Abstractive User Review Consolidation

CS492 Project

CSU16120 MDL16CS043 Bhairavi Sameer Shah
CSU16123 MDL16CS046 Diya Liza Varghese
CSU16138 MDL16CS074 Michelle Elizabeth
CSU16159 MDL16CS117 Theres Mary Jose
B. Tech Computer Science & Engineering

Department of Computer Engineering
Model Engineering College
Thrikkakara, Kochi 682021
Phone: +91.484.2575370
http://www.mec.ac.in
hodcs@mec.ac.in

June 2020

Model Engineering College, Thrikkakara
Department of Computer Engineering

C E R T I F I C A T E

This is to certify that, this report titled ***Abstractive User Review Consolidation*** is a bonafide record of the work done by

CSU16120 MDL16CS043 Bhairavi Sameer Shah
CSU16123 MDL16CS046 Diya Liza Varghese
CSU16138 MDL16CS074 Michelle Elizabeth
CSU16159 MDL16CS117 Theres Mary Jose

Eighth Semester B. Tech. Computer Science & Engineering

students, for the course work in **CS492 Project**, which is the second part of the two semester project work, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of **A. P. J. Abdul Kalam Technological University**.

Guide

Aysha Fymin Majeed
Assistant Professor
Computer Engineering

Coordinator

Manilal D L
Associate Professor
Computer Engineering

Head of the Department

Manilal D L
Associate Professor
Computer Engineering

June 16, 2020

Acknowledgements

This project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere gratitude to all of them.

First of all, we would like to thank our esteemed Principal, Prof. (Dr.) Vinu Thomas, for his guidance and support in maintaining a calm and refreshing environment to work in and also for providing the facilities that this work demanded.

We are highly indebted to our Project Coordinator and Head of the Department, Manilal D L, Associate Professor for his guidance, support and constant supervision throughout the duration of the work and for providing all the necessary information and facilities required.

We would like to thank our Project Guide, Aysha Fymin Majeed for her support and valuable insights and also for helping us out in correcting any mistakes that were made during the course of this work.

We offer our sincere gratitude to all our friends and peers for their support and encouragement that helped us get through the tough phases during the course of this work.

Last but not the least, we thank the Almighty God for guiding us through and enabling us to complete the work within the specified time.

Bhairavi Sameer Shah

Diya Liza Varghese

Michelle Elizabeth

Theres Mary Jose

Abstract

E-commerce websites allow customers to leave reviews for various products. There are usually hundreds of reviews for a single product, each review could be lengthy and repetitive. A customer would find it difficult to make a well-informed decision after reading all the reviews. Therefore, automatic review summarization has a huge potential to help customers by providing an authentic summary of the reviews found online on the E-commerce sites. We propose the method of abstractive summarization, which provides more accurate summaries and are closer to human generated summaries. The system also provides the general sentiment of the summaries generated which will help the customers make a decision quickly. The sentiment of the summaries would help the customers know the tone of the text.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information is always available in plenty in the modern era of the Internet. Anything that happens in any part of the world reaches the other side in the blink of an eye, and hence, access to information is the last thing that you will need to worry about. According to Google Search Engine results stats, around 1 billion articles, posts or information of any kind is released for a single focused keyword every day. And, for a consumer, information available in the web will remain incomprehensible, unless it is transformed in a way that would help them understand the data in the most comprehensive way possible.

The best way to provide the right amount of information that a user requires is to limit the content into precise and accurate points. Providing the gist of a whole lot of content will reduce the overhead of processing unwanted information for consumers, and the difficulty of providing meaningful data for the providers. This is where summaries find their way into the scenario.

A summary is a subtle way of representing a lot of information in a minimal form. This seems to be profitable on a lot of platforms. Now, making a summary manually would require the user to read all the content again, which nullifies the sole aim of summaries. This is where an automatic summary generator comes into the picture.

Automatic summary generators provide you with a summary of the input text that you provide to the system. The automated summary generator can find its application in different forms, in educational fields, content creation, E-commerce, marketing, etc. As long as huge amounts of information needs to be processed in daily life, an automatic summarizer will also be pertinent.

1.1 Proposed Project

Automatic summarization has great potential in the E-commerce field. E-commerce websites allow customers to leave reviews for various products. There are usually hundreds of reviews for a single product and each review could be lengthy and repetitive. A customer would find it difficult to make a well-informed decision due to the huge amount of data available for a single product. Therefore, automatic review summarization has a huge potential to help customers by providing an authentic summary of the reviews found on the E-commerce sites.

1.1.1 Problem Statement

The existing systems for summarization do not consider the sentiment of the user reviews which are necessary to produce sound reviews of products. Also, some methods consider only a single document for summarization while others do not consider repetition of semantically equivalent words. To overcome these issues, we introduce the system Abstractive User Review Consolidation.

1.1.2 Proposed Solution

We propose a Sequence to Sequence (Seq2Seq) model with Attention Mechanism which consists of an encoder, decoder, and attention layer to perform abstractive summarization of user reviews. The dataset includes user reviews of products from E-commerce websites. The data from social networking platforms which are sentimentally rich are used to train a Naive Bayes model to identify the sentiment of the review text. We use lemmatization to avoid repetition of semantically equivalent words.

Chapter 2

System Study Report

2.1 Literature Survey

1. A Multi-View Abstractive Summarization Model Jointly Considering Semantics and Sentiment[2]

   Short text summarization generates a short summary consisting of only a few sentences which captures the main idea of the original article. Usually, summaries are produced considering only the semantic meaning of the text. This paper jointly considers semantics and sentiment and proposes a model that uses encoder-decoder recurrent neural networks for semantic view and Sentiment Embedding(SE) and Sentiment Memory(SM) for sentiment view.The proposed multi-view model, contains a basic summary component which performs sentence compression using encoder-decoder and a sentiment component which uses SE to extract and represent the sentiment features automatically, and SM to capture the sentiment dynamics during the process. The encoder reads the input sentences dynamically as a sequence of vectors into a context vector. The decoder predicts the next word based on the context vector and the previously predicted words and uses Attention Mechanism for better performance. Bi-LSTM model was used as sentiment classifier and ROUGE was used as the evaluation metric.

2. Generating Abstractive Summaries Using Sequence to Sequence Attention Model[3]

   Automatic text summarization using abstractive methods is significant in the field of information retrieval. Abstractive methods are effective as they closely resemble the summarization techniques adopted by humans. This paper applies Deep Neural Network model, namely Sequence to Sequence model to summarise research articles by considering the introduction and conclusion sections. Temporal attention mechanism is used to subdue the occurrence of repetitive words or sentences in summaries. The temporal attention model keeps record of the prior attentional weights produced by the decoder and restrains the decoder to attend the same chunks of the documents in the subsequent time stamps. ROUGE has been used as the evaluation metric for comparing sequence to sequence model with global attention and temporal attention.

3. Abstractive Multi-Document Summarization[4]

   According to this paper, summarization can be done by 3 methods: compressive summarization, extractive summarization and abstractive summarization. Abstractive summarization is the best method among these, as new sentences can be created and repetition can be avoided. There are two approaches for performing abstractive summarization of multiple documents. First is phrase selection and merging which uses linear optimization method to obtain an optimal solution for a summary. It uses the noun phrase (NP) or verb phrase (VP) to separate the sentences into phrases. Salience Score is provided to each phrase and a new sentence is generated based on the maximum salience score (Summary Content Unit). The second method is the Semantic Information Extraction Approach which uses BSU basic semantic network which depicts semantic information. This network is analysed to create the summary. BSU is extracted, a semantic link network is created, reduced and thus, the sentence is generated. The clustered semantic graphs use semantic role labelling and ranking algorithms.

4. Extracting Aspects and Mining Opinions in Product Reviews Using Supervised Learning Algorithm[5]

   Semantic analysis and opinion mining aim to automatically extract opinions expressed in the user-generated content. Opinion mining tools allow businesses to understand new product opinion, product sentiments, brand view and reputation management. There are three general categorizations for opinion mining tasks: document-level, sentence-level, and phrase-level. Aspect level opinon mining extracts aspect opinions from documents. The projected system identifies the number of positive and negative opinions of each aspect in online reviews. The system preprocesses data and performs sentence and aspect extraction. Stop word removal, stemming and POS tagging are the processes in data preprocessing. Naive Bayesian algorithm is used to identify opinions.

5. An Abstractive Summarizer Based on Improved Pointer-Generator Network[6]

   The deep learning model has been widely applied in natural language processing and has achieved improved results. This paper introduces the decoder attention mechanism in the pointer-generator network to improve the accuracy and fluency of the summary. The multi-hop attention mechanism is introduced to improve the copy probability distribution. The pointer-generator network is an improvement of the sequence-to-sequence attention model which includes the encoder, decoder and attention mechanism. The coverage vector, used as a new input to compute the attention mechanism, tells the model which words it has focused on before so that fragment duplication is avoided. Using attention mechanism at the decoder ensures that generating ability has improved since both the original text and the partial summary are considered at every time-step. It follows the process of constantly observing the original text and the generated summary when people do information extraction, and improve the copy probability based on the baseline pointer-generator network model. ROUGE metric was used to evaluate the performance of the system.

