

May 2017

Deep Learning Implementation for Comparison of User Reviews and Ratings

Nishit Shrestha
University of Nevada, Las Vegas

Follow this and additional works at: <https://digitalscholarship.unlv.edu/thesesdissertations>



Part of the [Computer Sciences Commons](#)

Repository Citation

Shrestha, Nishit, "Deep Learning Implementation for Comparison of User Reviews and Ratings" (2017).
UNLV Theses, Dissertations, Professional Papers, and Capstones. 3032.
<http://dx.doi.org/10.34917/10986144>

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Theses, Dissertations, Professional Papers, and Capstones by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

DEEP LEARNING IMPLEMENTATION
FOR
COMPARISON OF USER REVIEWS AND RATINGS

by

Nishit Shrestha

Bachelor's in Computer Engineering (B.E.)
Kathmandu Engineering College
2012

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas

May 2017

© Nishit Shrestha, 2017

All Rights Reserved



Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 20, 2017

This thesis prepared by

Nishit Shrestha

entitled

Deep Learning Implementation for Comparison of User Reviews and Ratings

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Fatma Nasoz, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

Kazem Taghva, Ph.D.
Examination Committee Member

Justin Zhan, Ph.D.
Examination Committee Member

Qing Wu, Ph.D.
Graduate College Faculty Representative

Abstract

Sentiment Analysis is the task of identifying and classifying the sentiment expressed in a piece of text as of positive or negative sentiment and has wide application in E-Commerce. In present time, most e-commerce websites have product review sections, which can be used to identify customer satisfaction/dissatisfaction for their product. In E-COMMERCE websites such as Amazon.com, E-bay.com etc, consumers can submit their reviews along with a specific polarity rating (e.g. 1 to 5 stars at Amazon.com). There is a possibility of mismatch between review submitted and polarity of rating. For Amazon.com, a customer can submit a strongly positive review but give it a low rating. The objective of this thesis is to develop a web-service application which can be used to tackle this situation.

We will perform Sentiment Analysis using Deep Learning on Amazon.com product review data. Product reviews will be converted to vectors using “PARAGRAPH VECTOR” which will later be used to train a Recurrent Neural Network with Gated Recurrent Unit. Our model will incorporate both semantic relationship of review text as well as product information. We have also developed an application in Python, that will predict rating score for the submitted review using the trained model. If there is a mismatch between predicted rating score and submitted rating score, a warning/info will be provided.

Acknowledgements

First, I would like to thank my advisor Dr. Fatma Nasoz for her continuous support and guidance during this research. Her strong knowledge, patience and willingness to motivate and bring the best in her students, helped me to complete my thesis. Thank You!

I would also like to thank Dr. Justin Zhan. His guidance in my previous class for deep learning motivated me to research in the field of “Sentiment Analysis”.

I would like to express my sincere gratitude to my committee member Dr. Kazem Taghva and Dr. Qing Wu.

Special thanks go to my friends and colleagues. Last but not the least, I would like to thank my parents, Narayan Prasad Shrestha and Shushila Shrestha. I owe all my success to them.

NISHIT SHRESTHA

University of Nevada, Las Vegas

May 2017

Table of Contents

Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
List of Algorithms	x
Chapter 1 Introduction	1
1.1 Motivation and Objective	1
Chapter 2 Literature Review	4
2.1 Learning distributed representation of text	4
2.2 Learning Product Information	6
Chapter 3 Methodology	7
3.1 Data and Data Pre-processing	7
3.2 Modeling of Reviews using Paragraph Vector	10
3.3 Learning Product Embeddings using Recurrent Neural Network with Gated Recurrent Unit	13
3.3.1 Recurrent Neural Networks	14
3.3.2 Difficulties in training Recurrent Neural Network	15
3.3.3 Recurrent Neural Network with Gated Feedback Unit	15
3.3.4 Learning Product Review Embedding Sequence Using GRU	16

3.3.5	Sentiment Classification using SVM	18
3.3.6	Web Service To Detect User Rating Mismatch	19
Chapter 4	Experiment and Results	25
4.1	Sentiment Classification using Review Embedding	25
4.2	Sentiment Classification using Review Embedding and Product Embedding	30
Chapter 5	Conclusion and Future Works	38
	Bibliography	39
	Curriculum Vitae	41

List of Tables

3.1	Review sequence for a product	14
4.1	Precision, Recall, and Classification accuracy of our model in 10-fold cross validation with only review embedding.	26
4.2	Precision, Recall, and Classification accuracy of our model in 10-fold cross validation with both review and product embedding.	31

List of Figures

3.1	Architecture of our proposed model.	8
3.2	Data Collection	10
3.3	An unsupervised framework for learning word vectors. Word w_t is predicted from words $(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2})$ in that context.	11
3.4	Distributed Memory Model of Paragraph Vector.	12
3.5	Distributed Bag of Words Model of Paragraph Vector.	13
3.6	Unrolled RNN	17
3.7	GRU with reset and update gate, r and z respectively. h and \tilde{h} are current hidden state and new hidden state.	17
3.8	Example of Review Rating Mismatch. Retrieved from [SL13].	20
3.9	Architecture of our webservice	21
3.10	Sample web interface to submit review and rating.	23
3.11	Warning on review rating mismatch.	24
4.1	Confusion matrix for Paragraph Vector: Trial 1.	27
4.2	Confusion matrix for Paragraph Vector: Trial 2.	27
4.3	Confusion matrix for Paragraph Vector: Trial 3.	27
4.4	Confusion matrix for Paragraph Vector: Trial 4.	28
4.5	Confusion matrix for Paragraph Vector: Trial 5.	28
4.6	Confusion matrix for Paragraph Vector: Trial 6.	28
4.7	Confusion matrix for Paragraph Vector: Trial 7.	29
4.8	Confusion matrix for Paragraph Vector: Trial 8.	29
4.9	Confusion matrix for Paragraph Vector: Trial 9.	29
4.10	Confusion matrix for Paragraph Vector: Trial 10.	30

4.11	Comparison of metrics computed by review embedding only approach and both review embedding and product embedding approach	32
4.12	Confusion matrix for review embedding and product embedding approach: Trial 1. . . .	33
4.13	Confusion matrix for review embedding and product embedding approach: Trial 2. . . .	33
4.14	Confusion matrix for review embedding and product embedding approach: Trial 3. . . .	34
4.15	Confusion matrix for review embedding and product embedding approach: Trial 4. . . .	34
4.16	Confusion matrix for review embedding and product embedding approach: Trial 5. . . .	35
4.17	Confusion matrix for review embedding and product embedding approach: Trial 6. . . .	35
4.18	Confusion matrix for review embedding and product embedding approach: Trial 7. . . .	36
4.19	Confusion matrix for review embedding and product embedding approach: Trial 8. . . .	36
4.20	Confusion matrix for review embedding and product embedding approach: Trial 9. . . .	37
4.21	Confusion matrix for review embedding and product embedding approach: Trial 10. . . .	37

List of Algorithms

1	Computing hidden states and outputs of RNN	14
2	Computing gradients using BPTT	15
3	Training GRU	18

Chapter 1

Introduction

1.1 Motivation and Objective

“Sentiment is an attitude, thought, or judgment prompted by feelings” [FZ15]. Sentiment analysis is the task of computationally identifying and categorizing the sentiment expressed by an author in a piece of text. Sentiment analysis has a wide application in industry and hence has been able to recently attract a lot of research. From forecasting market movements based on sentiment expressed in news and blogs, to identifying customer dissatisfaction from their social media post; sentiment analysis is critical for today’s industry. Sentiment analysis also forms the basis for other applications like recommender systems. In the present time, most e-commerce websites have a separate section where their customer can post reviews for the product or service. Important information like customers’ opinion on their product, reasons for negative reviews, suggestions etc, can be extracted from the posted reviews by performing sentiment analysis on them.

In the past, a review text was converted to fixed length feature vector using bag-of-words or bag-of-n-grams and these feature vectors were later used to train a “shallow classifier” such as Naive Bayes or Support Vector Machine. Although bag-of-words surprisingly performed well and was popular for many years, it has two major flaws. It loses ordering of words and doesn’t consider the semantic relationship between words. For example, words such as “bad”, “worst” and “Las Vegas” are equally distant despite the fact that “bad” should be semantically closer to “worst” than “Las Vegas”. Although bag-of-n-grams somewhat considers word order in short context, it suffers from data sparsity and high dimensionality [LM14].

Recently, deep learning has shown promising results in the field of sentiment analysis. Mikolov et al.[MSC⁺13] introduced “Word Vector”, which is an unsupervised algorithm to efficiently capture semantics of words. Word Vector represents words as vectors and semantically similar words are closer to each other in vector space. In other words, similar words such as “good” and “great” are closer to each other in vector space and far apart from unrelated words like “quick”, etc. Interestingly, this vector representation of text also captures linguistic patterns. For example, the result of vector calculation $\text{vec}(\text{"King"}) - \text{vec}(\text{"Queen"}) + \text{vec}(\text{"Woman"})$ will output vector which is closer to the vector representation of word “Man” than to any other words. Following on his previous work, Mikolov et. al [LM14] introduced “Paragraph Vector” which is also an unsupervised algorithm similar to “Word Vector” that learns fixed length feature vectors for variable length text such as sentences, paragraphs or an entire document. “Paragraph Vector” has shown to outperform bag-of-words models and other form of text representation and has achieved state-of-the-art results in several classification tasks including sentiment analysis. Since they are faster to train and have high efficiency in capturing syntactic and semantic relationship of a text, our model will use “Paragraph Vector” to learn low-dimensional vector of review text.

