meaningful, the memory constraint is high given that there can be a large number of Vocabularies. In Bengali there are around 10000 base words without considering the inflection. One way to reduce the dimensionality is to use Singular Value Decomposition on the co-occurrence matrix. SVD factorizes a matrix and find the basis on the K most variant directions. In that way we can reduce the dimension as much as we want by adjusting the value of K.

A predictive method for computing word vectors were proposed in [8] and [21] where the authors used 1 layer shallow neural network to learn the word vectors. Here the author used NN to predict context words and optimize the probability of a word occurring in the window of a center word. We used this approach in our research to produce the word vectors that were later fed into the LSTM network for Sentiment Analysis.

The latest state-of-the-art method for word vectors is GloVe [22] which is a count base model. In GloVe the co-occurrence matrix is factorized into lower dimension. The steps in GloVe can be highly parallelize so it is possible to train on much larger dataset.

2.5 LSTM Network

Normal Recurrent Neural Networks have the problem of vanishing and exploding gradients. The problem appears because of the multiplier at each unfolding of the RNN. It is very common for RNN's to have hundreds of steps and with normal RNN the gradient gradually decreases or explodes during the back-propagation [23] algorithm. The back-propagation adjusts the weight by propagating the gradient backwards. To improve

upon this problem [9] proposed a different way of constructing a neural network unit called Long Short Term Memory

LSTM networks are an intuitive way to model time-series or sequence of data. For Sentiment Analysis the sentence can be considered as sequence of words in time. In [24] authors trained LSTM networks to predict the Sentiment of IMDB movie review dataset [25].

Chapter 3

Dataset preparation

In this chapter we describe how we collected and annotated the data. One of our research objective is to publish a dataset from which researchers can build up their work. In this research we publish two types of dataset.

- Corpora of text
- Annotated sentences

The text corpora is useful for different unsupervised model. We can find information through clustering, word embeddings, frequent word distributions and etc. In our work we used the text corpora to build word embeddings which was later used for the Sentiment Classifier.

We also publish 10000 annotated sentences belonging to the class positive,negative and neutral. This dataset opens possibility to supervised learning algorithms.

3.1 Collection of Text Articles

Tools we used for the collection of articles are

- **scrapy**: A web crawler which requests to a given URL and fetches the returned response from the web-server. We can tune how many request per second to make to the server. We can white-list or blacklist websites to crawl. It also has a built-in tracker which avoids sending requests to duplicate URLs.
- **beautifulsoup**: An HTML parser. With this package we can extract links in a given HTML text. We can also parse and find HTML elements with certain properties.
- **json**: Abbreviated as JavaScript Object Notation is a data exchange format. It stores data in object format as key-value pairs.
- **AWS**: Amazon Web Services, where we can remotely control virtual machines.

Sachalayatan text corpus already downloaded and stored as a pickle file in bitbucket by Md. Iftekhar Tanveer. We downloaded and loaded the text in a python dictionary.

For scraping prothomalo articles we used scrapy with beautifulsoup. First we created a spider that starts at the homepage and recursively checks each link that is on that webpage. Given an HTML text beautifulsoup is used to extract the links on the page.

Each article in prothomalo website has an "itemprop" property of "articletitle" and "articlebody". With the help of beautifulsoup we searched for the div element which has "itemprop" property of "articlebody".Moreover each article in prothomalo has an article number, which is the last part of the URL. So we extracted the article number from the URL by parsing it. We then created a json object which has properties "articleNo" and "article" corresponding to the article number and the article body. We then used scrapys huiltin file saving to save each json object per line in a file. The scraper was deployed into an AWS remote machine and We ran it for 12 hours. After 12 hours around 30000 articles were collected and saved.

3.2 Collection of Comments

Extra Tools we used for the collection of comment besides what was used for collection of articles are

1. Chrome Developer Tools: It allows to see the networking requests made from a particular web-page.

The comment section of sachalayatan has the "id" value "comments". So with the help of beautifulsoup we extracted the list of comments under a blog post. We then created a json file with property "URL" and "comments" to indicate the URL of the blog post from where the comments were collected. We ran the scraper for 12 hours in AWS and was able to obtain around 389295 comments.

Collecting comments from prothomalo was more trickier. Prothomalo uses dynamic rendering of comments. Meaning after loading the page, it sends ajax requests to internal API with the article id and the API returns a json Array containing all the comments for that article. At first we try to simulate the browser and used scrapy's builtin selenium browser to render the comments. But it was too time consuming, so we used Chrome Developer Tools to find out the actual API where the requests were sent. Then we used the article ids from the previous saved prothomalo articles and made requests to the API by ourselves to obtain the comments. In this way we were able to obtain around 97159 comments.

3.3 Preprocessing

We collected approximately more than 500000 comments. Some comments were especially large and described stories rather than opinion. As a result we decided to discard comments that were more than 250 characters long.

After splitting the comments into sentence level there are about 81341 comments from prothomalo. Sacahalayatan users have self poems after each comment. There comments is often followed by series symbols like " ","*****" etc and then by their profile motto. We discarded all the text after these symbols which left us with 280000 comments from sachalayatan.

Bengali written in roman character is a popular form of expression in Social Media and blogs of Bangladesh. In our research we focused mainly on Bengali characters. As a

result we discarded the comments which contained more than 10% English characters. We then removed the carriage return and line feeds from the sentences.

After cleaning the comments we found that comments contained multiple sentences and thus have multiple levels of sentiment. So we split the comments into sentence level. To split the sentences we use the "—" character which is Bengali equivalent of a full stop.

So, Finally we were left with 528366 sachalayatan sentences and 100926 prothomalo sentences after splitting. We kept the sachalayatan comments and used only the prothomalo comments for annotation.

3.4 Annotation of Comments

We build our own website to annotate data for this project and use different methodologies to annotate the data. We were able to annotate around 13000 comments out of which all of the comments were annotated by the two authors independently and around 3000 comments were annotated by various batches from the department.

Tools we used for annotation of the comments

- PHP a scripting language for building website
- MySQL as the database.
- Digital Ocean [1] for the hosting of the website.
- namecheap [2] for domain hosting.

Here are some examples of positive, negative, neutral and ambiguous comments.