6. Dual Encoding for Abstractive Text Summarization[7]

   Recurrent neural network-based sequence-to-sequence attention models have proven effective in abstractive text summarization. In this paper, we model abstractive text summarization using a dual encoding model. The proposed dual encoding model consists of a primary encoder, a secondary encoder, and a decoder. It conducts the primary encoder and decoder as the standard attentional encoder–decoder model. The secondary encoder is based on the input and the previously produced output, and generates a new context vector as an additional input of the decoder. The primary encoder calculates the semantic vectors for each word in the input sequence. The secondary encoder first calculates the importance weight for each word in the input sequence and then recalculates the corresponding semantic vectors. The decoder with attention mechanism decodes by stages and generates a partial fixed-length output sequence at each stage. The context vector makes the decoder obtain more meaningful information and generate better output. A pointer mechanism is used to handle OOV words and a coverage mechanism to address the problem of repetition. Besides, an additional multistep decoding operation in the decoder models the decoded content at each stage as a semantic feature vector helping avoid the problem of repetition. The model was evaluated using ROUGE-1, ROUGE-2 and ROUGE-L for unigram, bigram and long text respectively.

7. Multi-Document Abstractive Summarization Based on Predicate Argument Structure[8]

   The summary is generated based on the predicate argument structure of the sentences. To obtain the predicate argument structure we use semantic role labelling. In the first step, we extract the predicate argument structure to represent text semantically. The second step includes grouping semantically similar predicate argument structure using hybrid approach of K-means (selected because of its better run-time) and agglomerative hierarchical clustering (selected because of its quality) by utilizing semantic similarity measures. Features are extracted from this, and finally the top ranked predicate argument structures are selected for summary. The sentences are formed using language generation techniques.

8. Evaluation of Automatic Text Summarizations Based on Human Summaries[9]

   Two sets of summaries of the same data were obtained, one by automatic text summarization and the other by manually producing summaries. Automatic summaries were obtained by using fuzzy method and vector approach. The ROUGE calculation clearly indicated that human summaries were much more accurate than automatically generated summaries. Though, summaries produced by fuzzy method were much more acceptable and understandable compared to the ones produced by vector approach.

9. Clustered Genetic Semantic Graph Approach for Multi-Document Abstractive Summarization[10]

   This paper uses a clustered genetic semantic graph approach for multi-document abstractive summarization. The semantic graph is constructed by ensuring the graph vertices represent the predicate argument structures (PASs), extracted automatically by employing semantic role labelling (SRL) and the edges of graph correspond to semantic similarity weight determined from PAS-to-PAS semantic similarity, and PAS-to-document relationship. The

PAS-to-document relationship is expressed by different features, weighted and optimized by genetic algorithm. The salient graph nodes (PASs) are ranked based on modified weighted graph based ranking algorithm. The clustering algorithm is performed to eliminate redundancy in such a way that representative PAS with the highest salience score from each cluster is chosen, and fed to language generation to generate summary sentences.

10. Integrating Extractive and Abstractive Models for Long Text Summarization[11]

    In this paper, a two-phase approach towards long text summarization is employed, namely, EA-LTS. In the extraction phase, it conceives a hybrid sentence similarity measure by combining sentence vector and Levenshtein distance, and integrates it into graph model to extract key sentences. In the abstraction phase, it constructs a recurrent neural network based encoder-decoder, and devises pointer and attention mechanisms to generate summaries. A real-life long text corpora, collected from sina.com is used for testing.

11. Multi Document Abstractive Summarization Using ILP Based Multi Sentence Compression[12]

    This approach finds the most important documents from multiple documents. It is done using LexRank. Sentences are aligned to generate clusters of similar sentences. The most relevant and linguistically well formed sentences are selected from the cluster of sentences formed. Then, it generates K-shortest paths from the sentences in each cluster using word-graph. Finally, sentences from the shortest paths generated from all clusters are selected using Integer Linear Programming (ILP). After selecting sentences from the cluster, the summary sentences are generated. Constraints were imposed on the number of sentences selected from each cluster to avoid redundancies. The ROUGE evaluation is done to ensure the quality. It also undergoes manual evaluation.

12. A Neural Attention Model for Abstractive Sentence Summarization[13]

    In this work, a fully data-driven approach to abstractive sentence summarization is used. This method uses a local attention-based model that generates each word of the summary conditioned on the input sentence. The network contains both neural language model and an encoder. The language model estimates the probability of the next word. It is adapted from a standard feed-forward neural network model. The attention model is combined with a generative algorithm to produce accurate abstractive summaries. Since this system makes no assumptions about the vocabulary of generated summary, it can be directly trained on any document-summary pair.

13. Multi-Document Abstractive Summarization Using Chunk-Graph & Recurrent Neural Network[14]

    In this method, several sentences in the document set are grouped into clusters manually or by clustering algorithms. Sentences in the same clusters are compressed to one sentence using chunk-graph (CG). To construct CG, the chunks are labelled in each sentence in a cluster. The chunk units are taken as nodes & relations among them as edges. After constructing CG, the nodes which refer to the same entities are merged to one node. Beam search & RNNLM is applied to find the best path in CG as the summary of a cluster. In the end, all

the summaries generated from the CGs are ranked and these scores decide the order of each sentence in the final summary.

14. Sequence Generative Adversarial Network for Long Text Summarization[15]

    A new adversarial training framework is used for text summarization task. This framework consists of two models: A generator that generates summaries and a discriminator that evaluates the generated summaries. The co-training of generator and discriminator is guaranteed by reinforcement learning (RL) in which the discriminator improves its performance by learning from more and more training samples and the generator gets feedback from the discriminator and improves itself. For effective summarization of long text, the attention mechanism is introduced in the generator, on both encoder and decoder and a triple-RNN model is used in the discriminator.

15. Multiple Text Document Summarization System Using Hybrid Summarization Technique[16]

    This paper presents a novel approach to generate abstractive summary from extractive summary using WordNet ontology. There are 2 main blocks in this model: extractive and abstractive. The sentences are extracted from documents and tokenization is performed. Repetitive sentences and words are removed. Now, the weight of the sentence is calculated and extractive summary is generated. Abstractive summary is the second phase of multiple document text summarization. By applying heuristic rules, important nodes from the extractive summary is found. Ontology represents the domain which talks about the same topic having same knowledge. It provides a vocabulary and a set of synset. Next, meaningful terms are produced by preprocessing and classifier classifies those terms. WordNet is a lexical database of English. It defines meanings and models. It consists of synsets, which provides different semantic relationships.

16. Multi-Layered Sentimental Analytical Model for Product Review Mining[17]

    The product review classification is the mechanism used to analyze the sentiment or opinion in the reviews posted by the users to prepare the product review. In this paper, the mechanism is to use opinion mining over text review data for the generation of product review report, which is based on multiple features. The product reviews undergo aspect based summarization after sentiment analysis. Stemming porter is used to convert non-root words into root words. There are 3 levels of sentiment analysis: document-level, sentence level, aspect and entity level. The system uses review analytical algorithm and automatic review classification algorithm.

Sl. No	Title	Author	Year	Technique	Advantages	Disadvantages
1	A Multi-view Abstractive Summarization Model Jointly Considering Semantics and Sentiment	Moye Chen, Lei Li, Wei Liu	2018	Encoder-Decoder Recurrent Neural Network	Multi-view model extracts sentiment features automatically. Low dimension vector for SE proved to be a powerful representation since sentiment categories are not complex.This model was proven to be an improvement to existing baseline systems	Sentiment labeling should be done manually since there is no dataset with labeled sentiment. More attention should be given to intra-relationship of words. Sentiment factor was not considered while computing loss function.
2	Generating Abstractive Summaries Using Sequence to Sequence Attention Model	Tooba Siddiqui, Jawwad Ahmed Shamsi	2018	Sequence to Sequence Attention Model	The rouge score of the temporal attention model is higher as compared to the rouge score of the global attention model.	Computational cost for summarisation increases with documents, layers and iterations.

Sl. No	Title	Author	Year	Technique	Advantages	Disadvantages
3	Abstractive Multi-document Summarization	Ranjitha N S, Dr. Jagadish S Kallimani	2017	Linear Optimization Method, Semantic Information Text Approach	Repetition avoided using agglomerative hierarchical clustering. This approach is applicable with respect to any domain and requires no interventions of human experts. Provides PAS representation with high salience value.	It fails for words with opinions.
4	Extracting Aspects and Mining Opinions in Product Reviews Using Supervised Learning Algorithm	A.Jeyapriya, C.S. Kanimozhi Selvi	2015	Sentiment Analysis/Aspect Based Opinion Mining	Positive and negative opinions are separated from the documents. Sentiment orientation algorithm is used to find the probability of positive and negative opinions. Good accuracy.	New sentences are not created.
5	An Abstractive Summarizer Based on Improved Pointer-Generator Network	Wenbo Nie, Wei Zhang, Xinle Li, Yao Yu	2019	Pointer-Generator Model, Attention Mechanism	It has a higher Rouge score than the basic pointer-generator model. It can handle out-of-vocabulary(OOV) words.	Repetition is not fully eliminated.

Sl. No	Title	Author	Year	Technique	Advantages	Disadvantages
6	Dual Encoding for Abstractive Text Summarization	Kaichun Yao, Libo Zhang, Dawei Du, Tiejian Luo, Lili Tao, Yanjun Wu	2018	Dual Encoding Model	Uses an enhanced repetition avoid mechanism which improves the quality of the generated summary. The secondary encoding is more likely to fulfil a fine and selective encoding to help decoder produce better summary. More suitable for long sequence generation tasks.	A large decoding length makes the secondary encoder out of function. A small decoding length is not able to capture enough information and increases computational cost due to more secondary encoding operation.
7	Multi-document Abstractive Summarization Based on Predicate Argument Structure	Alshaina S, Ansamma John, Aneesh G Nath	2017	Predicate Argument Structure	Better computation time than existing systems. Abstractive text summarization produces highly meaningful, knowledge rich and less redundant summary. Provide a viable solution than other algorithms.	Feature selection is made randomly.