Along with syntactic and semantic relationship of reviews, our model will also learn product relation as well as temporal relations of reviews. Most work on sentiment analysis ignores product and temporal relations among reviews. We believe that these are important features and can significantly improve sentiment classification accuracy. Our work is based on the argument that, a popular product such as “Apple” receives more number of higher rating and less popular product might receive higher number of lower rating. We will use a recurrent neural network to capture such information as they are great in capturing sequences.

Chen et.al [CXH⁺16] experimented with temporal relations of reviews and reported state-of-the-art results on IMDB and YELP datasets. Their work is based on the argument that a product that receives positive reviews initially is likely to get more positive reviews in the future and vice-versa. They used a recurrent neural network (RNN) to learn the temporal information as well as to capture both product and user information. Motivated by the results of their model, we have trained Recurrent Neural Network with Gated Recurrent Unit (GRU) with our dataset.

In summary, we will first convert 3.5 million product reviews collected by Fang et. al [FZ15] to

fixed length feature vector using “Paragraph Vector”. These feature vectors will be then grouped by product and sorted in a temporal order. Each group is then used to train a RNN (GRU). The vectors generated in the penultimate layer of the RNN will be called product embeddings. These embeddings capture important information like product qualities and temporal relations among reviews. We will further concatenate product embeddings with fixed length vectors generated by “Paragraph Vector” and train a machine learning classifier (SVM).

Furthermore, we will also present a noble approach in tackling the issue of review-rating mismatch. There are situations where a user may write a highly positive review but give it a 1 or 2 star or write a highly negative review but give it a 4 or 5 star. Though such situations are rare, such reviews create confusions and most e-commerce websites would like to solve this. Since reviews with rating 4 and 5 and reviews with rating 1 and 2 are very similar to each other, it will be challenging to classify them into their respective rating class. Hence, our classifier will be trained to classify the reviews into three classes only, i.e ‘negative’, ‘neutral’ and ‘positive’. All reviews with rating 1 and 2 are marked as ‘negative’ reviews, reviews with rating 3 is marked as ‘neutral’ and reviews with rating 4 and 5 are marked as ‘positive’. We will then develop a separate web-service application that will use our classifier to predict a class from the user review. If the predicted class and the class that the submitted rating belongs to are different an error will be submitted back to the user so that they can correct their rating.

Chapter 2

Literature Review

Our Amazon review dataset was collected by Fang et. al [FZ15]. They performed both sentence level as well as review level sentiment analysis on this dataset. For review level sentiment analysis, 4 and 5-star reviews were marked as positive, 1 and 2-star reviews were marked as negative and 3-star reviews were marked as neutral. They identified all negation phrases (for e.g. not worth, not be happy etc), positive and negative words in the dataset. Each negative/positive words and negation phrases were given a sentiment score based on their number of occurrence in positive, negative or neutral reviews. The feature vector of a review was created by two binary strings representing phrases and sentiment words, an averaged sentiment score of the review, a ground truth label and one more element that is computed as $-1 * m + 1 * n$ where m is number of positive sentences and n is number of negative sentences in that review. Using an SVM, they were able to achieve an F1 score of 0.73. On the other hand, using only “Paragraph Vector” we achieved an accuracy of 81% which shows the remarkable strength of paragraph vector.

2.1 Learning distributed representation of text

Deep learning models have shown to outperform bag-of-words approach when it comes to feature generation. Convolutional Neural Network is one of these deep learning models that has been widely successful in capturing distributed representation of text. Kim [Kim14] designed a simple yet powerful convolutional neural network that achieved a very good performance across different datasets for the task of sentiment analysis. The input layer of this network comprised of concatenated word2vec word embeddings, which is then followed by a convolutional layer with multiple filters, then a max pooling layer and finally a softmax layer. The paper also experimented with

static and dynamic word embeddings. For regularization, they employed dropout except on the output layer with a constraint on l_2 norms of the weight vectors. They experimented with different dataset 1) Movie review dataset (MR) 2) Stanford Sentiment Treebank (SST-1) with five labels: very positive, positive, neutral, negative and very negative 3) Stanford Sentiment Treebank (SST-2) with neutral class removed and binary labels. For every dataset, they used rectified linear units, filter width of 3,4,5 with 100 feature maps each and a dropout probability of 0.5. They also used l_2 constraint of 3 and training batch size of 50. To improve performance, they trained their CNN on publicly available word2vec vectors that were trained using 100 billion words from Google News. They were able to achieve state-of-the-art results in MR and SST-2 dataset with an accuracy of 81.5 and 88.1 respectively. Chen et. al [CXH⁺16] used a similar 1-layer convolutional neural network to learn review embeddings of IMDB and YELP datasets. Their convolutional neural network takes reviews of varying length and produces 300-dimensional vectors. To handle varying lengths of review, they padded shorter reviews with zero vectors. Filters of width 3 and 5 moves on the word embeddings to perform one-dimensional convolution and produces multiple feature maps. Only useful features were captured by using max-over-time pooling in pooling layer. The output of multiple filters was then concatenated to create 300-dimensional vector. Softmax function was used as an activation function to train the network over K-classes. [OZLL15] used a seven layer convolutional neural network architecture to perform sentiment analysis on movie reviews collected from rottentomatoes.com. Their architecture consists of three convolutional layers, three max-pooling layers and a fully-connected layer with softmax activation. Johnson et. al [JZ14] used a variation of bag-of-words model to create feature vectors instead of using low dimensional word vectors as input to the convolutional neural network. They also experimented with parallel CNN that had two or more convolutional layers in parallel to learn multiple types of embeddings from one hot input vector. Socher et al. [SPW⁺13] proposed a recursive neural tensor network for semantic compositionality over a sentiment tree bank. Their proposed model achieved state-of-the-art result for binary sentiment classification of Stanford sentiment tree bank. Mikolov et. al [MCCD13] introduced an efficient way of representing words in vector space by predicting next word from given context or by predicting context from given word. Their proposed unsupervised algorithm was able to embed words in a continuous vector space where semantically similar words are mapped to nearby points. Following on their previous work, Mikolov et. al [LM14] proposed “Paragraph Vector” that is able to learn fixed-length feature vector for a variable length text.

2.2 Learning Product Information

In past, most researchers were interested in only in syntactic and semantic relationship between review texts. Currently, there is a growing interest in adding user and product information to strengthen feature vector. Tange et. al [TQLY15] proposed a neural network model that not only captures the semantics of the review text but also user information that expresses that sentiment. A user was represented as a continuous matrix and the product of word matrix and user matrix was used for sentiment classification using a neural network. Tang et. al [TQL15] used a neural network model to embed both user and product information into a vector representation of a document. They first converted a document text to continuous vector representation using a CNN. Each user and product were represented as a continuous matrix and the concatenation of user-text and product-text was fed to a CNN for sentiment analysis. IMDB, YELP 2014 and YELP 2013 were used for sentiment analysis using their model. They were able to achieve state-of-the-art result with an accuracy of 0.435 with IMDB dataset, 0.608 with YELP 2014 dataset and 0.596 with YELP 2013 dataset.

Chapter 3

Methodology

In this section, we will present our approach to learning a) low-dimensional vector representation of reviews using unsupervised “Paragraph Vector” framework and b) product embeddings using a recurrent neural network. Machine learning algorithms cannot directly learn from raw text, it must be represented numerically or as vectors. In the past, bag-of-words used to be a popular approach to convert words to vectors, but they fail at capturing word orders. With bag-of-words, two different sentences will have the exact same vector representation if they contain the same words in any order. Mikolov introduced an unsupervised framework “Paragraph Vector” which is inspired from “Word2Vec” [LM14], an unsupervised algorithm for learning word vectors. Word2Vec is highly efficient in capturing a word’s meaning from its previous occurrences. Similarly, vectors generated by “Paragraph Vector” for two reviews that are similar in sentiment will be close to each other in vector space. Hence we are using “Paragraph Vector” to learn vector representations of reviews.

Section 3.1 describes our dataset and our data-preprocessing steps. Section 3.2 details our approach of learning vector representations of our reviews using “Paragraph Vector”. Afterwards, section 3.3, describes our approach of learning product information along with temporal relations using recurrent neural network with gated recurrent units (GRU). Figure 3.1 illustrates architecture of our proposed model.