3.4.1 Building The Website

We build a website where anyone can simply register, then login and start annotating. There is a welcome page that guides the newcomers to annotate data properly. We also

[1] https://www.digitalocean.com/
[2] https://www.namecheap.com/

comment	annotation
এটাই হয়া উচিৎ	positive
সত্য সাহসি লেখার জন্য উনাকে ধন্যবাদ	positive
সত্য সাহসি লেখার জন্য উনাকে ধন্যবাদ	positive
আশা করছি আমরাই ফাইনাল জীতবো	positive
এইটাই হলো শেখ হাসিনার উননয়নের জোয়ার।	positive
এইটাই হলো শেখ হাসিনার উননয়নের জোয়ার।	positive
কেউ দেখেও দেখে না ! কেউ শুনেও শুনে না	positive
ইরফান পাঠান আমার প্রিয় ক্রিকেটার	positive
ইরফান পাঠান আমার প্রিয় ক্রিকেটার	positive
ইরফান পাঠান আমার প্রিয় ক্রিকেটার	positive
প্রিয়তমা স্ত্রীকে ভালবাসার এক সুনদর উদাহরন	positive
মাহমুদুল্লাহ, ফরহাদ রেজা, সাব্বির কে একাদশে দেখতে চ...	positive
আমার একজন প্রিয় খেলোয়ার তার দৃষ্টিনন্দন কি শট আমি ...	positive
ঠিকই বলেছেন জাকির হোসেন	positive
রাষ্ট ও সমাজ ব্যবস্থায় সরকারকে হতে হবে গণতান্ত্রিক...	positive
চমৎকার তো	positive
দাতা দেশ গুলো দারিদ্র-বিমোচন, জঙ্গিবাদ নির্মূল, না...	positive

FIGURE 3.1: Examples of positive sentences

comment	annotation
আর প্রেমে ব্যর্থ হলে যে মিষ্টি পানির স্বাদটা কী রক...	negative
আর প্রেমে ব্যর্থ হলে যে মিষ্টি পানির স্বাদটা কী রক...	negative
আশা করছি আমরাই ফাইনাল জীতবো	negative
একবার চিন্তা করুন,এ ঘটনাটা যদি অন্যদল করত,দাদার কল...	negative
একবার চিন্তা করুন,এ ঘটনাটা যদি অন্যদল করত,দাদার কল...	negative
স্বৈরতন্ত্র যত নিষ্ঠুর হয় তার যাওয়াটা কিন্তু তত কর...	negative
স্বৈরতন্ত্র যত নিষ্ঠুর হয় তার যাওয়াটা কিন্তু তত কর...	negative
কেউ দেখেও দেখে না ! কেউ শুনেও শুনে না	negative
এগুলোর জবাব দেবে কে ?	negative
এগুলোর জবাব দেবে কে ?	negative
বাহাতিরা কোনদিক থেকে স্পেশাল কিনা ? বা তারা কোন বি...	negative
এভাবে চললে ফাইনালে আর খেলতে হবেনা	negative
মাহমুদুল্লাহ, ফরহাদ রেজা, সাব্বির কে একাদশে দেখতে চ...	negative
পিপীলিকার পাখা হয় মরিবার তরে	negative
আপনারা ফাইনালে উঠলে যে দুই মোড়ল বিশেষ করে দাদাবাবু...	negative
মমতা ব্যানার্জি আসলেই কট্টর ধরণের মানুষ	negative
মেসিই সেই পার্থক্য	negative
প্রথম আলোর অনলাইন ভার্সন ও প্রিন্টেড ভার্সনের বিশা...	negative
একটি মেয়ে দেশের সবচেয়ে সুরক্ষিত এলাকায় ঘর থেকে বের...	negative

FIGURE 3.2: Examples of negative sentences

comment	annotation
এখনে তর্ক-বিতর্কের কিছু নেই	neutral
এখনে তর্ক-বিতর্কের কিছু নেই	neutral
এভাবে চললে ফাইনালে আর খেলতে হবেনা	neutral
প্রিয়তমা স্ত্রীকে ভালবাসার এক সুনদর উদাহরন	neutral
অনলাইনে বাংলাদেশের খেলা দেখুন নিচের লিংক থেকে	neutral
অননরা তাকে আনুসরন করে	neutral
রাজনৈতিকভাবে জার্মানি দুভাগ হয়েছিল, ভৌগোলিকভাবে ...	neutral
অ্যাকশান শেষ, এবং আবার শুরু	neutral
সিডব্লিউসি ২০১৫'র এটাই ছিল ভবিতব্য	neutral
মুক্তিযোদ্ধারাই আমাদের সকল প্রেরণার উৎস	neutral
এই মুহূর্তে বিএনপি বা জামাত কি ভুল করল তা বিশ্লেষণ...	neutral
এই মুহূর্তে বিএনপি বা জামাত কি ভুল করল তা বিশ্লেষণ...	neutral
দেখা যাক	neutral
দেখা যাক	neutral
দেখা যাক	neutral
পুলিশের চারিত্রিক ও নীতিগত দৃঢ়তা বৃদ্ধি পেলেই জনগ...	neutral
আজকের খেলার ফলাফল ব্রাজিল -০ ক্রোয়েশিয়া - ১	neutral
আজকের খেলার ফলাফল ব্রাজিল -০ ক্রোয়েশিয়া - ১	neutral
মাসে ৩০,০০০ টা কেবল নিজেরি খরচ	neutral

FIGURE 3.3: Examples of neutral sentences

comment	annotation
এটাই হয়া উচিৎ	ambiguous
বাহাত্তিরা কোনদিক থেকে স্পেশাল কিনা ? বা তারা কোন বি...	ambiguous
আওয়ামীলীগ আছে রাজনীতি ধ্বংসের পথে ! বিএনপি উদ্ধারে...	ambiguous
আর নির্বাচন ভাল হয়েছে	ambiguous
নিরবাচন কমিশনলীগ,পুলিশ লীগ বা র‍্যাব লীগ ---ছাত্রল...	ambiguous
বাস্তব কথা লেখার জন্য অনেক অনেক ধন্যবাদ, পাবলিক প্...	ambiguous
আবহে ইংগিতে মনে হচ্ছে, কমপক্ষে ঢাকা উত্তর তাদের চা...	ambiguous
আবহে ইংগিতে মনে হচ্ছে, কমপক্ষে ঢাকা উত্তর তাদের চা...	ambiguous
আমাদের পরের টারগেট ইন্ডিয়া, বেডি তো বাংলাওয়াস এর ...	ambiguous
আজকের ম্যাচের পর নাকে তেল দিয়ে ঘুমাবে	ambiguous
আঃলীগ বাদে সব নিষ্ক্রি	ambiguous
মাসে ৩০,০০০ টা কেবল নিজেরি খরচ	ambiguous
কেন নারীর জন্য জাতীয় সংসদে ৫০ টি আসন সংরক্ষিত থাকে...	ambiguous
ব্যাপারটা মাথায় রাখতে হবে	ambiguous
ভুল নীতি দিয়ে সহজে জঙ্গি তৈরী করা যায়, কিন্তু নির্...	ambiguous
ভারত কাপ জিতবে	ambiguous
ে্খন বাঘ মামা আবার ঘুম না দিলে হয়	ambiguous
রাজনীতি করবে শুধু	ambiguous
মনে হয় যখন ইনারা স্বপদে ছিলেন তখন আসলে কি করছে	ambiguous

FIGURE 3.4: Examples of ambiguous sentences

have all the information about the users. The comments are divided into chunks. Each chunk contains 100 comments. After annotating a hundred comment the user can move on to the next chunk. Here we also used the constraint that a user cannot get the same chunk more than once. Also we made sure that each chunk will be annotated by at-least two users.