Sl. No	Title	Author	Year	Technique	Advantages	Disadvantages
8	Evaluation of Automatic Text Summarization Based on Human Summaries	Farshad Kiyoumarsi	2014	Fuzzy Method, Vector Approach	Quality of summaries produced by humans and by using automatic summarization was comparable. Automatic more efficient when volume of data increases. Most of the time, readers are able to understand the summaries using their common sense and make the summaries coherent in their mind.	Automatically generated summaries are not coherent and intelligent as human summaries, since humans can think and decide on the best option.
9	Clustered Genetic Semantic Graph Approach for Multi-document Abstractive Summarization	Atif Khan, Naomie Salim, Haleem Farman	2016	Clustered Genetic Semantic Graph Approach	The semantic similarity measures assists in detecting redundancy by capturing semantically equivalent predicate argument structures thereby improving results. Does not require any intervention of human experts.	The proposed approach assumes semantic structure of sentence. The impact of Cross-Document Structural Theory (CST) relations for multi-document abstractive summarization is not considered.

Sl. No	Title	Author	Year	Technique	Advantages	Disadvantages
10	Integrating Extractive and Abstractive Models for Long Text Summarization	Shuai Wang, Xiang Zhao, Bo Li, Bin Ge, Daquan Tang	2017	Graph Model, Recurrent Neural Network	Integration of the state-of-art models leverages the advantages of both extractive and abstractive summarization methods, and achieves significant performance improvements when dealing with long text. A real world dataset from financial domain for long text summarization is used.	Requires a large-scale structured training data. Not capable of multi-document text summarization.
11	Multi Document Abstractive Summarization using ILP Based Multi Sentence Compression	Siddhartha Banerjee, Prasenjit Mitra, Kazunari Sugiyama	2015	Inter Linear Programming Model(ILP)	Achieves promising results on informativeness and readability. Log probability score is assigned as an indicator of linguistic quality. ILP is a novel methodology to be used.	Requires manual evaluation
12	A Neural Attention Model for Abstractive Sentence Summarization	Alexander M. Rush, Sumit Chopra, Jason Weston	2015	Neural Network	Can easily scale to train on a large amount of data. Can be trained directly on any document-summary pair.	Repeating semantic elements. Altering semantic roles. Improper generalization.

Sl. No	Title	Author	Year	Technique	Advantages	Disadvantages
13	Multi-Document Abstractive Summarization using Chunk-Graph and Recurrent Neural Network	Jianwei Niu, Huan Chen, Qingjuan Zhao, Limin Sun, Mohammed Atiquzzaman	2017	Chunk Graph and Recurrent Neural Network	Using CG instead of words as basic unit greatly reduces the graph size. Smaller graph size can reduce the computation load effectively.	CG filters the sentence paths in low linguistic quality.
14	Sequence Generative Adversarial Network for Long Text Summarization	Hao Xu, Yanan Cao, Ruipeng Jia, Yanbing Liu, Jianlong Tan	2018	Generative Adversarial Network(GAN)	Discriminator model provides additional improvement in performance. It is more prominent in effective summarization of long source text.	The generated summary still contains repeating phrases. This model uses supervised learning, which rely on high quality datasets which is scarce.
15	Multiple Text Document Summarization System using Hybrid Summarization Technique	Harsha Dave, Shree Jaswal	2015	Hybrid Technique, WordNet Ontology	The generated abstractive summary is in well-compressed, grammatically correct and human readable format.	As the size of the document increases, system will take more time to generate summary.
16	Multi-Layered Sentimental Analytical Model For Product Review Mining	Jagbir Kaur, Meenakshi Bansal	2016	Sentiment Analysis/Opinion Mining	The proposed system has very high accuracy(99 percent). It is efficient and accurate in terms of assessed parameters.	Does not consider compression error.

Table 2.1: Literature Survey

2.2 Proposed System

The existing systems for summarization do not consider the sentiment of the user reviews which are necessary to produce genuine reviews of products. Also, some methods consider only a single document for summarization while others do not consider repetition of semantically equivalent words. To overcome these issues, we introduce the system Abstractive User Review Consolidation.

We propose a Sequence to Sequence (Seq2Seq) model with Attention Mechanism which consists of an encoder, decoder, and attention layer to perform abstractive summarization of user reviews. The dataset includes user reviews of products from e-commerce websites. The tweets from social networking platforms like Twitter which are sentimentally rich are used to train a Naive Bayes model to identify the sentiment of the review text. We use lemmatization to avoid repetition of semantically equivalent words. The automatic summary is generated out of the reviews received from the e-commerce sites and the repetition of the words are taken into account while creating the summary. The tweets are used to train the model with the sentiment of words that are mostly used in the online space. The proposed system gives a concise summary and sentiment of the reviews and average-rating of the product selected by the user.

Chapter 3

Software Requirement Specification

3.1 Introduction

This section includes the purpose of Software Requirement Specification, document conventions, intended audience and reading suggestions, project scope and overview of developers responsibilities.

3.1.1 Purpose

This document is the software requirement specification for the project titled ABSTRACTIVE USER REVIEW CONSOLIDATION - VERSION 1.0. The aim of this project is to develop a system to summarize and generate reviews about the products available on e-commerce platforms. This document maintains all the functions and specifications of the system and contains the detailed descriptions for the requirements specified.

3.1.2 Document Conventions

- NLP - Natural Language Processing
- NLTK - Natural Language Tool Kit
- API - Application Programming Interface
- IDE - Integrated Development Environment
- GUI - Graphical User Interface
- Seq2Seq - Sequence to Sequence
- LSTM - Long Short Term Memory
- OOV - Out-Of-Vocabulary
- POS - Parts-Of-Speech
- HTTP - Hyper Text Transfer Protocol
- HTTPS - Hypertext Transfer Protocol Secure

- GPU - Graphical Processing Unit
- CPU - Central Processing Unit
- UI - User Interface
- DOM - Document Object Model
- HTML - Hyper Text Markup Language
- JSON - JavaScript Object Notation
- DFD - Data Flow Diagram
- SRS - Software Requirement Specification
- SDD - Software Design Document

3.1.3 Intended Audience and Reading Suggestions

The document is intended for stakeholders, developers, project managers, testers and documentation writers. The rest of this SRS contains further details of the project which includes the scope of the project as well as software and hardware requirements.

3.1.4 Project Scope

Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning. With the rise in usage of e-commerce platforms, it is important to provide a concise form of user reviews about the products available online. The web application would also be useful for sellers of the products, making it easier for them to go through customer feedback by going through their products on the application.
The project aims to create a web application, which can be used to obtain consolidated form of customer reviews of various products available on e-commerce websites. This will help buyers to go through different products and their major pros and cons easily, without having to spend hours reading through all the reviews available. The application would also provide the general sentiment of other buyers, obtained from the sentiment analysis of the reviews. This would provide the buyer with the percentage of people satisfied and dissatisfied with the product, which would further help in the decision to purchase the product.

3.1.5 Overview of Developer's Responsibilities

The responsibilities of the developer include the following.

- Develop and integrate an efficient algorithm by incorporating abstractive summarization using Seq2Seq Model with Attention Mechanism.
- Reduce complexity of the text by making use of various preprocessing techniques such as lemmatization.

- Improve the understanding of the review summary by identifying the general sentiment of the reviews; generated using a trained Naive Bayesian model.
- Display the summarised review, average rating and the sentiment so as to improve readability.
- Provide the end user with a clear and smooth interface.
- Configure a fast and efficient server system.

3.2 Overall Description

This section includes product perspective, product functions, user classes and characteristics, operating environment, design and implementation constraints, user documentation, general constraints and assumptions and dependencies of the project.

3.2.1 Product Perspective

The existing systems for summarization use extractive methods which identify important sections of the text and generate them verbatim producing a subset of the sentences from the original text. Also, they do not consider the sentiment of the user reviews and some methods consider only a single document for summarization. On the other hand, the abstractive summarization method reproduces important material in a new way after interpretation and examination of the text using advanced natural language techniques to generate a new shorter text that conveys the most critical information from the original one along with the sentiment quotient of the generated summary.