3.1 Data and Data Pre-processing

The data used in this research is a set of product reviews collected from amazon.com by Fang et. al[FZ15]. They collected about 3.5 million product reviews. These online reviews includes

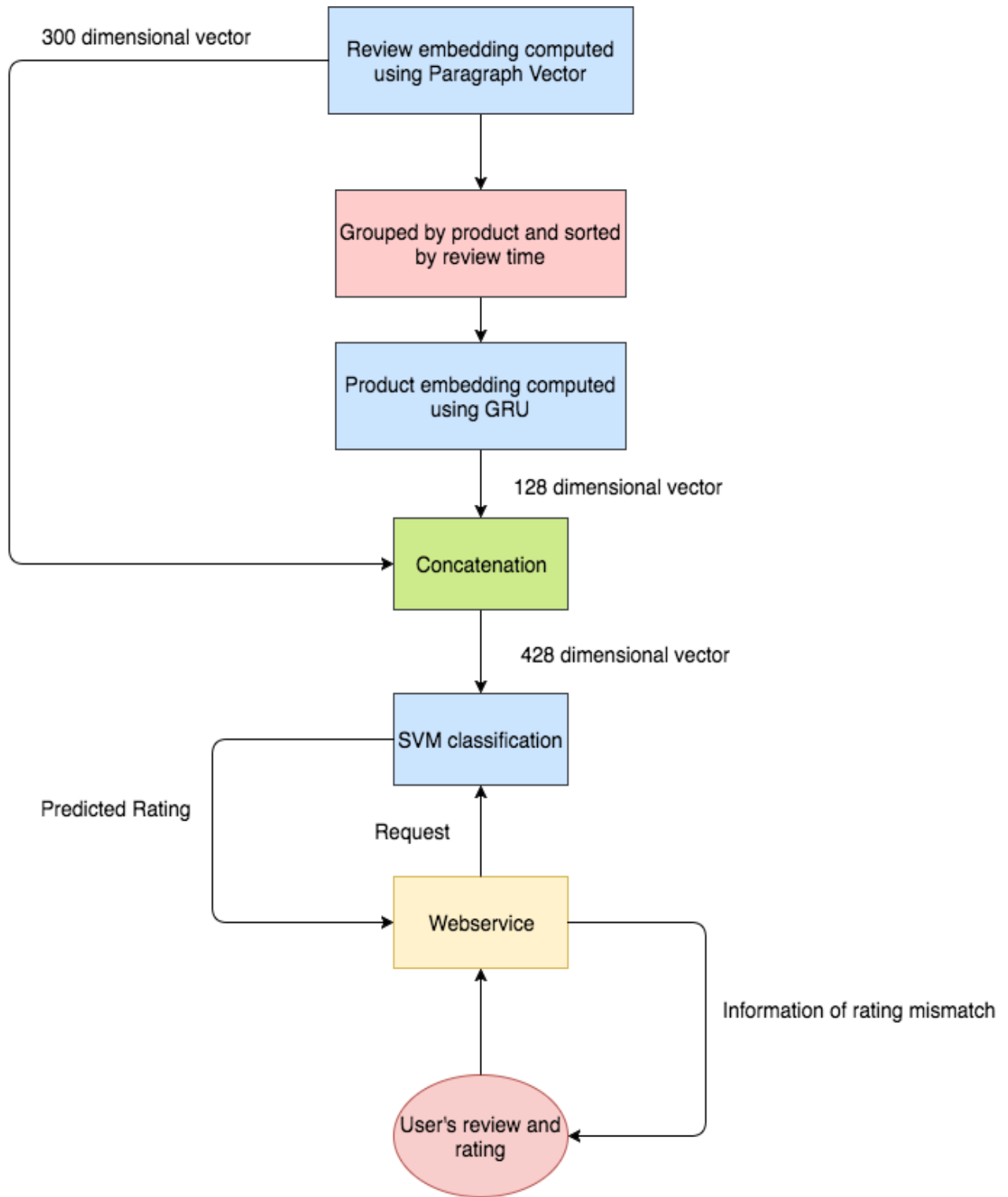


Figure 3.1: Architecture of our proposed model.

information on 1) reviewer ID, 2) product ID, 3) rating, 4) time of the review, 5) helpfulness and 6) review text. The rating is based on 5-star scale. There are 2.2 million 5-star reviews; 622,308 4-star reviews; 265,684 3-star reviews; 171,153 2-star reviews; and 288,789 1-star reviews as shown in figure 3.2 .Below is a sample of a product review.

```
rating: 5.0 out of 5 stars
product_ID: B00DS842HS
helpfulness: 4/4
ID: A28R8UNBXGLFOR
review_by: Melliemel
title: It's working!
review_time: 20140308
review: So far so good. I bought this because I wanted to start oil pulling. It's
      been working great. Great taste (while swishing it around and NOT
      swallowing it). Put some on my arm that was very dry. It helped. Haven't
      cooked with it yet, but I'm sure it will be great!
rating: 2.0 out of 5 stars
product_ID: B000NWGCZ2
helpfulness: 2/3
ID: A1BBW2AZ49E4AR
review_by: T. Haralson "Bella por dentro"
title: Much prefer CeraVe for my son's eczema
review_time: 20121228
review: I really wanted to like this, but it just wasn't as effective in clearing
      up my son's eczema as CeraVe. I did purchase 2 orders to give it a fair try.
      The second order had leaked out during shipping (about 25% of the product). On
      a positive note, I do like the thick consistency and the pump dispenser.
      However, for the price and effectiveness, I prefer other products.
```

Online reviews contain lots of noise like hyperlinks, html tags, informal words, etc. and many words don't have any significant impact on the sentiment of the review. Keeping such words in the review text will increase the dimensionality of the problem. To address this issue we pre-process

the review text before converting to its corresponding vector. Our step-by-step approach is listed below.

- Remove hyperlinks.
- Remove unwanted spaces between words.
- Convert informal words such as ‘I’ll’, ‘I’ve’ to its formal form ‘I will’, ‘I have’ etc
- Add spaces between punctuation. For example ‘This is great!It works.’ will be converted to ‘This is great ! It works .’. Punctuations are treated as separate tokens to try to improve accuracy of the classifier.

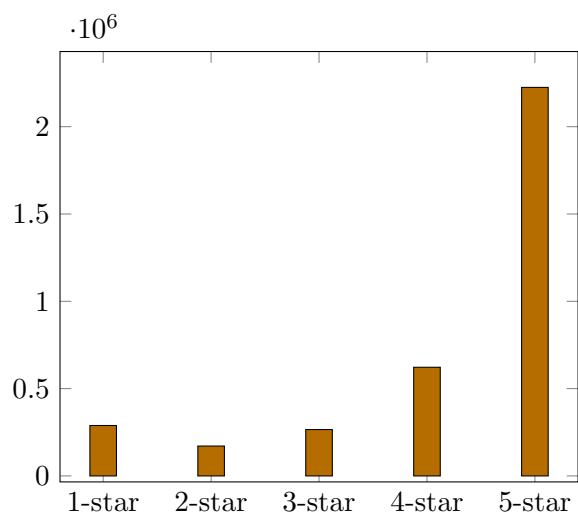


Figure 3.2: Data Collection

3.2 Modeling of Reviews using Paragraph Vector

Paragraph vector [LM14] is an unsupervised learning algorithm that learns vector representation of variable-length text. It is inspired from another unsupervised framework for learning vectors for words (Word Vectors) as illustrated in figure 3.3. We will briefly describe “Word Vector” [MCCD13] first and afterwards describe “Paragraph Vector”.

The task of “Word Vector” is to predict the occurrence of a word given other words in that context. As an example, let us consider the dataset

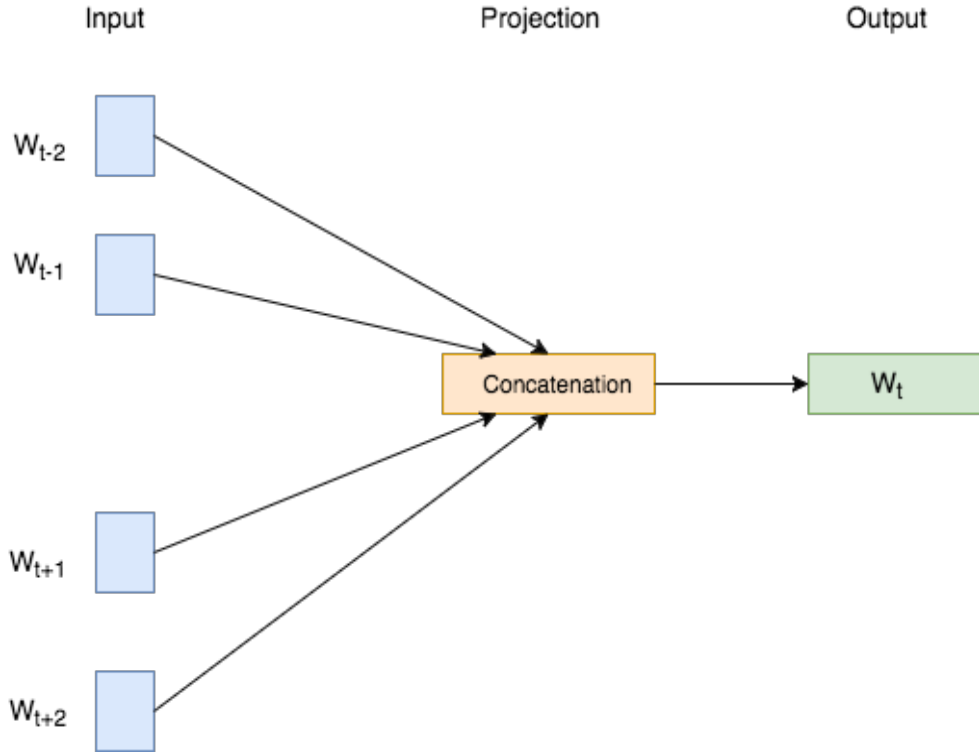


Figure 3.3: An unsupervised framework for learning word vectors. Word w_t is predicted from words $(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2})$ in that context.