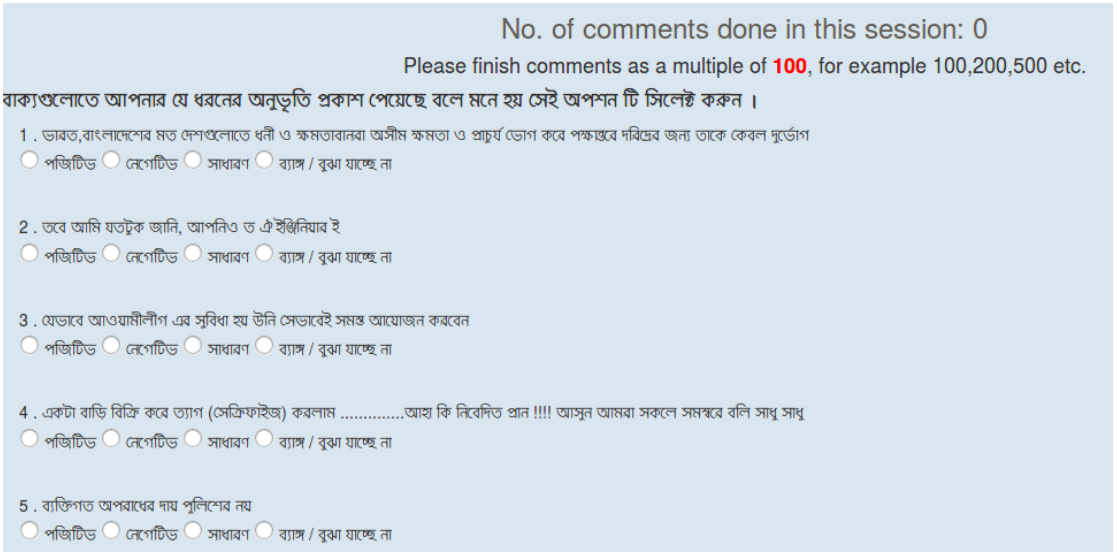

FIGURE 3.5: Partial view of our Website where annotators annotate Sentences

We provided the users about their information that they shared and their current status of annotation. It includes their personal information and how many comments they have annotated.

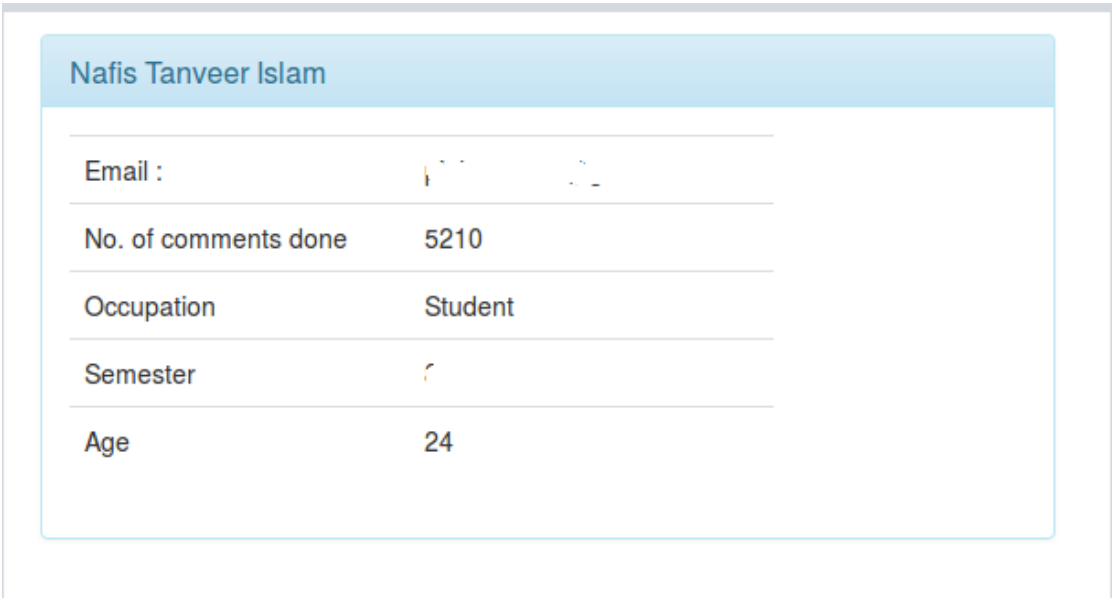

FIGURE 3.6: Profile

3.4.2 Validating the Annotation

As much of the annotation is done by volunteers there might be some noise in the annotation. So there might me a comment that is annotated as positive as one user and negative by another. As there were not enough volunteers we could not annotate a comment 3 times. So we only took the comments that has same annotations by both the users. The volunteers were also instructed not to look at the context of a sentence.

3.5 Crowdsourcing

Crowdsourcing is a specific sourcing model in which individuals or organizations use contributions from Internet users to obtain needed services or ideas[26]. In our project we used crowdsourcing as a way to collect data. As we have a running website we wre able to easily use crowdsourcing as a method of annotating the comment in a convenient time with a low cost. We made the website[3] public, so students studying in different universities can contribute to our work. By crowdsourcing we collected almost 1500 annotations in an hour.

[3] onuvuti.me

Table 3.1: Summary of Collected data

Total number of prothomalo comments collected	90000
Total number of annotated comments	12000
Singly annotated comments	8474
Doubly annotated comments	4000
Comments with three annotations	249
Number of positive comments	4000
Number of negative comments	4700
Number of neutral comments	1500

A brief overview of all the data that we have collected and annotated:

Chapter 4

Dataset Analysis

In this chapter we describe how we analyzed our data. First of all we describe word2vec[8] model which we ran on the whole corpus of articles and comments. After that we describe LSTM and how we applied LSTM in our model. We will assume the following notations for the rest of the chapter. $\boldsymbol{T}$ is the number of words in a text corpus of words $\boldsymbol{w_1, w_2, w_3 ... w_T}$. For a particular text window of size $\boldsymbol{n}$, $\boldsymbol{w_t}$ is the center word. The vocabulary of the corpus is denoted by $\boldsymbol{V}$ and the size of vocabulary is given by $—\boldsymbol{V}—$.

4.1 Word2Vec Analysis

In this section we define the word2vec model and then describe the results found in Bengali Corpus by the word2vec model.