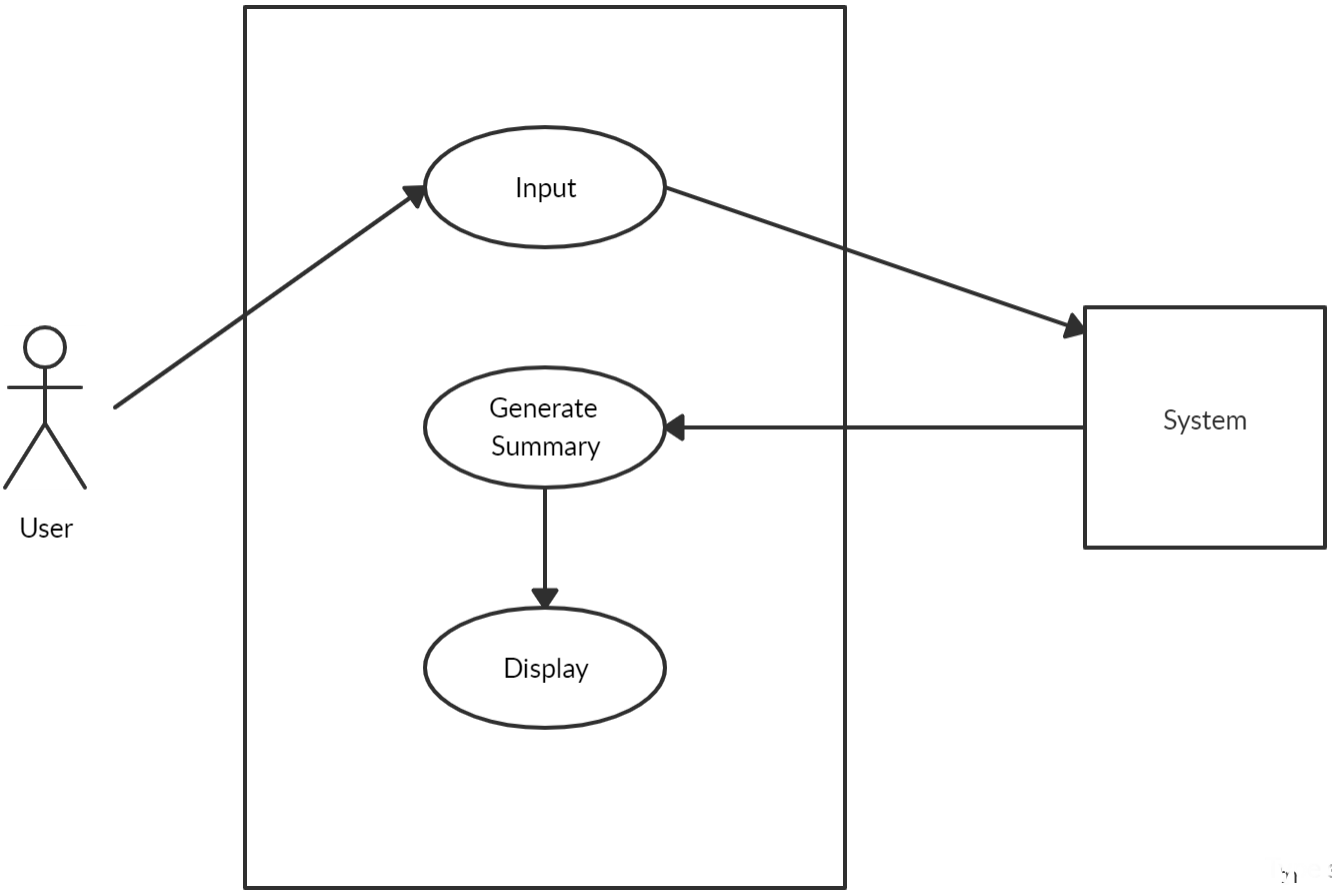

Figure 3.1: System Overview

3.2.2 Product Functions

The product aims to implement the following functions:

- Generate an abstractive summary of the available reviews of the selected product using a trained deep learning model.
- Determine the general sentiment of the users for the selected product.
- Compute the average rating of the selected product.
- Display product data including the name, image and price.

3.2.3 User Classes and Characteristics

End Users

- **Online Shoppers**
  The application is intended to be used by online shoppers who directly buy goods over the internet. They simply need to have access to the internet and a basic knowledge on using web pages. Increased exposure to technology increases the probability of shopping online and thus using this application.
- **Manufacturers**
  This application can also be used by product manufacturers in order to receive the customers opinions and feedback on their product in brief. They may use this data to improve their strategies and to produce better products which satisfies the customers' needs.

3.2.4 Operating Environment

A hardware device like a smartphone, tablet, laptop or desktop with good internet connectivity is required. The user will interact with the system via a web application. The device used to access the system must run on Linux or Windows Operating System.

3.2.5 Design and Implementation Constraints

- The server must run without a time constraint.
- Increase in number of reviews affects the computational cost for summarization.
- There may be problems with out-of-vocabulary(OOV) words.
- The language used in the reviews must be English.

3.2.6 User Documentation

The system has a simple GUI, so there is no need for additional user documentation. Basic understanding of computers and the internet is sufficient for using this system.

3.2.7 General Constraints

- Training of the seq2seq model requires a lot of time depending on the amount of data used.
- CPU requirements are relatively high and increases with the amount of training data.
- Training a huge amount of data is infeasible.
- A stable internet connection is required to use the web application.

3.2.8 Assumptions and Dependencies

- The interface of the system will be easy to use and accessible without any time or location constraints.
- The user must have an updated web browser.
- The application assumes a stable network connection.
- The text parsed by the application is assumed to be in English.

3.3 External Interface Requirements

This section describes the user interfaces, hardware interfaces, software interfaces and communication interfaces.

3.3.1 User Interfaces

The user will interact with the system via a web application. The product name will be selected from an auto-complete drop-down list provided in the homepage. A 'Submit' button will process the input. The output will be displayed as a sweet alert box containing the name, image, price, summary of the reviews, sentiment of the reviews and rating of the product.

3.3.2 Hardware Interfaces

A hardware device like a smartphone, tablet, laptop or desktop with good internet connectivity is required to access the proposed system through a web browser. No specialised hardware is required.

3.3.3 Software Interfaces

The web application is programmed in Python with Flask as the framework for the application. React is used as the front end library along with Javascript. Libraries like Keras and Tensorflow would be used for machine learning and deep learning applications i.e., for training and testing various models. Pandas library would be used to manipulate the data for all computations along with the NLTK library for countless natural language processing of the review text.

3.3.4 Communication Interfaces

The system can be accessed via any popular web browsers like Chrome, Firefox etc. The communication between the application and the web-server will be done using HTTP. The application will send a HTTP request to the web-server. The HTTP POST method will be used to return the response. The web-server returns a HTTP response containing JSON data related to the request.

3.4 Hardware and Software Requirements

This section includes the hardware and software requirements of the proposed system.

3.4.1 Hardware Requirements

- System: **CPU - 2.4GHz** (Minimum)
  Model training and generating abstractive summary consumes a lot of system resources. Therefore a server with Clock speed of at least 2.4GHz is required.

- RAM: **10GB** (Minimum)
  The system deals with a large amount of memory requirements in order to train the model. Hence a RAM of at least 10GB is required.

- Hard Disk: **50GB** (or higher)
  The system deals with a large amount of data. Hence it requires a hard disk of 50GB.

3.4.2 Software Requirements

- Operating System: Any operating system (preferably Linux (64-bit), Windows)
  Linux, Windows is preferred because these are developer friendly, has a powerful shell, flexible and is the most popular

- Language: **Python 3.6.2, JavaScript**
  - **Python**
    Python is an interpreted, high-level, general-purpose programming language. It takes very less time to develop. It is typically 3-5 times shorter than other equivalent programming languages. It has powerful polymorphic list and dictionary types, for which rich syntactic support is built straight into the language.
  - **JavaScript**
    JavaScript, often abbreviated as JS, is a high-level programming language, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. JavaScript is the programming language of HTML and the Web.

- Frameworks: **Flask**
  - **Flask**
    Flask is a python framework used for building the Web interface. Flask is much more fully featured than other frameworks. Python Flask framework supports the use of human-readable website URLs. It also has its own bootstrapping tool. Flask separates a project into individual applications, where Pyramid and Django expect a project to be a single "application" with several views or models.
- IDE: **Google Colaboratory, Visual Studio Code**
  - **Google Colaboratory**
    Google Colaboratory is a free cloud service by Google and supports free GPU. It provides us with a cloud environment consisting of n1-highmem-2 instance machine type, 2 virtual CPU at 2.2GHz, 13GB RAM and 64GB Free Space. It has an idle cut-off 90 minutes and can be used to run code for free maximum for 12 hours.
  - **Visual Studio Code**
    Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring.
- Libraries:
  - **pandas**
  - **Keras**
  - **TensorFlow**
  - **sklearn**
  - **NLTK**
  - **pickle**
  - **React**
  - **json**
  - **numpy**
  - **BeautifulSoup**
  - **sweetalert**

3.5 Functional Requirements

This section briefly describes the functional requirements of the system.

3.5.1 Input Product Name

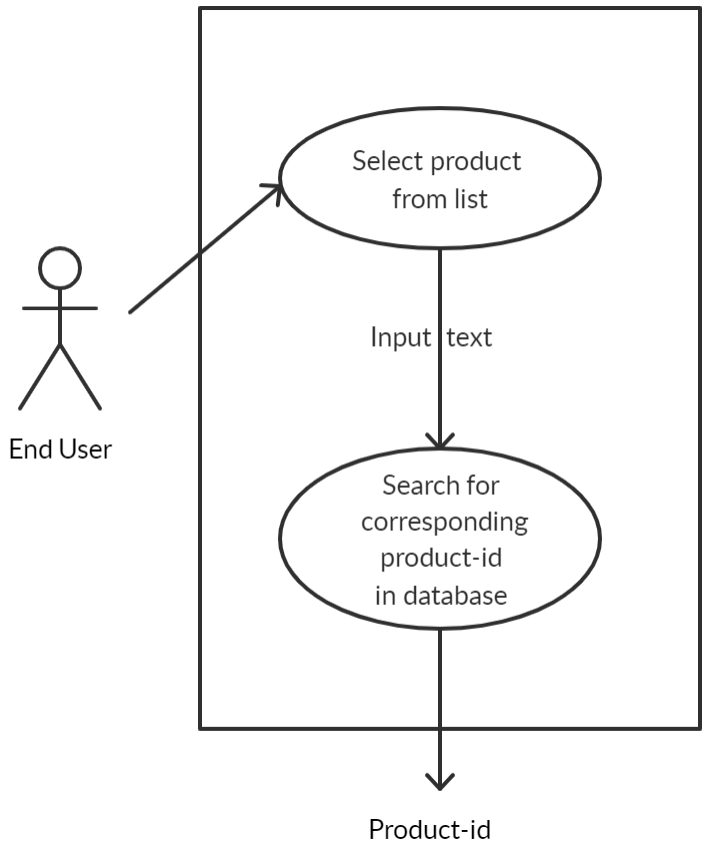

Figure 3.2: Usecase Diagram - Input Product Name

- The user inputs the product name via the auto-complete drop-down list provided in the web application.
- The input product name is mapped to its product ID.
- The product ID is then retrieved as partial output of this stage.