the quick brown fox jumped over the lazy dog

using a window size of 1, datasets in the form of (context, target) such as ([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), etc. will be used to train the word vector. For every context it sees, it will predict the target and weights are adjusted via back-propagation. Mathematically, let $w_1, w_2, w_3, \dots, w_T$ be a sequence of training words, the objective of the “Word Vector” is to maximize the average log probability

$$\frac{1}{T} \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \dots, w_{t+k}) \quad (3.1)$$

This prediction task is typically done by a multi-class classifier such as softmax, however for faster training, hierarchical softmax is preferred. The neural network based implementation of word vectors are typically trained using stochastic gradient descent where the gradient value is obtained via back-propagation. After the training, words with similar meanings are mapped closer in vector

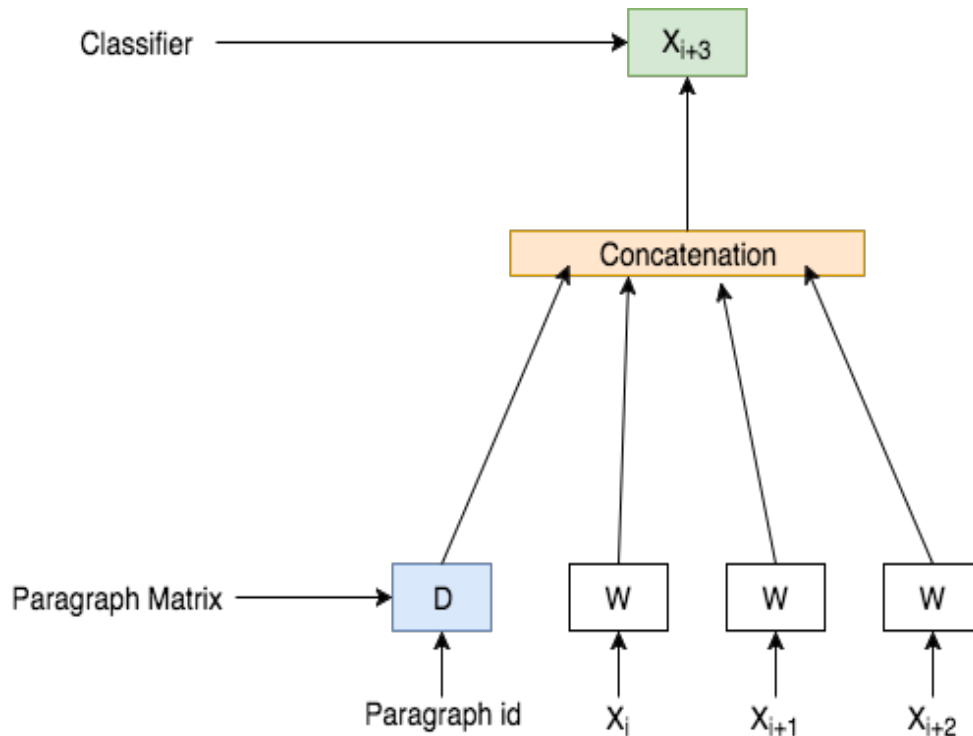


Figure 3.4: Distributed Memory Model of Paragraph Vector.

space. There is a popular neural network based implementation of word vector called “Word2Vec” which was proposed by Google in 2013. This network was trained using Google News Dataset (100 billion words) and the pre-trained vectors are often used as input to a deep neural network.

“Paragraph Vector” is highly inspired from the working of word vectors. Word vector learns semantic relationship by predicting the next word from words in a given context. Similarly, paragraph vectors learn vectors by predicting next word given many contexts that are sampled from a paragraph. There are two flavors of paragraph vector, a) Distributed memory model (PV-DM) b) Distributed bag of words model (PV-DBOW). The difference between the two models is that PV-DM captures word order while PV-DBOW ignores it. Figure 3.4 describes distributed memory model of the paragraph vector. Every paragraph is mapped to a column of matrix D and similarly every words is mapped to a column of matrix W . Given a context sampled from a paragraph or document (for example x_i , x_{i+1} and x_{i+2}), the model predicts the next word x_{i+3} by concatenating the paragraph vector with vectors of words in that context. The model updates both the paragraph matrix and the word matrix while training to minimize error.

Figure 3.5 describes distributed bag of words model of paragraph vector. This is different

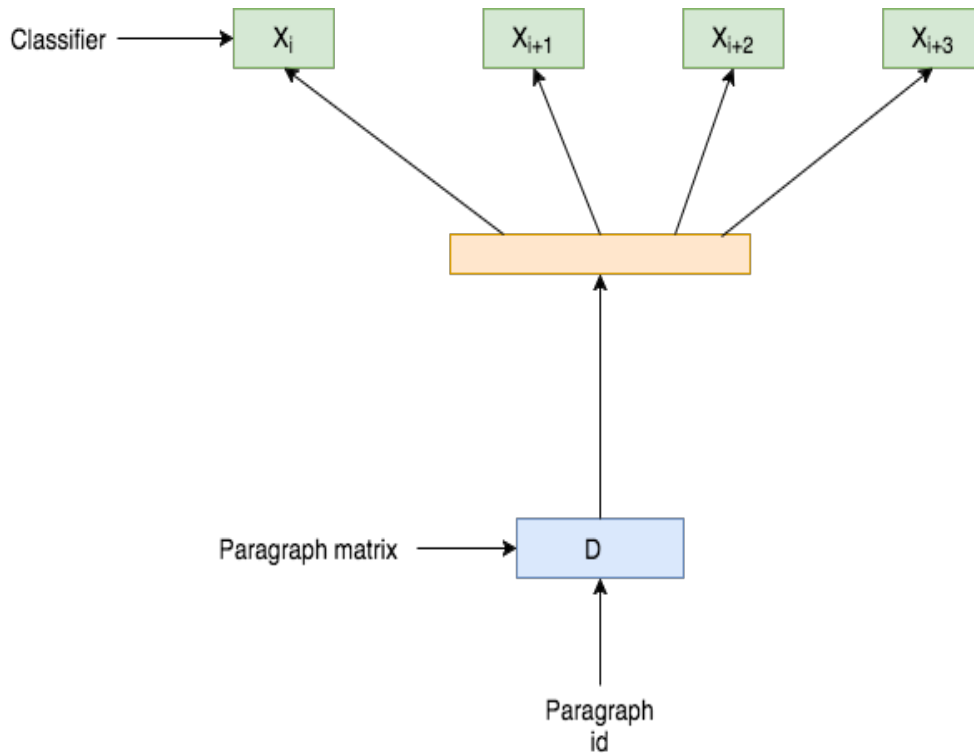


Figure 3.5: Distributed Bag of Words Model of Paragraph Vector.

from PV-DM as it doesn't consider word ordering. PV-DBOW samples a random context from a paragraph and then a random word from that context, then based on that word it tries to predict the context. It doesn't use a word matrix and hence requires less data to store. According to the experiments carried out in [LM14], PV-DM is shown to be more efficient than PV-DBOW, therefore we have applied PV-DM to convert the reviews in our dataset to fixed length vectors.

3.3 Learning Product Embeddings using Recurrent Neural Network with Gated Recurrent Unit

In this section we will describe our approach of learning Product embeddings. By using "Paragraph Vector" we converted 3.5 million product reviews to 300 dimensional fixed-length vectors. To compute product embeddings, we group our reviews by 'Product Id' and then order them by their 'Posted Time'. Table 3.1 describes how a particular review is ordered for computing product embedding for that review. Review text is substituted by their respective "Paragraph Vector". This temporally sorted vectors are the input sequence and their corresponding ratings are the targeted output and used to train a GRU to learn embeddings for that particular product.

Table 3.1: Review sequence for a product

product id	review	posted time
0060245867	This is the first of several of this type and clearly the best of the group.....	2002-08-21
0060245867	I loved the movies, however, I wasn't interested in merely "reading the movies" again. There was no need.....	2006-07-01
0060245867	"If You Give a Mouse a Cookie" really has been the "It" book for some time. Parents love this story,.....	2006-09-05

3.3.1 Recurrent Neural Networks

Most traditional neural networks consider inputs to be independent of each other. But this approach is flawed for tasks such as predicting the next word in a sentence. Here, the next word depends upon the previous word or sequence of previous words. In such cases, RNNs are useful as they are great at capturing sequential information. RNNs have loops in their architecture that allow them to pass information they have collected from previous inputs while processing new input. Figure 3.6 shows a recurrent neural network unrolled through all time steps. In most architecture X_t is input at time step t and Y_t is output at the same time step. U , V and W are weight matrices that needs to be learned and s_t is hidden state that is computed at every time step t . s_t is also known as the memory of the network as it stores features about the input sequences based on its observation of previous inputs. For a given input sequence $x_1, x_2, x_3, \dots, x_T$, RNN computes sequence of hidden states and outputs using the Algorithm 1.