4.1.1 Word2Vec basics

The classic model is a feed forward NN takes a word from the vocabulary, feeds it to the model to embed them into vectors of lower dimensional space, and then fine tune the vectors through backpropagation. In this way the embeddings are effectively the weights of the first layer. The classic neural network model proposed by Bengio et al. [27] at 2003 is shown at figure 4.1.

This objective function for the model is

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log f(w_t, w_{t-1}, \cdots, w_{t-n+1})$$

where $f(w_t, w_{t-1}, \cdots, w_{t-n+1})$ is the probability of the current word given the previous words $p(w_t \mid w_{t-1}, \cdots, w_{t-n+1})$ and is calculated by the final softmax layer. The key problem with this approach is the final output layer. The vocabulary size can be very large from thousands to millions and thus effectively restricting the size of the text corpus on which the model can be trained.

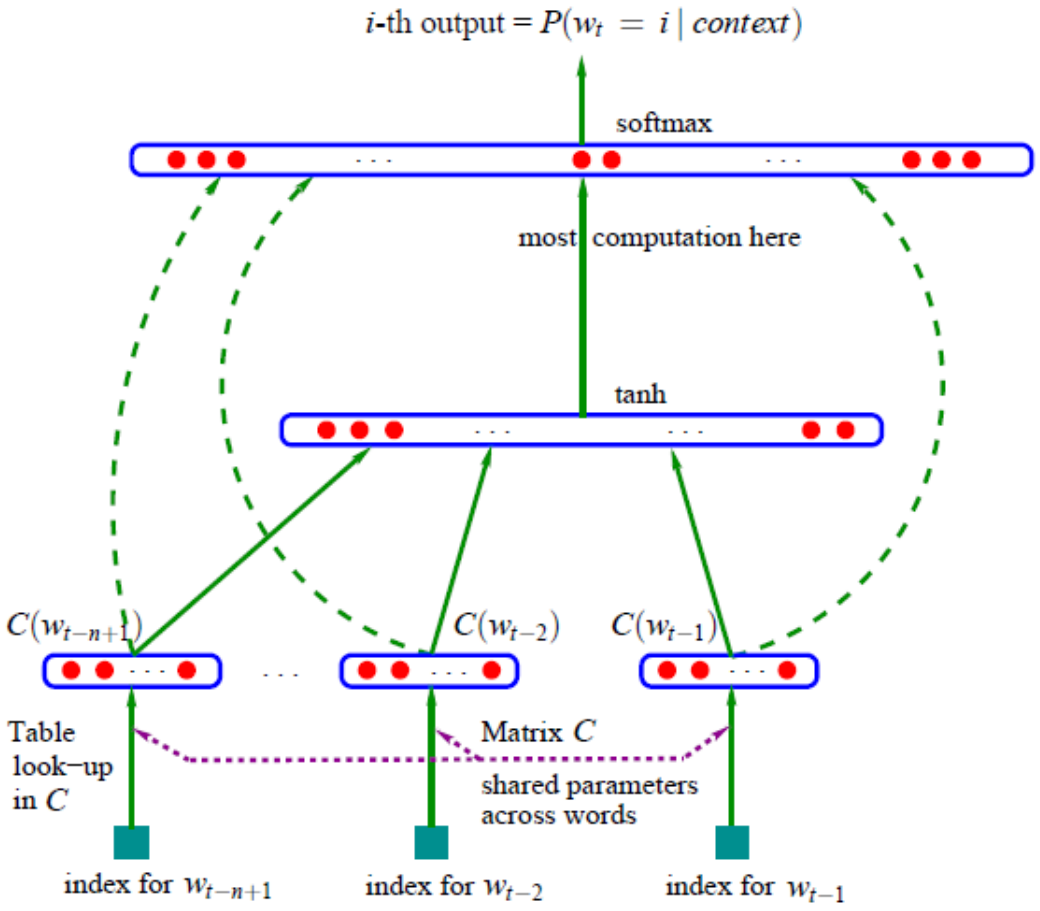

FIGURE 4.1: Classic neural language model (Bengio et al., 2003)

Word2Vec improves upon the limitation of the classic model by restricting the output to the vector size and also by removing the hidden layer. In word2vec we explicitly try to optimize the word vectors without the embedding layer. The main idea of word2vec is to take a window and try to correspond the center word with each of the word given in the window. Every word $\boldsymbol{w}$ has two vectors associated with it. An input vector $\boldsymbol{u_w}$

and an output vector $\boldsymbol{v_w}$.

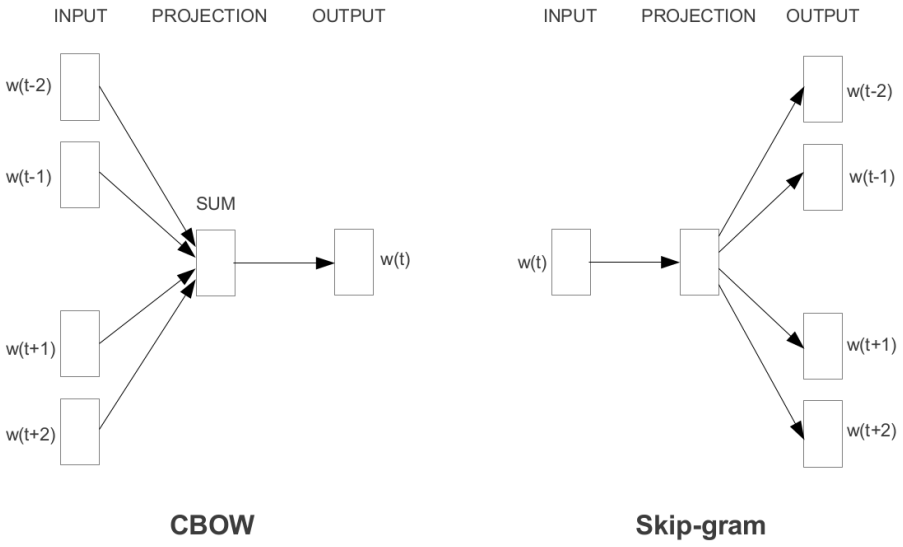

FIGURE 4.2: Architecture of the two models of Word2Vec Taken from [8]

In the CBOW model, we try to predict the center word for each context word within the window. The objective function for the CBOW model is

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t \mid w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n})$$

In the skip-gram model we try to predict the context words given the center word as input. and the objective function for Skip-gram is

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n \le j \le n, \neq 0} \log p(w_{t+j} \mid w_t)$$

And the final layer softmax is given by the equation

$$p(w_O|w_I) = \frac{\exp(v_{w_O}^\top u_{w_I})}{\sum_{w=1}^{V} \exp(v_w^\top u_{w_I})}$$

But we can still see the softmax function which is running over the entire vocabulary. To reduce the computation time of the softmax different techniques are proposed. In

our work we use negative sampling. In negative sampling, instead of taking the entire vocabulary, We take 20-30 random words outside the window to approximate the softmax.