3.5.2 Search for Reviews

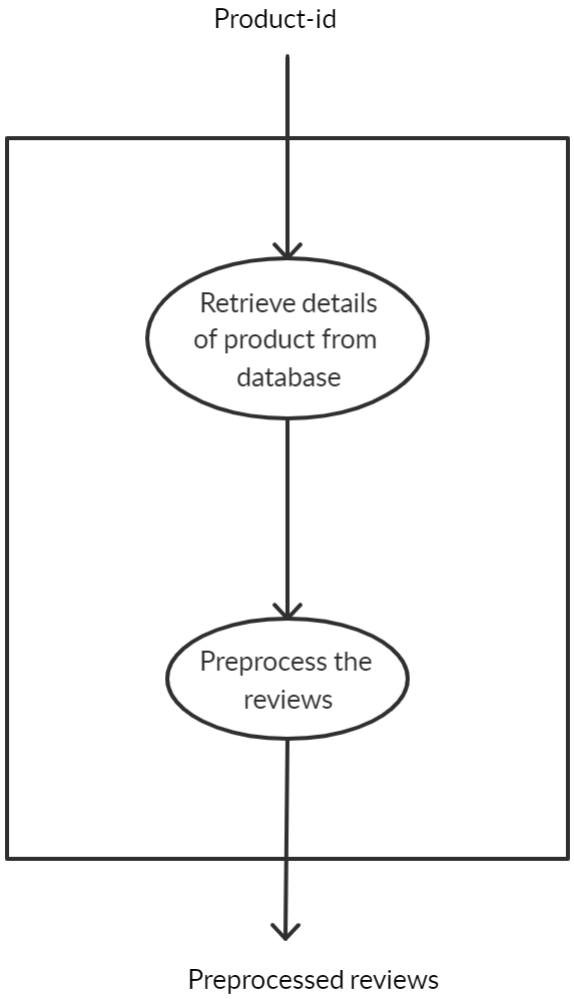

Figure 3.3: Usecase Diagram - Search Reviews

- The product ID is given as input to this stage.
- The product database is searched using this product ID to retrieve relevant reviews of the product.
- The review data from the database is then given for preprocessing.

3.5.3 Preprocessing

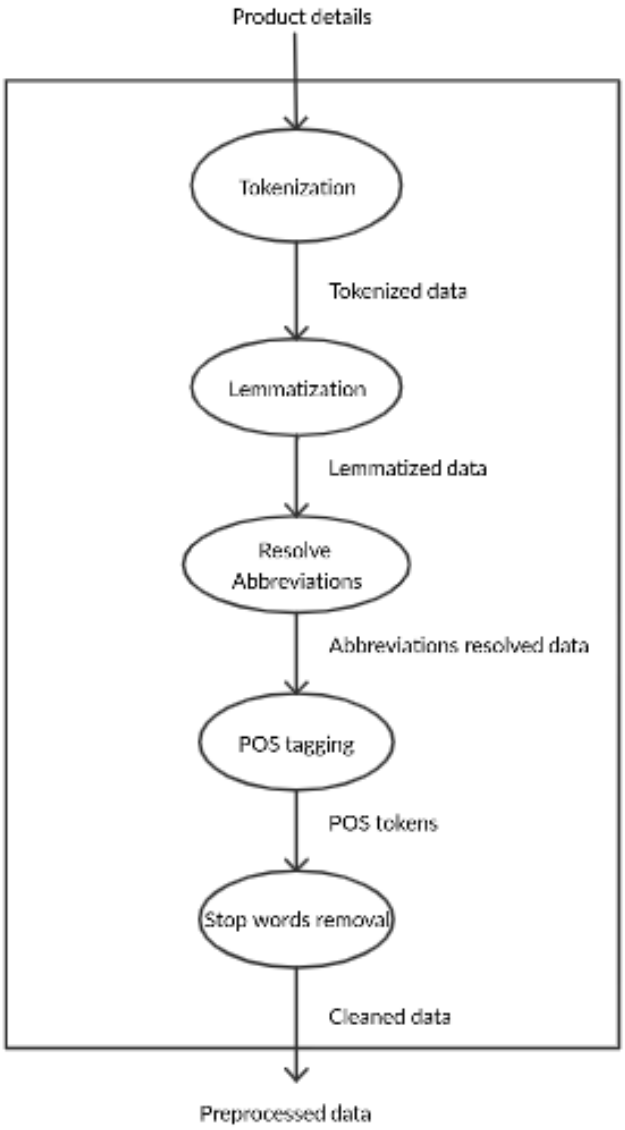

Figure 3.4: Usecase Diagram - Preprocessing

- The collected review data is given as input to this stage.
- The data undergoes tokenization, lemmatization, resolution of abbreviations, POS tagging and removal of stop words.
- Preprocessed data is obtained as output for this stage.

3.5.4 Sentiment Analysis

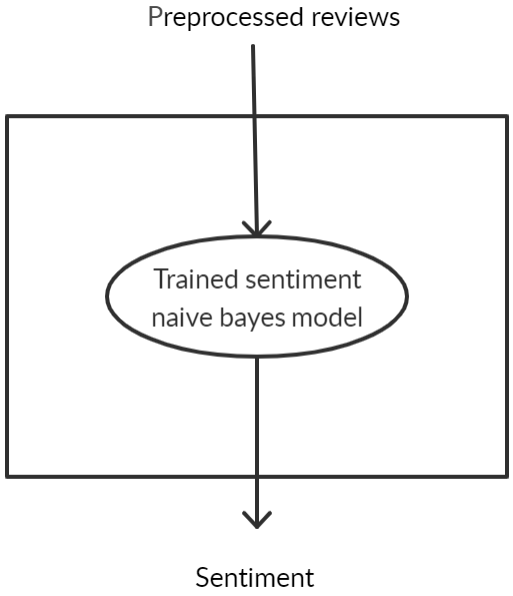

Figure 3.5: Usecase Diagram - Sentiment Analysis

- The preprocessed data is fed into the trained Naive Bayes machine learning model for sentiment analysis.
- The sentiment of the preprocessed data is generated, which is given to the web application.

3.5.5 Summary Generation

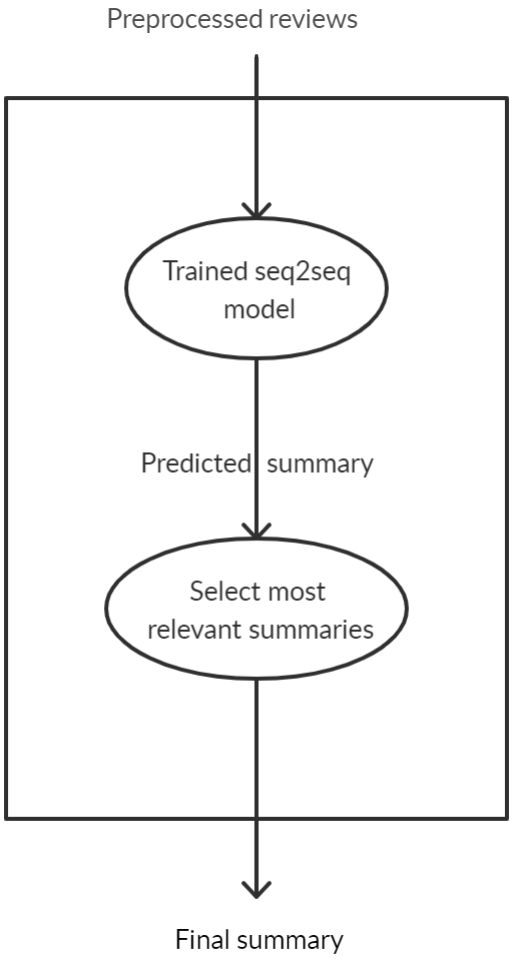

Figure 3.6: Usecase Diagram - Summary Generation

- Preprocessed data is fed into the trained machine learning model for generating summary.
- A summary is generated by the model, which is then given to the web application.