Algorithm 1: Computing hidden states and outputs of RNN

```

for  $t=1$  to  $T$  do
   $u_t \leftarrow Ux_t + Ws_{t-1} + b_h$ 
   $s_t \leftarrow f(u_t)$ 
   $o_t \leftarrow Vs_t + b_o$ 
   $y_t \leftarrow g(o_t)$ 
end for

```

b_h and b_o are bias vectors applied at hidden and output layers respectively. f and g are non-linear functions.

Loss of the RNN is sum of losses across all time steps. Let \hat{y}_t be the predicted output and y_t

be the actual output, then the loss of the network is given as:

$$L(y, y_t) = \sum_{t=1}^T L(\hat{y}, y_t) \quad (3.2)$$

Recurrent neural networks are trained by gradient descent of error using an extension of back-propagation called “Back Propagation Through Time” (BPTT). Algorithm 2 is used for computing gradients using BPTT.

Algorithm 2: Computing gradients using BPTT

```

for t=T to 1 do
   $do_t \leftarrow g'(o_t) \cdot \frac{dL}{dy_t}$ 
   $db_o \leftarrow db_o + do_t$ 
   $dV \leftarrow dV + do_t s_t^T$ 
   $ds_t \leftarrow ds_t + V^T do_t$ 
   $dy_t \leftarrow f'(t_t) \cdot ds_t$ 
   $dU \leftarrow dU + dy_t x_t^T$ 
   $db_h \leftarrow db_h + dy_t$ 
   $dW \leftarrow dW + dy_t \cdot ds_{t-1}^T$ 
   $ds_{t-1} \leftarrow W^T \cdot dy_t$ 
return  $dV, dU, dW, db_o, db_h$ 
end for

```

3.3.2 Difficulties in training Recurrent Neural Network

When recurrent neural networks (RNN) are trained, the weight matrices U, V, and W are updated using Backpropagation Through Time (BPTT). Update to each weight matrix is proportional to the gradient of the error with respect to that matrix and BPTT computes the gradients using the chain rule. When RNN is learning long-term context, depending on the activation functions, the gradient tends to get smaller (vanishing gradient problem) or bigger (exploding gradient problem) towards the earlier layers, which makes it difficult to train the network.

3.3.3 Recurrent Neural Network with Gated Feedback Unit

In section 3.3.2, we learned how recurrent neural networks have problems learning long term dependencies because of “Vanishing Gradients”. Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) were proposed to combat vanishing gradients problem through gating mechanism. As

shown in Figure 3.7, GRU has two gates, a reset gate ‘r’ and an update gate ‘z’. Reset gate ‘r’ decides whether to ignore or combine previous hidden state with newly computed hidden state and is computed as:

$$r = \sigma(U_r x_t + W_r h_{t-1}) \quad (3.3)$$

Update gate ‘z’ decides how much of previous memory needs to be added to compute new memory and is computed as:

$$z = \sigma(U_z x_t + W_z h_{t-1}) \quad (3.4)$$

In this architecture U_z, U_r, W_z, W_r are weight matrices that need to be learned and t is a time step sequence in sequence $x = (x_1, x_2, x_3, \dots, x_T)$. The actual activation of proposed unit h_t is computed as follows:

$$h_t = (1 - z_t) \odot \tilde{h} + z_t \odot h_{t-1} \quad (3.5)$$

$$\tilde{h} = \tanh(U_{\tilde{h}} x_t + W_{\tilde{h}} (h_{t-1} \odot r)) \quad (3.6)$$

3.3.4 Learning Product Review Embedding Sequence Using GRU

As described in earlier section, we first compute 300-dimensional feature vectors for all reviews using “Paragraph Vector”. Each review text is then replaced with its corresponding feature vector in entire dataset. For every unique product, we sort their review vectors based on the review’s posted time. These sorted review vectors along with their corresponding ratings will be the training sequence for that particular product. These sequences are then fed to a GRU to learn 128 dimensional feature vector representation of product information. Training of GRU is performed using Algorithm 3.

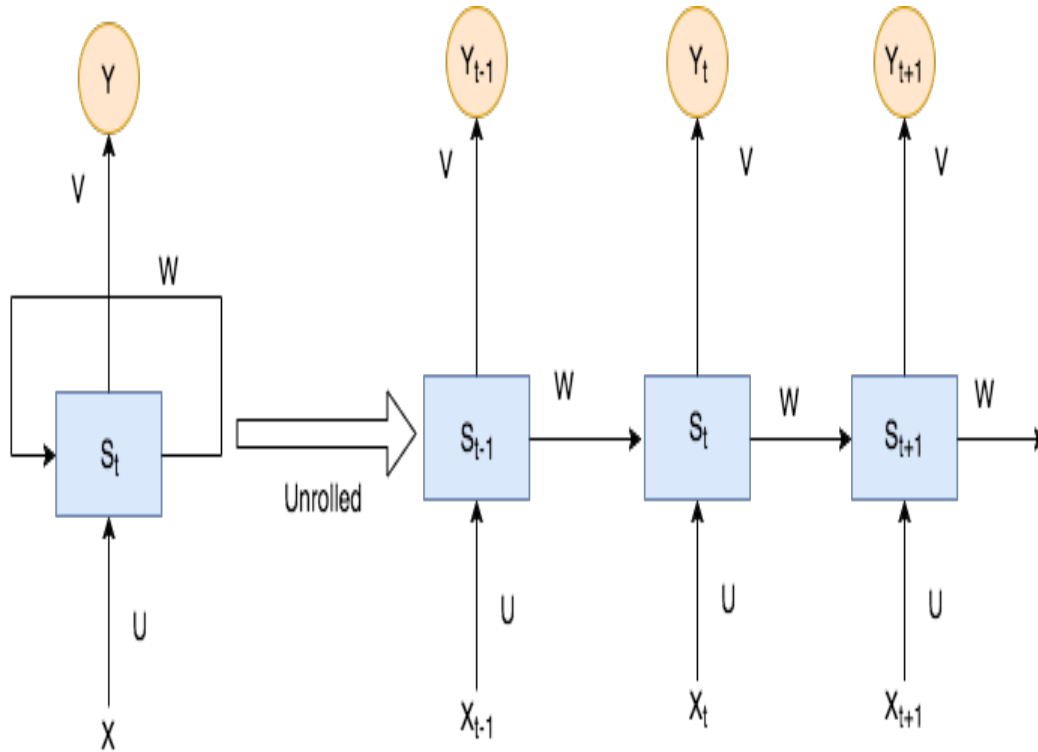


Figure 3.6: Unrolled RNN

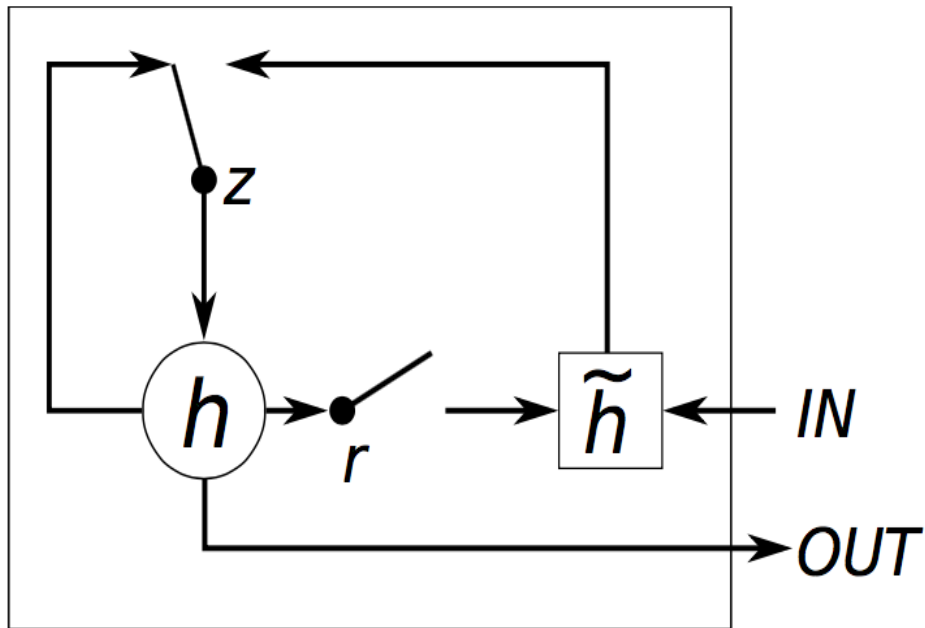


Figure 3.7: GRU with reset and update gate, r and z respectively. h and \tilde{h} are current hidden state and new hidden state.