4.1.2 Word2Vec Experiment

In this section we describe how we implemented word2vec in our research and the results. The tools that were used in our analysis are

- **gensim**: A python word2vec implementation
- **jupyter notebook**: A browser based python shell with markdown support

gensim is the library we used to train our word2vec model. We define the three corpuses.

1. **prothomalo** : All the articles of prothomalo corpus
2. **sachal** : All the articles of sachalayatan
3. **combined**: Articles of both the corpus combined plus the comments

We used "Intel Core i5-6300HQ CPU @ 2.30GHz 4" 64-bit CPU. gensim doesn't use GPU acceleration so the we ran the process on 4 CPU cores. We ran the algorithm for different vector lengths and different window size and we note the time taken and the top 2 most similar words for 2 famous Bengali words. In table 4.1 the two values in each column corresponds to the closest word. The method by which closeness is determined is not published in the docs, it basically ranges from 0 to 1 where 1 being exact same and 0 being most dissimilar.

First we ran the experiment on the sachal corpus. The number of sentences in sachal corpus is 1240752 and the number of tokens is 16568392. Figure 4.3 shows the result of running the algorithm with different parameters

As we can see from the table, for small vector lengths the model wasn't quite able to capture the relationship of the words. But as we increase the length of the vector the training time increases. But we also see semantic relationship for the smilar words. The reason for not being able to capture the relation is because the corpus is 16 million words long, it is simply not possible to capture the relationship with that small vector space dimension.

training_time	vectorLen	window	input_word	most_similar	second_most_similar
105.755968	20.0	3.0	সুন্দর	(আশ্চর্য, 0.8478591442108154)	(গুলশান-বাড়িধারায়, 0.8406362533569336)
105.755968	20.0	3.0	খেলা	(শিল্পী, 0.8088500499725342)	(সম-বয়সী, 0.7956321835517883)
102.828966	20.0	4.0	সুন্দর	(আশ্চর্য, 0.8419991731643677)	(Vail, 0.8375908136367798)
102.828966	20.0	4.0	খেলা	(লঞ্চগুলোর, 0.8133596181869507)	(আমেরিকানগুলা, 0.8117883801460266)
113.276567	20.0	5.0	সুন্দর	(ময়ূরকণ্ঠী, 0.8624696135520935)	(আশ্চর্য, 0.8520610332489014)
113.276567	20.0	5.0	খেলা	(দ্বারোন্মোচন, 0.8448063731193542)	(কথা`,, 0.8106421232223511)
119.456805	40.0	3.0	সুন্দর	(গোছানো, 0.7816325426101685)	(চমৎকার, 0.7804107069969177)
119.456805	40.0	3.0	খেলা	(নাচ, 0.7641902565956116)	(সিনেমা, 0.710064172744751)
110.211672	40.0	4.0	সুন্দর	(চমৎকার, 0.7786230444908142)	(সুন্দর,, 0.74245685338974)
110.211672	40.0	4.0	খেলা	(নাচ, 0.7793666124343872)	(ফুটবল, 0.7515888214111328)
113.888654	40.0	5.0	সুন্দর	(চমৎকার, 0.7929639220237732)	("বা:,, 0.7804896235466003)
113.888654	40.0	5.0	খেলা	(নাচ, 0.7717021703720093)	(ফুটবল, 0.7232729196548462)
120.119121	60.0	3.0	সুন্দর	(চমৎকার, 0.758243203163147)	(সুন্দর,, 0.7135251760482788)
120.119121	60.0	3.0	খেলা	(ফুটবল, 0.7312338948249817)	(নাচ, 0.6869816780090332)
118.892202	60.0	4.0	সুন্দর	(চমৎকার, 0.7578220963478088)	(সুন্দর,, 0.7143462896347046)
118.892202	60.0	4.0	খেলা	(ফুটবল, 0.7388536930084229)	(নাচ, 0.7244335412979126)
120.033217	60.0	5.0	সুন্দর	(চমৎকার, 0.7477772235870361)	(সুন্দর,, 0.7155603170394897)
120.033217	60.0	5.0	খেলা	(নাচ, 0.7196791172027588)	(ফুটবল, 0.7121820449829102)
119.552177	80.0	3.0	সুন্দর	(চমৎকার, 0.7427949905395508)	(সুন্দর,, 0.6917991638183594)
119.552177	80.0	3.0	খেলা	(ফুটবল, 0.7269425392150879)	(নাচ, 0.6684321165084839)
117.894567	80.0	4.0	সুন্দর	(চমৎকার, 0.734603762626648)	(সুন্দর,, 0.6913392543792725)
117.894567	80.0	4.0	খেলা	(ফুটবল, 0.721659779548645)	(নাচ, 0.6774715781211853)
200.112181	80.0	5.0	সুন্দর	(চমৎকার, 0.7369921207427979)	(সুন্দর,, 0.7208763957023621)
200.112181	80.0	5.0	খেলা	(ফুটবল, 0.7203596830368042)	(নাচ, 0.6589330434799194)
138.431462	100.0	3.0	সুন্দর	(চমৎকার, 0.753799319267273)	(সুন্দর,, 0.7041245698928833)
138.431462	100.0	3.0	খেলা	(ফুটবল, 0.7267901301383972)	(নাচ, 0.6603020429611206)
156.485874	100.0	4.0	সুন্দর	(চমৎকার, 0.7449390888214111)	(সুন্দর,, 0.7131303548812866)
156.485874	100.0	4.0	খেলা	(ফুটবল, 0.7019464373588562)	(খেলাটা, 0.6467525959014893)
193.591343	100.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
193.591343	100.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)

FIGURE 4.3: Results of running word2vec with different parameters sachal corpus

Then we ran the same experiment with the prothomalo corpus. Prothomalo had 11638995 tokens and 937377 sentences. As Prothomalo has more formal and reporting tone, rather than sachalayatan which is a blog, we can see the difference there.

training_time	vectorLen	window	input_word	most_similar	second_most_similar
95.610849	20.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
95.610849	20.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
91.383247	20.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
91.383247	20.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
103.804048	20.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
103.804048	20.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
83.790569	40.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
83.790569	40.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
91.782071	40.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
91.782071	40.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
83.342591	40.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
83.342591	40.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
102.054499	60.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
102.054499	60.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
102.378425	60.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
102.378425	60.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
100.896544	60.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
100.896544	60.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
94.879210	80.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
94.879210	80.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
68.657617	80.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
68.657617	80.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
69.642972	80.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
69.642972	80.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
73.942573	100.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
73.942573	100.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
73.896549	100.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
73.896549	100.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
71.689142	100.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
71.689142	100.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)

FIGURE 4.4: Results of running word2vec with different parameters on Prothomalo corpus

Then we ran the algorithm with both of the corpus plus the comments from the articles combined. Figure 4.5 describes the results obtained from the combined corpus.