3.5.6 Display Output

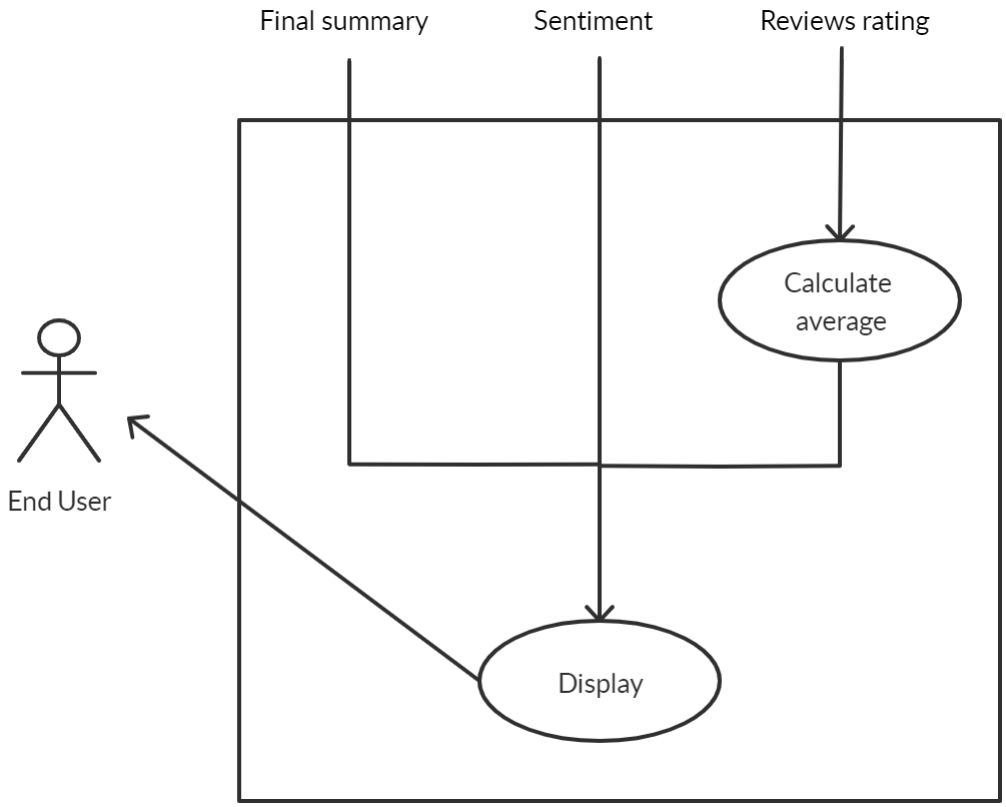

Figure 3.7: Usecase Diagram - Display Output

- The summary and the sentiment found from the trained model along with the basic information of the product is sent back to the web application.
- The details are displayed in the user interface components.

3.6 Non-Functional Requirements

This section describes performance requirements, safety requirements, security requirements and software quality attributes.

3.6.1 Performance Requirements

The time required to generate the output of the search should be minimum. The literal quality of the summary generated must be high. The summary generated must consider all aspects of the reviews, providing required weightages to important parts.

3.6.2 Security Requirements

The data given by the user is passed onto a secure server. HTTPS is used to establish the communication protocol. Data mining should not cause security breaches. So, precautions must be taken beforehand in terms of access and permissions.

3.6.3 Software Quality Attributes

- **Reliability**: The website can be used by multiple users concurrently. Any user can access the website.
- **Availability**: The system will be available 24 hours a day.
- **Maintainability**: The system is implemented as modules thus enhancing its maintainability.
- **Portability**: User can use the website at any time, anywhere as long as there is a stable internet connection.

3.7 Other Requirements

Besides the functional and non-functional requirements, there are also other requirements like ease of use and error tolerance. Ease of use describes the ease with which users can interact with the system and error tolerance says about the extent to which error is tolerated if it occurs in the system.

3.7.1 Ease of Use

- The system will be easy to handle with minimum delay.
- The system provides an easy and friendly web user interface.

3.7.2 Error Tolerance

- The system should have minimum accuracy level maintained in summary generation.
- It should be free from processing delays.

Chapter 4

System Design

4.1 System Architecture

The system is divided into 6 phases:

- **Phase 1** - Training the machine learning models
- **Phase 2** - Input data
- **Phase 3** - Data collection
- **Phase 4** - Data preprocessing
- **Phase 5** - Summary generation and sentiment analysis
- **Phase 6** - Output display

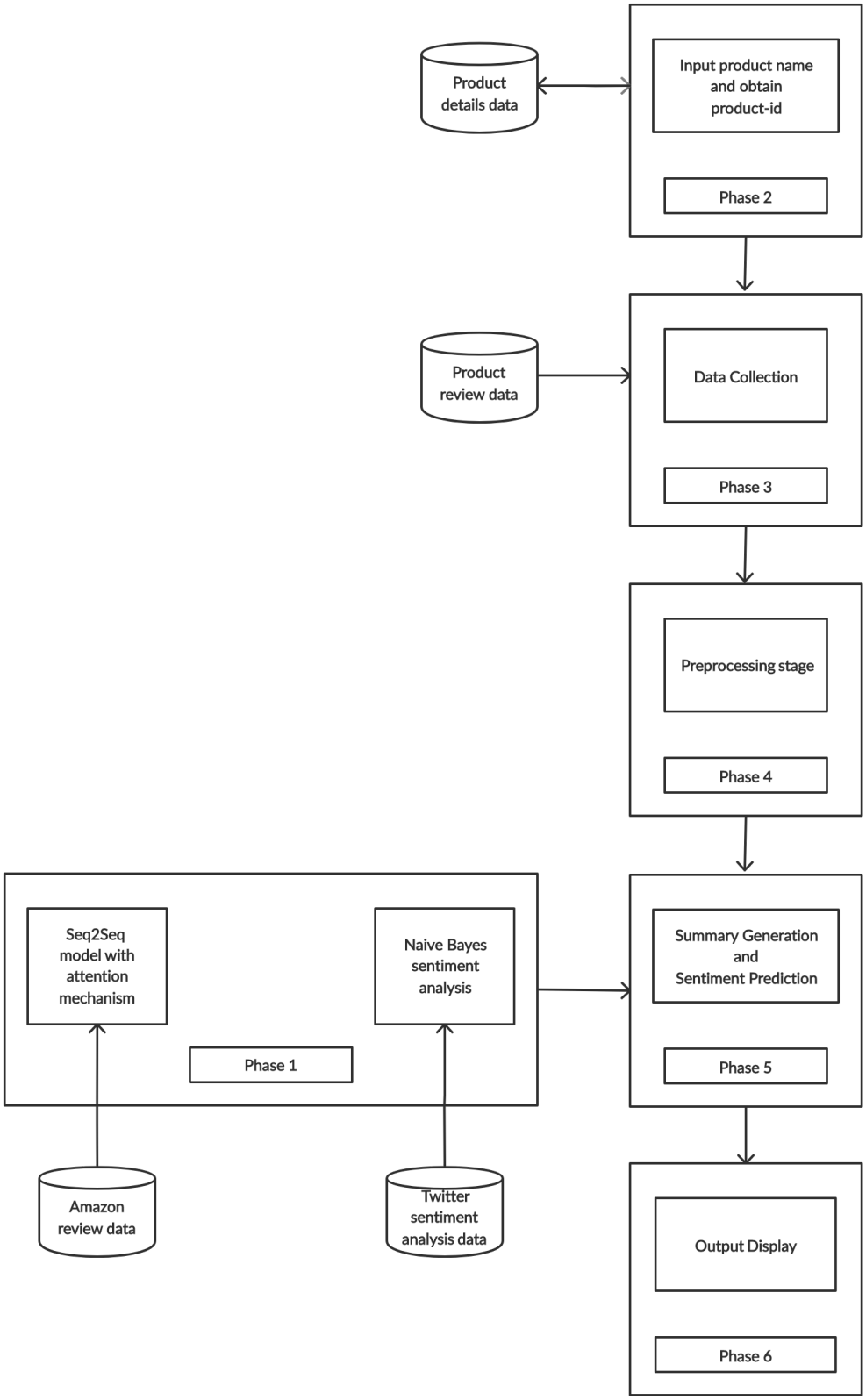

Figure 4.1: System Architecture

4.1.1 Training the Models

There are 2 machine learning models used in the system. One, to produce the summary of reviews and the other, to identify the sentiment of the reviews. A data set containing nearly 6,00,000 reviews of Amazon products and their summaries are fed into a Seq2Seq model with attention mechanism that includes an encoder, a decoder and an attention layer. The data set is used to train the model to produce summary of each review. This trained model is tested with a testing dataset of reviews to check its performance. The model is then stored for use to summarize the required reviews.

The Twitter sentiment analysis dataset, which contains tweets and their associated sentiment is used to train a Naive Bayes model to correctly identify the sentiment of any given sentence. The model is tested with sample sentences to check its accuracy.

4.1.2 Input Data

In an information system, the input is raw data that is processed to produce the output. The end user inputs the name of the product whose review summary is to be generated. The input is given via an auto-complete drop-down. The name of the product is mapped to its product ID and the same is sent to the next module.

4.1.3 Data Collection

Data is extracted from the product database using the product ID. The reviews about the selected product is collected and sent into the next phase which is the preprocessing stage.

4.1.4 Data Preprocessing

Preprocessing is done to obtain a more digestible form so that the machine learning algorithms can perform better. This phase includes tokenization, stop words elimination, lemmatization, POS tagging and resolution of abbreviations. This cleans up the data and converts it into an integer sequence which the model understands.

4.1.5 Summary Generation and Sentiment Analysis

Preprocessed data is fed into the trained seq2seq model for summary generation. A summary is generated by the model, which is then given to the web application. The summary generated is an abstractive summary, which generates new phrases and sentences that represent the most important information from the source text. The most relevant summaries are selected to be sent to the web application.

Preprocessed data is also sent to the sentiment analysis model. The percentage of positive and negative sentiment acquired from all the preprocessed reviews is generated and sent to the web application.

4.1.6 Output Display

The generated summary and the sentiment is sent to the web application. The details are presented to the user through a sweet alert. It contains the name, image, price, summary, rating and sentiment of the selected product.