Algorithm 3: Training GRU

```
for i=1 to Number of Epoch do  
  for Sequence S in training sequences do  
    Train GRU with S  
    if New product or user sequence starts then  
      Reset hidden states  
    end if  
  end for  
  Validate GRU with validation set  
end for
```

As described in algorithm above, we reset the hidden states after the start of a new product sequence since our objective is to capture information that exist between reviews belonging to a unique product sequence. Output layer of our GRU is a softmax layer and output at every time step t is computed as described in equation 3.7

$$y_t = \text{softmax}(Vh_t) \tag{3.7}$$

Output vector y_T at the last time step of every product sequence is consider to be the embedding sequence for that particular product. We train our GRU using 0.25 as dropout rate, Adam as stochastic optimization method, categorical cross entropy as loss function, time distributed dense layer and 128 hidden units. During the training phase, product embedding sequence for each unique product is retrieved and stored in a file for efficient retrieval later on. This product embedding is expected to capture product information.

3.3.5 Sentiment Classification using SVM

Support Vector Machine(SVM) is a traditional machine learning algorithm to classify both linear and non linear data. Given a training data with binary outputs, support vector machine tries to find a hyper-plane as the decision surface such that the separation between positive and negative samples is maximized. The equation of the hyper-plane can be written as $\vec{W} \cdot \vec{U} + b = 0$ where \vec{W} is an adjustable weight vector, \vec{U} is an input vector and b is a bias. Hence, $\vec{W} \cdot \vec{U} + b \geq 0$ for positive samples and $\vec{W} \cdot \vec{U} + b < 0$ for negative samples. To maximize the width of the hyper-plane, we need to minimize $\|\vec{W}\|$ and \vec{W} is computed as $\sum \alpha_i y_i \vec{X}$ where α_i is a numeric parameter and $y_i = 1$ for positive samples and $y_i = -1$ for negative samples.

If the data is non-linear, SVM transforms the data into a higher dimension and then solves the problem by finding a linear hyper-plane. The kernel used for such nonlinear data is called Gaussian Radial Basis function(RBF). Our implementation of SVM uses “Linear” kernel and default value of other parameters provided by scikit tool. We concatenate all review embeddings with their specific product embeddings to create a 428 dimensional feature vector and train our SVM using this final embedding vector.

3.3.6 Web Service To Detect User Rating Mismatch

We have also developed a web service that will prevent inconsistent review and rating. As shown in Figure 3.8, review 3 has an inconsistent review and rating. The sentiment of the review is negative while the rating given is positive. Our web service tries to prevent such inconsistency by using the SVM model trained with 3.5 million reviews and their product embeddings to predict the sentiment class for the given review. If the predicted class and the rating class doesn't match, a feedback will be submitted to user so that they can correct their rating if they wish.



Figure 3.8: Example of Review Rating Mismatch. Retrieved from [SL13].

As shown in Figure 3.9 our webservice takes three inputs a) review text b) given rating c) product id as shown below.

```
{
  "rating": "5",
  "review": "I didn't like this product. Not worth the price.",
  "product_id": "BCX00F5"
}
```

The 300 dimensional embedding for the review text is predicted by previously trained para-

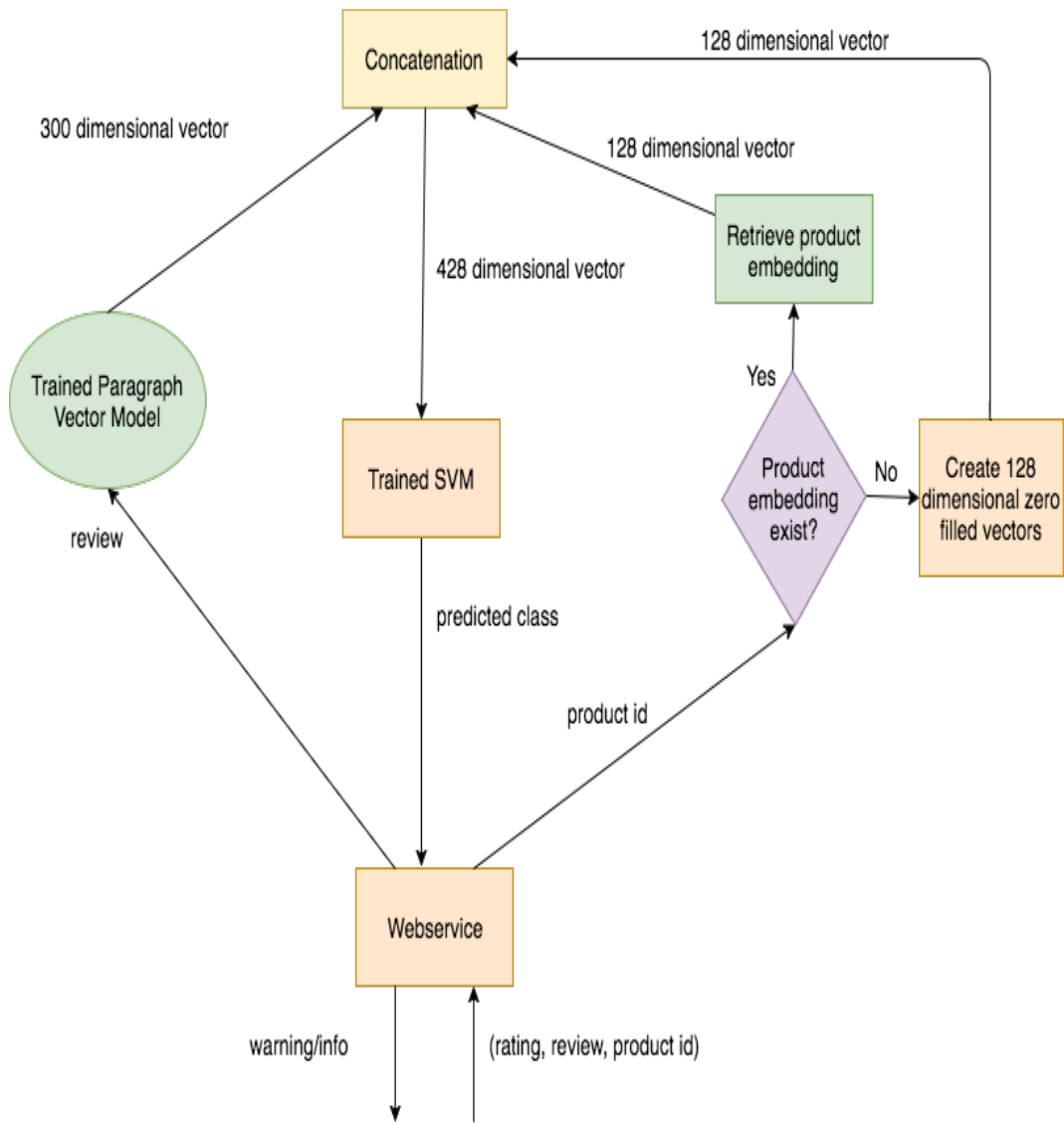


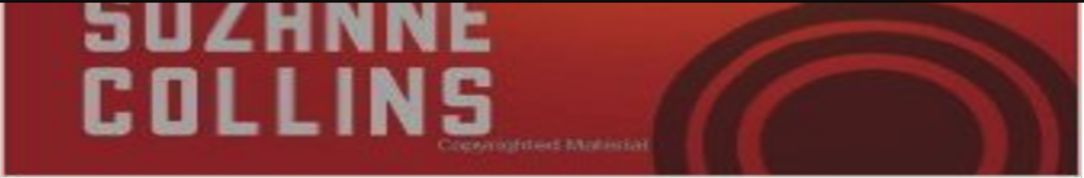
Figure 3.9: Architecture of our webservice

graph vector model. As described earlier, we save product embeddings for all products in a file. Embedding for the given product id is retrieved by the web service from the file and concatenated with review embedding to form the final 428 dimensional vector. This vector is then used by our trained SVM model to predict a sentiment class, if there is mismatch between the predicted class and the sentiment class the given rating falls under, a feedback is given.

For example, user review as shown below

```
"rating": "2",  
"review": "Bought this for my son to read along with the book (first audio  
purchase). To him, reading is a chore and he doesn't take a lot of enjoyment  
from it...but after buying this audio book, and having him read along with  
the cd, it made a HUGE difference! The woman reading on the cd read at the  
perfect pace and changed up her voice a bit when in different character. It  
was just enough to intrigue him to follow along and get lost in the story.  
:)",  
"product_id":"0545586178"
```

is of positive sentiment but the rating provided falls under the negative sentiment class. There is a rating review mismatch therefore our system will provide a feedback message as shown in Figure 3.11. The user can then choose to either change the rating or stick with the same one.