training_time	vectorLen	window	input_word	most_similar	second_most_similar
95.610849	20.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
95.610849	20.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
91.383247	20.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
91.383247	20.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
103.804048	20.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
103.804048	20.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
83.790569	40.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
83.790569	40.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
91.782071	40.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
91.782071	40.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
83.342591	40.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
83.342591	40.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
102.054499	60.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
102.054499	60.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
102.378425	60.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
102.378425	60.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
100.896544	60.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
100.896544	60.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
94.879210	80.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
94.879210	80.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
68.657617	80.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
68.657617	80.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
69.642972	80.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
69.642972	80.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
73.942573	100.0	3.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
73.942573	100.0	3.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
73.896549	100.0	4.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
73.896549	100.0	4.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)
71.689142	100.0	5.0	সুন্দর	(চমৎকার, 0.7501006722450256)	(সুন্দর,, 0.690374493598938)
71.689142	100.0	5.0	খেলা	(ফুটবল, 0.680849552154541)	(খেলাটা, 0.637977659702301)

FIGURE 4.5: Results of running word2vec with different parameters on combined corpus

খেলা	সুন্দর	সরকার	খারাপ
(খেলাটা, 0.7354362607002258)	(চমৎকার, 0.7439269423484802)	(সরকারকে, 0.7994978427886963)	(ভালো, 0.8500868082046509)
(ম্যাচ, 0.7257564663887024)	(সমৃদ্ধ, 0.6845118999481201)	(সরকারব্যবস্থা, 0.777680516242981)	(ভালো, 0.7986155152320862)
(ম্যাচটা, 0.7089033722877502)	(উজ্জ্বল, 0.6801276206970215)	(সরকারের, 0.7455475926399231)	(বাজে, 0.7244887351989746)
(ফাইনাল, 0.6927214860916138)	(সুন্দর,, 0.6670350432395935)	(সরকারব্যবস্থার, 0.709333062171936)	(ভালোই, 0.6833381652832031)
(ম্যাচটি, 0.673851490020752)	(মহৎ, 0.6572996377944946)	(সরকারব্যবস্থাকে, 0.6942031979560852)	(ভালোই, 0.6813988089561462)
(টুর্নামেন্ট, 0.6211658716201782)	(প্রাণবন্ত, 0.655677080154419)	(কমিশন, 0.6580020189285278)	(মন্দ, 0.6719990968704224)
(ম্যাচগুলো, 0.6175659894943237)	(আকর্ষণীয়, 0.6461231708526611)	(সরকারে, 0.6433284282684326)	(আক্রমণাত্মক, 0.6513991355895996)
(খেলাই, 0.615288257598877)	(শুদ্ধ, 0.6398459672927856)	(রাষ্ট্র, 0.6419847011566162)	(খারাপও, 0.6264801621437073)
(সেমিফাইনাল, 0.6146813631057739)	(আনন্দময়, 0.6393035650253296)	(প্রশাসন, 0.6355690956115723)	(আশানুরূপ, 0.619978666305542)
(সিরিজ, 0.6137917041778564)	(পরিচ্ছন্ন, 0.6364089250564575)	(সরকারও, 0.6348514556884766)	(সুবিধাজনক, 0.6097760796546936)

FIGURE 4.6: 10 most similar words as found by the combined model

In figure 4.6 we describe the 10 most similar words for the 4 Bengali words found by the combined model. We for the parameters we choose vector length of 100 and window size of 5. As we use this model later for Sentiment Analysis one advantage of this model is even if the neural network doesn't get trained with some words, it will learn the sentiments because the word vectors for those words will be fairly same. But this poses another problem, we can see from the table that the closest word for the word 'bad' is 'good' and this can have a potential impact in the network training.

4.2 LSTM Sentiment Analysis

4.2.1 LSTM

Introduced in [9], LSTM networks is a type of RNN. Normal RNN can backpropagate through time for many steps. And backpropagation involves multiplication at each layer and thus we meet the vanishing gradient problem where the errors that were present in the output layer gradually vanishes during backpropagation.

To improve upon this problem [9] described a different unit than normal network unit. In the LSTM we have 3 gates namely input gate, output gate and forget gate. So, on top of retaining memory the gates can choose to forget what is in the memory cell. In figure 4.7 we can see an LSTM unit showing the working principle.

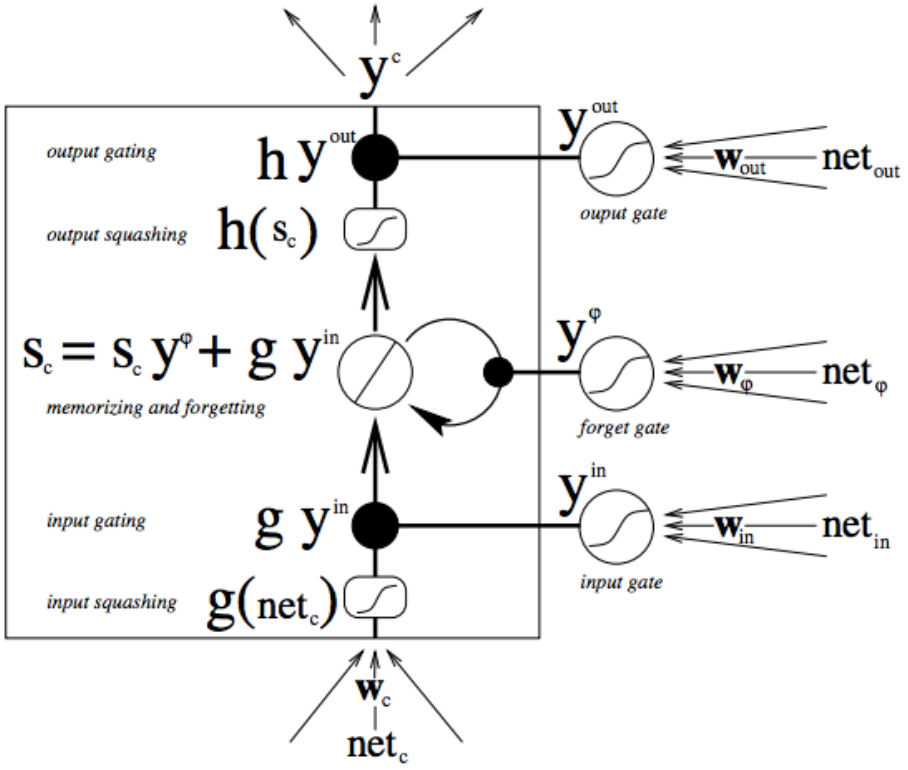

FIGURE 4.7: Figure LSTM unit[28]

4.2.2 Sentiment Analysis with LSTM

After we have trained the word vectors and annotated the comments we finally set up an LSTM network to train for Sentiment Analysis. We took the word2vec model trained with 100-dimensional vector with window size of 4. We only took positive and negative comments and solved the problem of binary classification. Tools we used for this analysis are

1. **keras**[29]: A deep learning library which has a nice interface for creating and training LSTM networks
2. **juypyter notebook**: To visualize the result we used jupyter notebook
3. **Tensorflow**: A neural network library for python which is used by keras as back-end

As tensorflow can use accelerated GPU, we used an Nvidia 960m GPU for the training of the Sentiments. keras doesn't support variable length timestamp as input, so we took

lengths from 5 to 15 and we report the accuracy in those purposes. We randomly took 10% data as the test set.