4.2 Data Description

4.2.1 Database Design

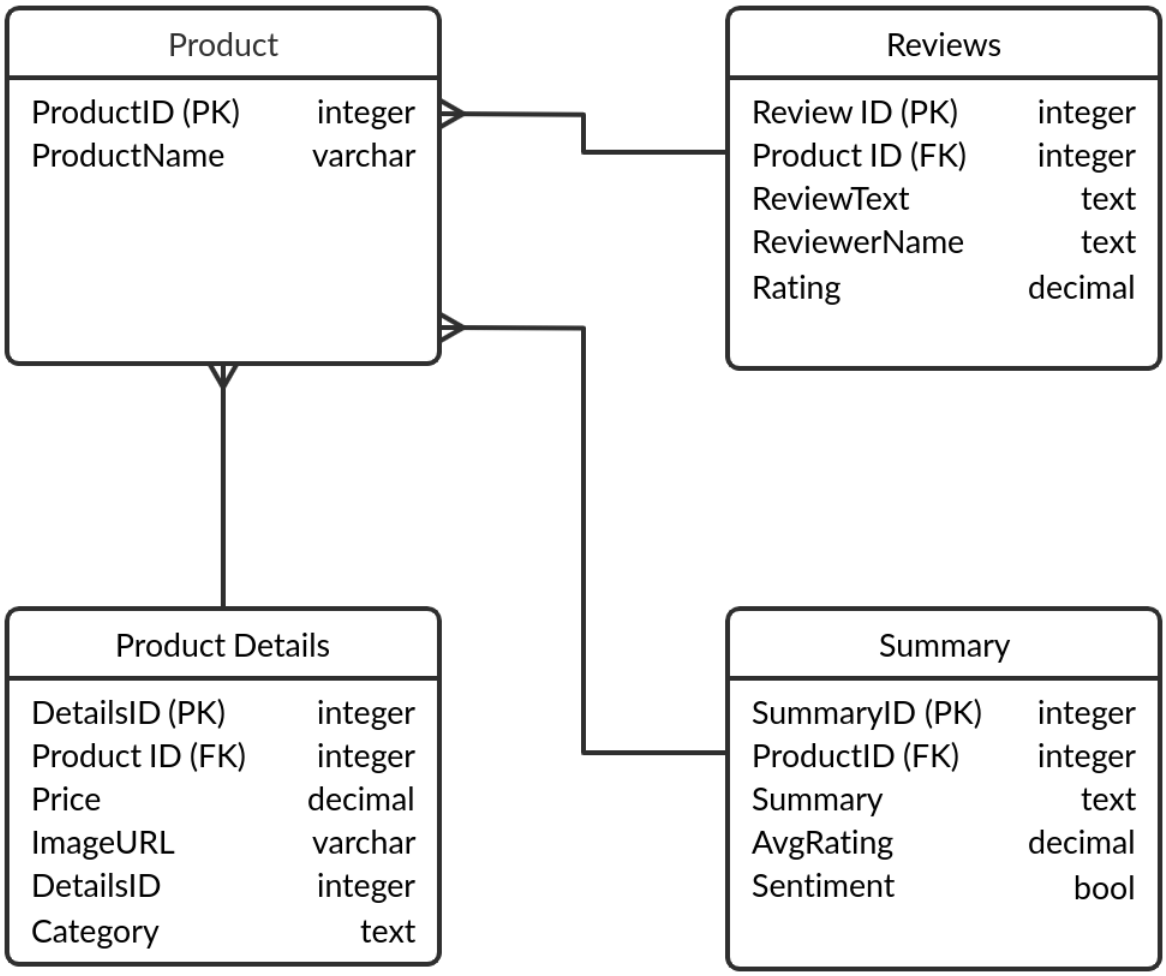

Figure 4.2: Database Design

4.2.2 Use Case Diagram

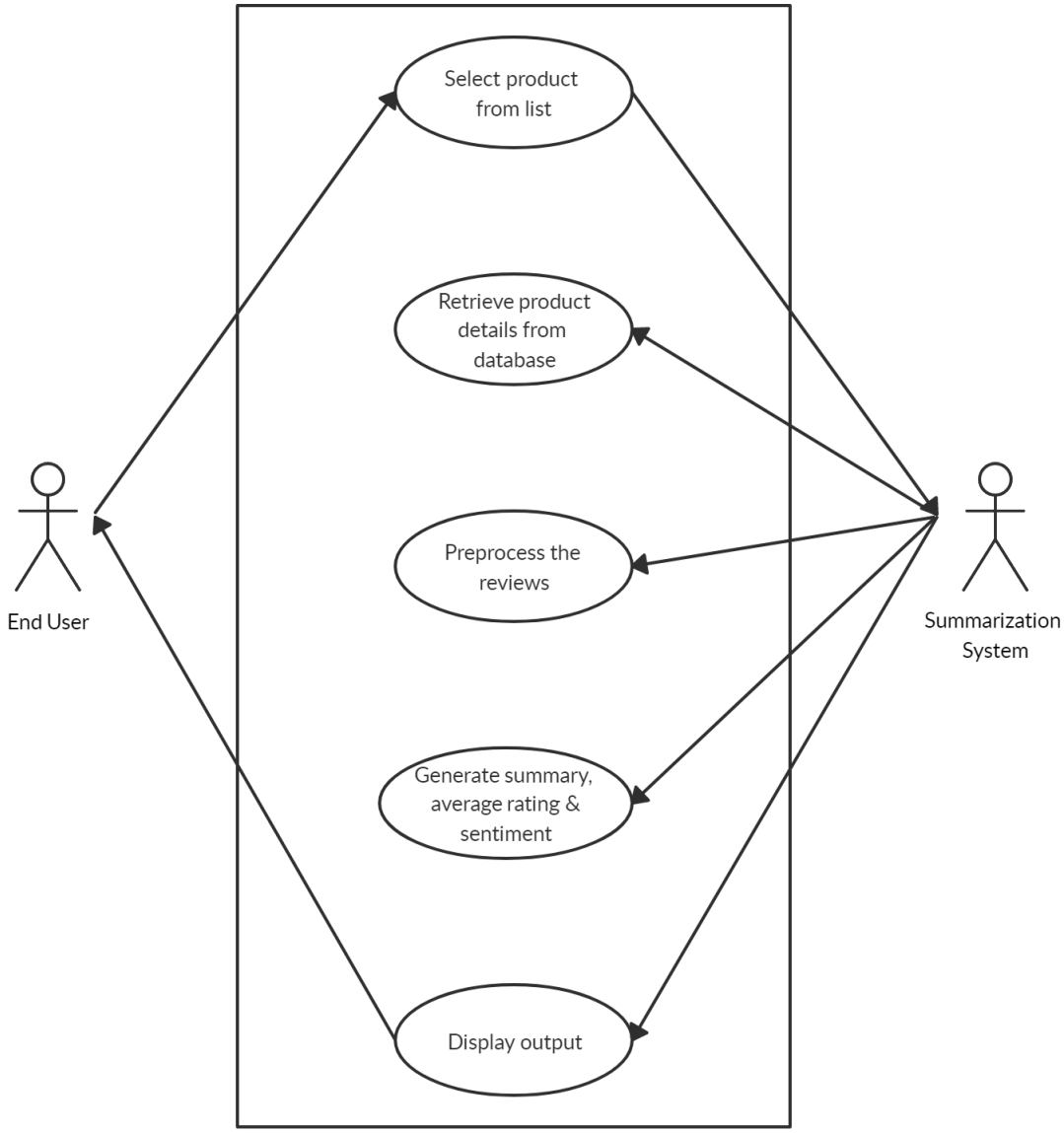

Figure 4.3: Usecase Diagram

4.2.3 Class Diagram

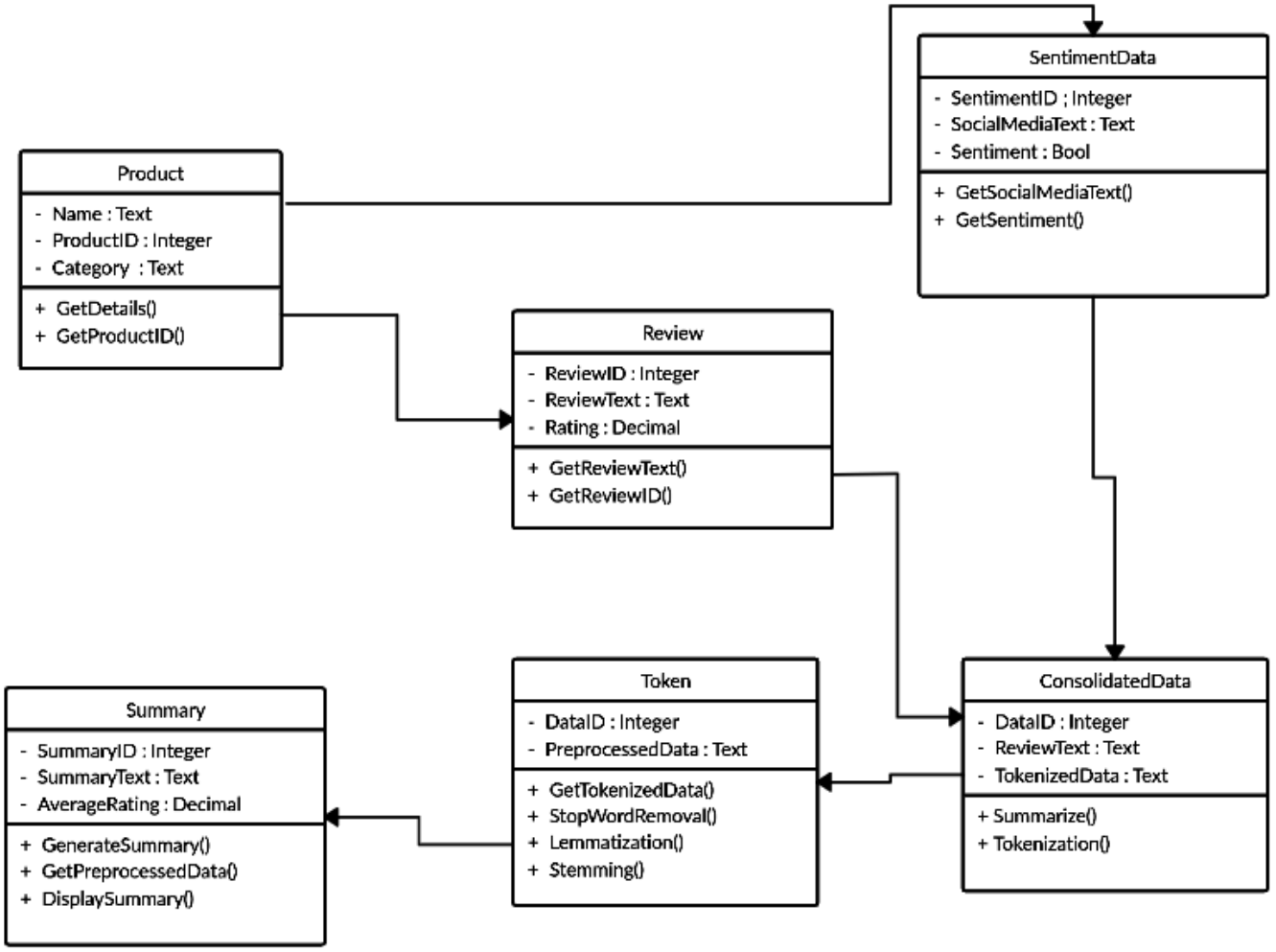

Figure 4.4: Class Diagram

4.2.4 Activity Diagram

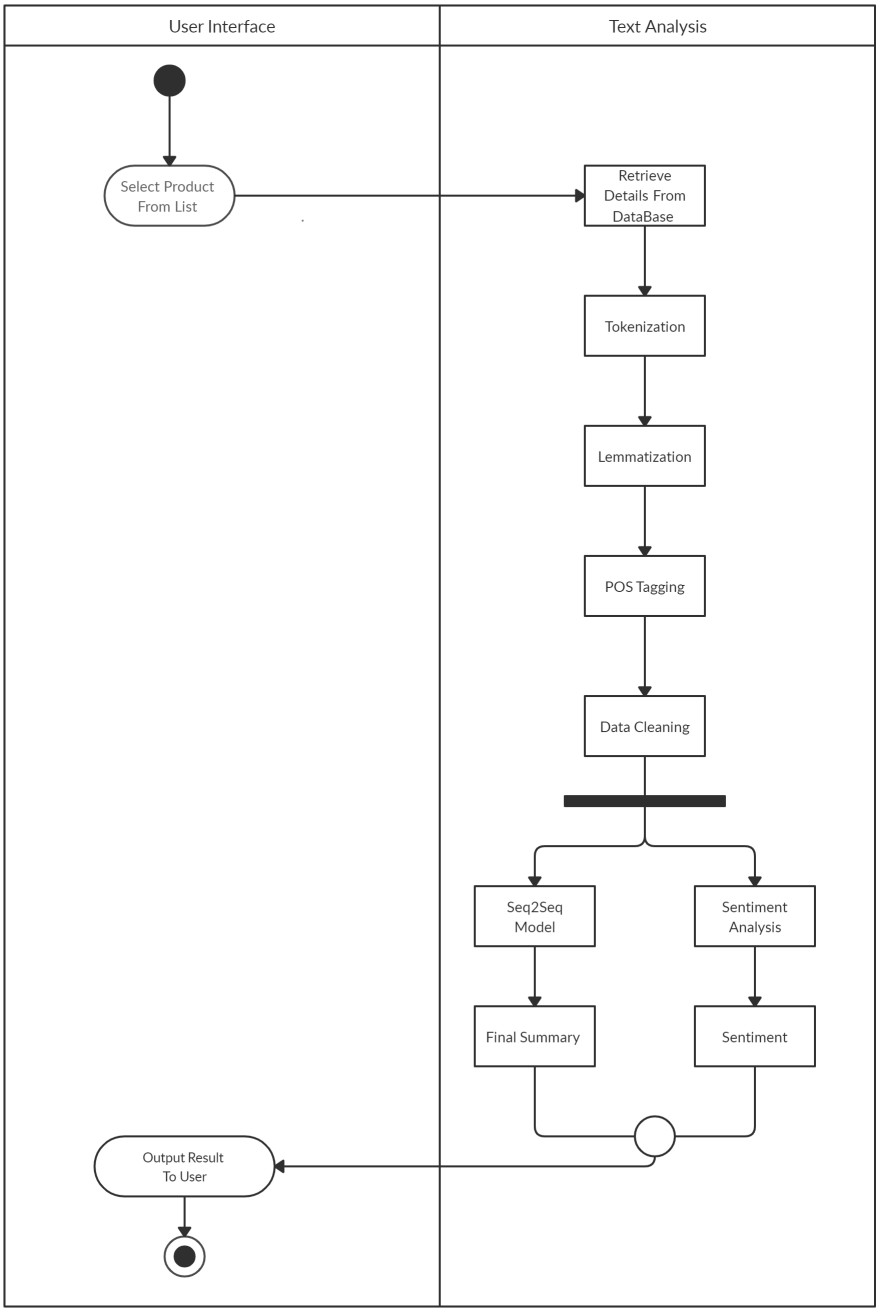

Figure 4.5: Activity Diagram

4.2.5 Dataset Design

For training the summary generation machine learning model, we to use the Amazon product data[20], an open source dataset composed of millions of amazon reviews. This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning the period from May 1996 - July 2014. It includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links to other relevant data (also viewed/also bought graphs).
Twitter is a well-known microblog service that allows public data to be collected via APIs. For training the text sentiment prediction model we make use of the NLTK's twitter corpus currently containing a sample of 20k tweets retrieved from the Twitter Streaming API.

4.3 Libraries and Packages Used

4.3.1 pandas

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It has intelligent data alignment and integrated handling of missing data.

4.3.2 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras has strong multi-GPU & distributed training support.

4.3.3 TensorFlow

TensorFlow is a free and open-source artificial intelligence software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, used for machine learning applications such as to create large-scale neural networks with many layers. It uses data flow graphs to build models. TensorFlow is helpful for Classification, Perception, Understanding, Discovering, Prediction and Creation.

4.3.4 sklearn

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours. It also supports Python numerical and scientific libraries like NumPy and SciPy. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

4.3.5 NLTK

NLTK (Natural Language ToolKit) provides a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning like nltk.sent-tokenization, nltk.word-tokenization, nltk.stopwords etc. When compared to other libraries like spacy, genism, scikit-learn, NLTK has many advantages such as fast tokenization, plenty of approaches to each nlp task, and it supports a large number of languages.

4.3.6 React

React is an open-source Javascript library that is used to build user interfaces. It helps build encapsulated components that manage their own state, then compose them to make complex UIs. React abstracts away the DOM from you, offering a simpler programming model and better performance.