Catching Fire |Hunger Games|2 \$24.99

Suzanne Collins continues the amazing story of Katniss Everdeen in the phenomenal Hunger Games trilogy. Against all odds, Katniss Everdeen has won the annual Hunger Games with fellow district tribute Peeta Mellark. But it was a victory won by defiance of the Capitol and their harsh rules. Katniss and Peeta should be happy. After all, they have just won for themselves and their families a life of safety and plenty. But there are rumors of rebellion among the subjects, and Katniss and Peeta, to their horror, are the faces of that rebellion. The Capitol is angry. The Capitol wants revenge

★★★★☆ 4.0 stars 3 reviews

Bought this for my son to read along with the book (first audio purchase). To him, reading is a chore and he doesn't take a lot of enjoyment from it...but after buying this audio book, and having him read along with the cd, it made a HUGE difference! The woman reading on the cd read at the perfect pace and changed up her voice a bit when in different character. It was just enough to intrigue him to follow along and get lost in the story. :)

★★★★☆

Figure 3.10: Sample web interface to submit review and rating.

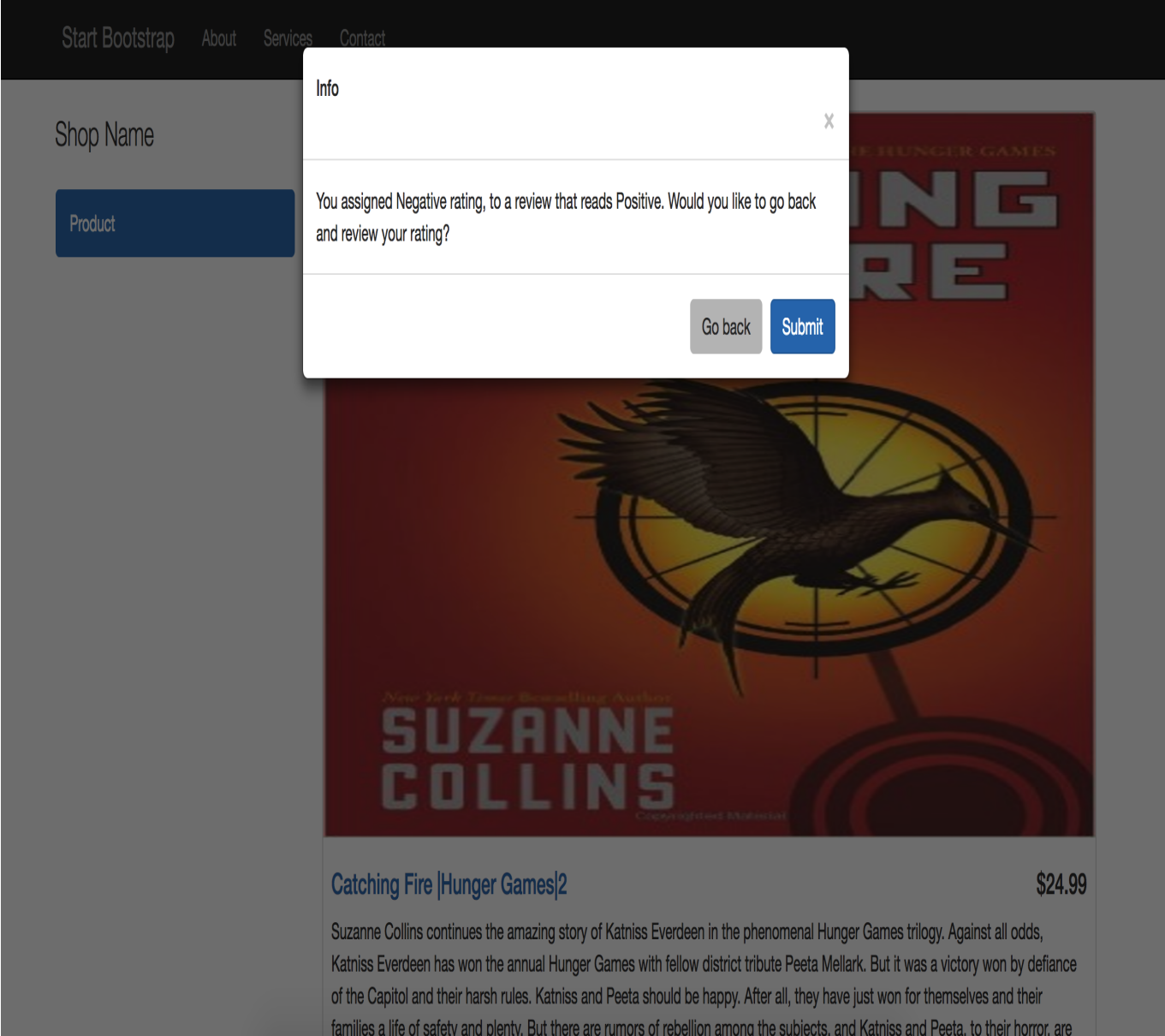


Figure 3.11: Warning on review rating mismatch.

Chapter 4

Experiment and Results

4.1 Sentiment Classification using Review Embedding

We first design a “Paragraph Vector” model using Genism framework. 3.5 million Amazon.com product reviews were converted to 300 dimensional fixed length vector. We train this model for 15 epochs with learning rate 0.025, a window size of 10 and 35 worker threads. The learning rate is decreased by 0.002 after each epoch. This vector with its corresponding rating was used to train a SVM. 10-fold cross-validation method was used to evaluate the performance of this SVM. In 10-fold cross-validation, we divide the dataset into 10 different subsets. In each validation, one of the subsets is used to test the model and remaining 9 subsets are merged to form a training set. To evaluate the performance of our model, we compute precision, recall, and classification accuracy. We use “macro” parameter provided by Scikit for computing precision and recall that calculates the metrics for individual label and computes the unweighted average. Table 4.1 list out the unweighted average of the parameters at each iteration. The final value of the parameters is the average of values computed at each iteration. With only review embedding, our model’s precision score is 0.586, recall score is 0.408, and classification accuracy is 81.29%

Iteration	Precision	Recall	Accuracy
1	0.5890	0.4051	0.8125
2	0.5693	0.4072	0.8115
3	0.5817	0.4092	0.8128
4	0.6004	0.4091	0.8127
5	0.6028	0.4086	0.8135
6	0.5819	0.4088	0.8138
7	0.5731	0.4077	0.8139
8	0.5859	0.4097	0.8120
9	0.5841	0.4081	0.8123
10	0.5926	0.4115	0.8144
Average	0.58608	0.4085	0.81294

Table 4.1: Precision, Recall, and Classification accuracy of our model in 10-fold cross validation with only review embedding.

Confusion matrix describing the prediction performance of this SVM in each iteration of 10-fold cross validation is given in Figure 4.1, 4.2, 4.3, 4.4, 4.5 , 4.6 , 4.7, 4.8, 4.9, 4.10.

	negative	neutral	positive
positive	4895	115	279681
neutral	2466	95	24037
negative	10554	59	35400

Figure 4.1: Confusion matrix for Paragraph Vector: Trial 1.

	negative	neutral	positive
positive	5373	143	278883
neutral	2563	91	23940
negative	10997	67	35245

Figure 4.2: Confusion matrix for Paragraph Vector: Trial 2.

	negative	neutral	positive
positive	5321	134	279133
neutral	2702	89	23854
negative	11213	42	34813

Figure 4.3: Confusion matrix for Paragraph Vector: Trial 3.

	negative	neutral	positive
positive	5421	94	279040
neutral	2620	81	23784
negative	11280	32	34949

Figure 4.4: Confusion matrix for Paragraph Vector: Trial 4.

	negative	neutral	positive
positive	5297	75	279474
neutral	2513	67	23815
negative	11148	28	34884

Figure 4.5: Confusion matrix for Paragraph Vector: Trial 5.

	negative	neutral	positive
positive	5414	129	279666
neutral	2614	93	23835
negative	11043	52	34455

Figure 4.6: Confusion matrix for Paragraph Vector: Trial 6.

	negative	neutral	positive
positive	5235	90	279737
neutral	2680	54	23584
negative	11018	30	34873

Figure 4.7: Confusion matrix for Paragraph Vector: Trial 7.

	negative	neutral	positive
positive	5582	101	278758
neutral	2776	76	24108
negative	11305	36	34559

Figure 4.8: Confusion matrix for Paragraph Vector: Trial 8.

	negative	neutral	positive
positive	5329	88	279008
neutral	2732	73	23714
negative	11158	50	35149

Figure 4.9: Confusion matrix for Paragraph Vector: Trial 9.

	negative	neutral	positive
positive	5579	100	279491
neutral	2805	83	23740
negative	11436	40	34027

Figure 4.10: Confusion matrix for Paragraph Vector: Trial 10.

4.2 Sentiment Classification using Review Embedding and Product Embedding

In this section, we evaluate the performance of concatenated review and product embedding. As described in section 4.1, we first generate 300-dimensional embeddings for entire 3.5 million Amazon.com product reviews. We group these reviews by their product and sort them by their posted time. These grouped and sorted reviews along with their corresponding ratings are the sequence for a product and are fed to a GRU. The embedding generated at the last time step of the sequence is treated as the embedding for that product. This product embedding is concatenated with the review embedding and is used to train an SVM. 10-fold cross-validation was used to evaluate this model. We compute precision, recall and classification accuracy as computed with paragraph vector only approach. Table 4.2 lists out the unweighted average of the parameters at each iteration.