For the model architecture of training we used the following parameters. After the input layer there were 100 unit LSTM layer and then 50 unit LSTM layer. In the 2nd layer we applied 20% dropout. After that we had 100 neurons whose activation is Rectified Linear Unit. After that there is one binary sigmoid output layer. The loss function used is binary crossentropy and the model is optimized with ADAM optimizer. We set the maximum epoch's to 10 but used the early stopping function of keras where by we stop as soon as validation error starts to increase. We used a batch size of 64. We set the length or timestamp of the input which basically means how many words in a sentence we are going to take. If the number of words is less than the length we pad with zeros. Otherwise we discard the extra words. Figure 4.8 shows the accuracy with respect to different sentence lengths and epochs.

For the experiment we took around 9200 annotated comments and then we took 10% for testing that is around 1000 comments. We ran the experiment 10 times, each time randomly selecting train, test set and then finally calculating the average accuracy for each epoch, step pair in the table.From trial and error it was seen that the best results are obtained between 14 to 18 steps and with 2-3 epochs with our current dataset. So we trained the data with this range and populate the table in Figure 4.7.

As we can see the model performs best where we feed it 17 words and train it for 2 epochs. When we go for 4 epochs the training accuracy increases but the test one decreases indicating an over-fitting. So, after these observation we again train the model with 17 steps and 2 epochs. Figure 4.8 shows the comments that our model failed to predict. In this case we assumed the comment is positive if the model's output is more than .5 otherwise negative.

step	epochs	avg_time	avg_train_acc	avg_test_acc
10.0	2.0	18.536921	80.031224	73.664122
10.0	3.0	23.073603	84.049478	73.217012
10.0	4.0	25.944728	87.652216	72.769902
11.0	2.0	19.261610	80.218566	73.609597
11.0	3.0	22.879106	84.295665	73.685932
11.0	4.0	26.387523	87.792722	72.977099
12.0	2.0	20.058845	80.557223	73.315158
12.0	3.0	24.549927	84.272847	73.827699
12.0	4.0	28.179675	87.747088	72.606325
13.0	2.0	20.862014	80.670109	73.805889
13.0	3.0	24.511297	84.182779	74.285714
13.0	4.0	28.765168	87.863576	73.696838
14.0	2.0	21.330740	80.552420	73.914940
14.0	3.0	26.394975	84.341299	73.620502
14.0	4.0	30.981770	87.783115	73.576881
15.0	2.0	22.196600	80.850246	74.460196
15.0	3.0	26.431096	84.439774	74.405671
15.0	4.0	31.306990	87.463672	73.446020
16.0	2.0	22.723627	81.042392	74.809160
16.0	3.0	28.125804	84.223610	74.558343
16.0	4.0	32.086920	87.410832	73.729553
17.0	2.0	23.685504	81.013570	74.798255
17.0	3.0	28.181941	84.119131	74.231189
17.0	4.0	33.710071	87.314759	74.100327
18.0	2.0	23.970333	81.253753	74.558343
18.0	3.0	29.486108	84.392939	74.013086
18.0	4.0	34.451151	86.315600	74.165758
19.0	2.0	24.657071	80.658100	74.318430
19.0	3.0	29.925851	83.925784	73.860414
19.0	4.0	35.610215	86.660262	73.544166

FIGURE 4.8: Accuracy for different step size and epoch

Chapter 5

Discussions of the Results

In this chapter we discuss our results and the areas of further research. Section 5.1 discusses about the results and graphs and section 2 about the possibilities of further research.

In our paper we used word-embeddings to feed into the neural network. The cardinal importance of using a word vector instead of a plain vocabulary index is that similar words maps to close proximity in vector space. Thus even if we don't have a word in our testing set that wasn't in the training set for Sentiment Analysis our model can predict the sentiment of the word as it is on close proximity to the word that was in the Sentiment dataset.

5.1 Discussion on Errors

Figrure 5.1 shows examples of some comments which aren't correctly predicted by our model. As we can see from the comments that the wrongly predicted comments are mostly of neutral category and the model predicts them between .4 to .7. Other comments that it made mistakes like the 2nd and 3rd and 4th comments are highly subjective in and it might be considered in the opposite way too.

Nevertheless our model was able to capture quite nice sentence structure and variation and predict them. Figure 4.9 shows two sentences where the prediction is plotted after

comment	true_pred	model_pred
অবশ্য কিছু কিছু খেলোয়ার সব ফর্মেটেই থাকতে পারে	negative	0.597853
সবার আগে পিতা মাতার উচিত নিজেদের স্বপ্নের বোঝা সন্তানদের ঘারে চাপিয়ে সন্তানদের নিজেদের স্বপ্নের মৃত্যু ঘটানো বন্ধ করতে হবে	negative	0.632948
ওয়ারীতে ভবন থেকে পড়ে দুই শ্রমিকের মৃত্যু নিজস্ব প্রতিবেদক | আপডেট: ২০১৩-০৮-২৬ ১৬:২৩ এই রকম খবর হর হামেশাই পাই কিন্তু এর কোন প্রতিকার নাই	positive	0.267676
আহা, লুটপাট	positive	0.142235
শুভাগত হোমের জাগায় আমারে নিলে আমি ওর চেয়ে ভালো খেলবো ইনশাআল্লাহ	negative	0.750561
বানিজ্যিক ছবির নায়িকা হওয়া মানেই কি অশালীন পোষাক আর নাচ গান করতে হবে ?	positive	0.154957
প্রধানমন্ত্রী ইচ্ছা প্রকাশ করেছেন তাই যে কোন কৌশলে হোক এটা হতেই হবে; এই হল প্রধান কথা	negative	0.612780
আনিস ভাই আপনার লেখাটা পড়ে কোন মন্তব্য করার ভাষা খুজে পাচ্ছি না	positive	0.401040
যে কোন বাজারের ধর্ম অধিক মুনাফা	positive	0.342539
এমন সংস্কার করতে হবে যাতে মন্ত্রী এমপি হলে টাকা পয়সা বানানোর রাস্তা না থাকে	positive	0.263356
বাংলাদেশের অন্যতম হ্যাকার দল সাইবার ৭১ এর কাজ এটি	positive	0.430986

FIGURE 5.1: Wrongly predicted comments

each word of the model. As we can see the sentiment is dropping or rising exactly to model the transitions of the sentence.