Iteration	Precision	Recall	Accuracy
1	0.5987	0.4183	0.8168
2	0.5940	0.4314	0.8189
3	0.5915	0.4265	0.8185
4	0.5962	0.4252	0.8186
5	0.5844	0.4310	0.8179
6	0.5853	0.4231	0.8186
7	0.5951	0.4249	0.8180
8	0.6016	0.4301	0.8182
9	0.5925	0.4206	0.8179
10	0.6064	0.4211	0.8182
Average	0.5945	0.4252	0.8182

Table 4.2: Precision, Recall, and Classification accuracy of our model in 10-fold cross validation with both review and product embedding.

Classification using both review embedding and product embedding gave us an increase in classification accuracy from 81.29 % to 81.82 %, increase in precision from 0.5860 to 0.5945 and increase in recall from 0.408 to 0.4252. The performance of both the approach is compared in Figure 4.11.

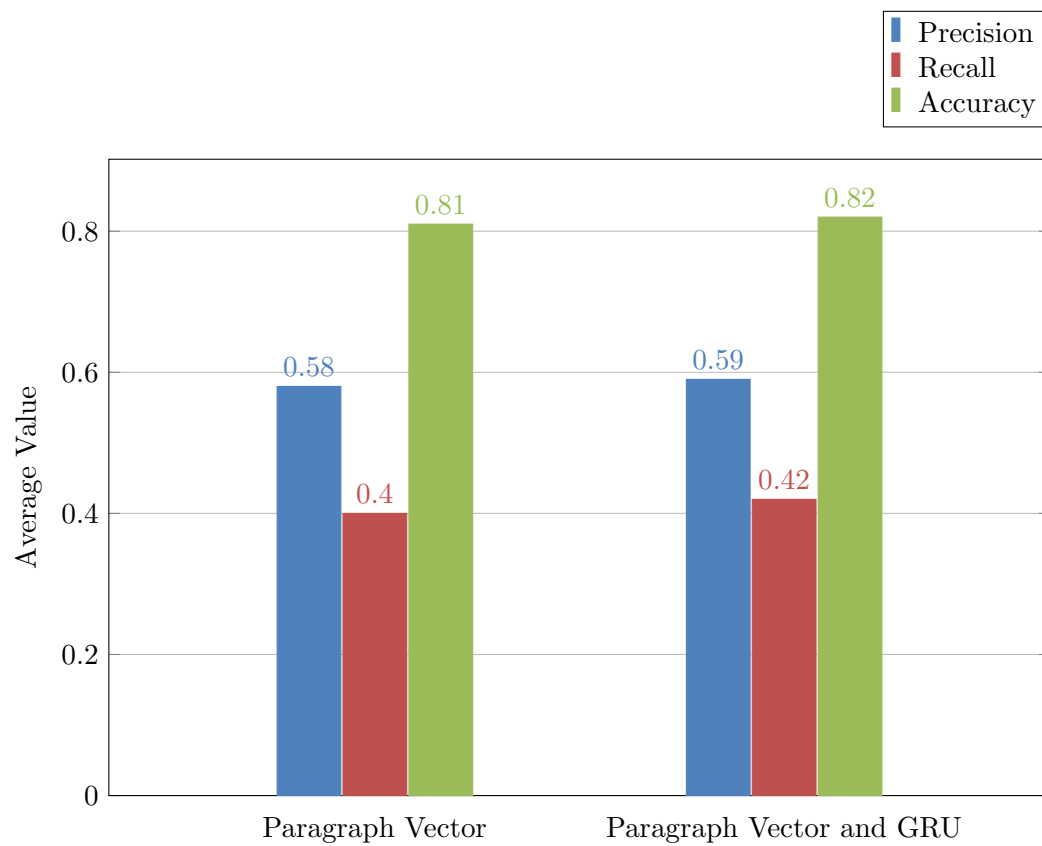


Figure 4.11: Comparison of metrics computed by review embedding only approach and both review embedding and product embedding approach

Confusion matrix for each iteration of 10-fold validation for this model is given in Figure 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21.

	negative	neutral	positive
positive	5340	90	279302
neutral	2728	74	23818
negative	12465	42	33443

Figure 4.12: Confusion matrix for review embedding and product embedding approach: Trial 1.

	negative	neutral	positive
positive	6437	126	278078
neutral	3214	99	23264
negative	14442	60	31582

Figure 4.13: Confusion matrix for review embedding and product embedding approach: Trial 2.

	negative	neutral	positive
positive	5841	121	278749
neutral	3102	88	23481
negative	13642	52	32225

Figure 4.14: Confusion matrix for review embedding and product embedding approach: Trial 3.

	negative	neutral	positive
positive	5723	193	278658
neutral	2979	148	23346
negative	13455	83	32716

Figure 4.15: Confusion matrix for review embedding and product embedding approach: Trial 4.

	negative	neutral	positive
positive	6469	106	277951
neutral	3330	76	23192
negative	14464	54	31659

Figure 4.16: Confusion matrix for review embedding and product embedding approach: Trial 5.

	negative	neutral	positive
positive	5540	146	279072
neutral	2900	94	23666
negative	13116	58	32709

Figure 4.17: Confusion matrix for review embedding and product embedding approach: Trial 6.

	negative	neutral	positive
positive	5685	210	278859
neutral	2946	170	23311
negative	13327	112	32681

Figure 4.18: Confusion matrix for review embedding and product embedding approach: Trial 7.

	negative	neutral	positive
positive	6417	216	277914
neutral	3209	175	23214
negative	14165	76	31915

Figure 4.19: Confusion matrix for review embedding and product embedding approach: Trial 8.

	negative	neutral	positive
positive	5321	68	279574
neutral	2799	49	23857
negative	12722	29	32882

Figure 4.20: Confusion matrix for review embedding and product embedding approach: Trial 9.

	negative	neutral	positive
positive	5233	159	279788
neutral	2838	127	23390
negative	12691	52	33023

Figure 4.21: Confusion matrix for review embedding and product embedding approach: Trial 10.

Chapter 5

Conclusion and Future Works

From this research, we showed how a powerful deep learning model can be built using “Paragraph Vector” and “GRU”, which can later be employed to prevent review and rating inconsistencies. We first employ “Paragraph Vector” to learn the syntactic and semantic relationship of a review text. We further group and sort review embeddings to form a product sequence which is fed to a GRU to learn product embeddings. The concatenation of review embedding generated from “Paragraph Vector” and product embedding generated from “GRU” is used to train an SVM for sentiment classification. Our classifier gives superior prediction accuracy of 81.82%. We also propose using this approach to prevent review and rating inconsistencies. We developed a webservice that takes user review text and given rating and warns if there are any inconsistencies with the given rating and review.

From our research, we showed how product information can be a powerful feature to be employed in the task of sentiment analysis. We believe that similar technique can be employed to learn user information. Tang et al. reported that “sentiment ratings from the same user are more consistent than those from a different user”. Some users are more lenient and can give much higher ratings for the same review text compared to critical users. We are motivated to exploit this information in future.

We would also like to experiment with other deep learning methods of learning distributed representation of text such as “CNN” and “RNN”. We would also like to experiment with “LSTM”, “Bidirectional LSTM” and other sequence learning model.

Bibliography

- [BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [CVMG⁺14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [CXH⁺16] Tao Chen, Ruifeng Xu, Yulan He, Yunqing Xia, and Xuan Wang. Learning user and product distributed representations using a sequence model for sentiment analysis. *IEEE Computational Intelligence Magazine*, 11(3):34–44, 2016.
- [FZ15] Xing Fang and Justin Zhan. Sentiment analysis using product review data. *Journal of Big Data*, 2(1):5, 2015.
- [JZ14] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.
- [Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.
- [LM14] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.
- [MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [MSC⁺13] 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.
- [OZLL15] X. Ouyang, P. Zhou, C. H. Li, and L. Liu. Sentiment analysis using convolutional neural network. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2359–2364, Oct 2015.
- [SL13] Kuldeep Sharma and King-Ip Lin. Review spam detector with rating consistency check. In *Proceedings of the 51st ACM Southeast Conference*, page 34. ACM, 2013.

- [SPW⁺13] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.
- [TQL15] Duyu Tang, Bing Qin, and Ting Liu. Learning semantic representations of users and products for document level sentiment classification. In *ACL (1)*, pages 1014–1023, 2015.
- [TQLY15] Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. User modeling with neural network for review rating prediction. In *IJCAI*, pages 1340–1346, 2015.

Curriculum Vitae

Graduate College
University of Nevada, Las Vegas

Nishit Shrestha

Degrees:

Bachelor's in Computer Engineering (B.E.)
Kathmandu Engineering College

Thesis Title: DEEP LEARNING IMPLEMENTATION FOR COMPARISON OF USER REVIEWS AND RATINGS

Thesis Examination Committee:

Chairperson, Fatma Nasoz, Ph.D.
Committee Member, Justin Zhan, Ph.D.
Committee Member, Kazem Taghva, Ph.D.
Graduate Faculty Representative, Qing Wu, Ph.D.