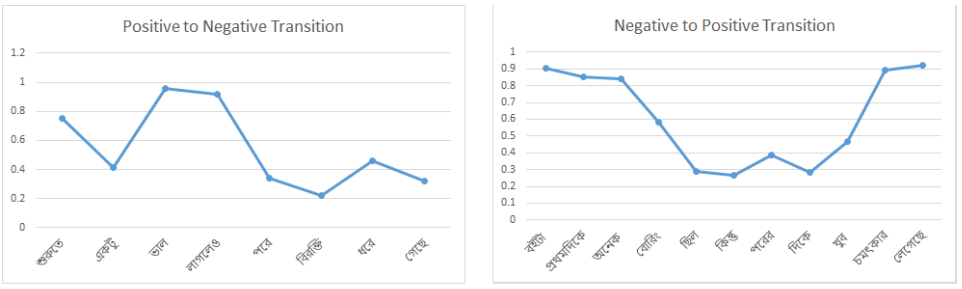

FIGURE 5.2: Different stage of prediction by the model

5.2 Further Research

We worked with the dataset from prothomalo. Most of the comments from prothomalo are on topic of Governments and Sports topic. We couldn't test it on other domains

like product reviews or restaurant reviews dataset. Various groups like Foodbank, Foodbloggers have thousands of restaurant reviews and it can be checked how well the model do in those area. We also didn't include the comments which include roman characters, so Bengali in roman character sentences can be an area of research.

We have around 4 lakh comments out of which we were only able to annotate 11 thousand of them. It is possible to further annotate more comments and check the kind of results for running on more comments. Another possibility is to train the word-vectors differently. In word2vec words of opposite sentiments like 'good' and 'bad' will be in close proximity as they happen to appear in similar contexts. So further research on how to increase proximity of opposite sentimental words and decrease between similar sentimental words would increase the accuracy. Another area would be to investigate include the neutral and ambiguous comments and see what there inclusion

For our analysis we didn't tune the network too much and relied on keras implementation. We zero-padded the shorter sentences. Custom LSTM network which can take variable length input can increase the accuracy on this model. We also didn't do any lemmatization and spelling correction in our analysis, so it would be interesting to see how lemmatization and spelling correction affects the accuracy.

So in summary the the further research that can be done are

- Annotating more comments from prothomalo and sachalayatan
- Creating word vectors differently
- Use exact steps instead of zero padding the network
- Adding lemmatization and spell correction

Chapter 6

Conclusion

Throughout this project we created standard dataset that various researchers can use to analyze their algorithm. We scraped articles and comments from popular news-portal. We kept our annotation procedure simple and standard. We our-self constructed a website where users can annotate sentences. All the annotators including us were soon to be university graduates, which ensures quality of the annotation. After annotating a portion of data, we still have a large number of sentences which can be annotated. We are constantly increasing the amount of our annotated data through crowd-sourcing and self annotation.

After collecting the corpus and comments we applied methods like word2vec and LSTM which are popular in English text to Bengali. We first used to train word vectors using word2vec model to map each word into a 100 dimensional vector space. Words that appear in similar context was shown to have mapped in close proximity in that vector space.

Then we took these word vectors and ran a Sentiment Analysis using LSTM network on our annotated dataset. We have shown that the initial performance on Bengali language is promising and with more fine tuning the model, we will be able to get comparable performance as English.

Bibliography

[1] Understanding Sentiment Analysis: What It Is and Why Its Used. https://www.brandwatch.com/blog/understanding-sentiment-analysis/. Accessed: 2017-02-10.

[2] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics, 2008.

[3] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM, 2008.

[4] Katarina Boland, Andias Wira-Alam, and Reinhard Messerschmidt. Creating an annotated corpus for sentiment analysis of german product reviews. 2013.

[5] A Pappu Rajan and SP Victor. Web sentiment analysis for scoring positive or negative words using tweeter data. *International Journal of Computer Applications*, 96(6), 2014.

[6] C. Bosco, V. Patti, and A. Bolioli. Developing corpora for sentiment analysis: The case of irony and senti-tut. *IEEE Intelligent Systems*, 28(2):55–63, March 2013. ISSN 1541-1672. doi: 10.1109/MIS.2013.28.

[7] Samit Bhattacharya, Monojit Choudhury, Sudeshna Sarkar, and Anupam Basu. Inflectional morphology synthesis for bengali noun, pronoun and verb systems. *Proc. of NCCPB*, 8:34–43, 2005.

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

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

[10] Amitava Das and Sivaji Bandyopadhyay. Sentiwordnet for indian languages. *Asian Federation for Natural Language Processing, China*, pages 56–63, 2010.

[11] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A high-coverage lexical resource for opinion mining. *Evaluation*, pages 1–26, 2007.

[12] SemEval (Semantic Evaluation). https://en.wikipedia.org/wiki/SemEval. Accessed: 2017-02-10.

[13] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 451–463, 2015.

[14] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4: Sentiment analysis in twitter. *Proceedings of SemEval*, pages 1–18, 2016.

[15] Dipankar Das and Sivaji Bandyopadhyay. Labeling emotion in bengali blog corpus–a fine grained tagging at sentence level. In *Proceedings of the 8th Workshop on Asian Language Resources*, page 47, 2010.

[16] Asif Hassan, Nabeel Mohammed, and AKA Azad. Sentiment analysis on bangla and romanized bangla text (brbt) using deep recurrent models. *arXiv preprint arXiv:1610.00369*, 2016.

[17] Geoffrey E Hinton, James L Mcclelland, and David E Rumelhart. Distributed representations, parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations, 1986.

[18] Cristina Bosco, Viviana Patti, and Andrea Bolioli. Developing corpora for sentiment analysis: The case of irony and senti-tut. *IEEE Intelligent Systems*, 28(2):55–63, 2013.

[19] Rémi Philippe Lebret. Word embeddings for natural language processing. 2016.

[20] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28 (2):203–208, 1996.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[22] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[23] DE Rumerhart, GE Hinton, and RJ Williams. Learning representations by back-propagation errors. *Nature*, 323:533–536, 1986.

[24] James Hong and Michael Fang. Sentiment analysis with deeply learned distributed representations of variable length texts. Technical report, Technical report, Stanford University, 2015.

[25] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

[26] Crowdsourcing. https://en.wikipedia.org/wiki/Crowdsourcing. Accessed: 2017-02-10.

[27] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[28] Deeplearning4j Development Team. Deeplearning4j: Open-source distributed deep learning for the jvm, apache software foundation license 2.0. https://github.com/fchollet/keras, 2016.

[29] François Chollet. Keras. https://github.com/fchollet/keras, 2015.