4.3.7 json

Python has a built-in package called json, which can be used to work with JSON data. json exposes an API familiar to users of the standard library marshal and pickle modules. The package contains all the functions that will help one use the JSON files and the allied activities.

4.3.8 pickle

Python pickle module is used for serializing and deserializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it "serializes" the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.). The module is used to save trained models in storable formats.

4.3.9 numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

4.3.10 BeautifulSoup

Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping. It is available for Python 2.7 and Python 3.

4.3.11 sweetalert

SweetAlert is a package in npm used to create beautiful, responsive and customized alerts in web applications. It gives you different options according to the need.

4.4 Module Description

4.4.1 Data Collection

- Amazon product dataset is downloaded for the Seq2Seq summarization model.
- Dataset of positive and negative tweets available in the NLTK library is downloaded for training sentiment analysis model.

4.4.2 Preprocessing

- The dataset is divided into training set and test set.
- Reviews from the training set and test set are preprocessed.
- Tokenization of sentences and words is done with the functions in NLTK library.
- Stop words are removed using the NLTK corpus of stop words.
- Unnecessary punctuation marks are eliminated.
- Abbreviations are resolved by replacing them with their full forms.
- Lemmatization is carried out to obtain the root word with respect to the context.
- The preprocessed training data is then used for training the model and the preprocessed testing data is used for testing the model.

4.4.3 Training and Testing the Models

- The Seq2Seq model with Attention Mechanism is developed by defining its encoder-decoder architecture and the attention layer.
- The model is trained with the preprocessed review data
- The model is tested using the preprocessed testing review data.
- The model is then saved in .h5 format.
- The Naive Bayes model is trained with the preprocessed Twitter sentiment data.
- Preprocessed testing review data is then fed into the model to obtain the sentiments.
- This model is also saved and stored as a pickle file.

4.4.4 Input Module

- User Interface is built using React.
- The input is received through an auto-complete drop-down list.
- The product name is mapped to the product ID and is sent to the back-end of the application.

4.4.5 Summary Generation and Sentiment Analysis

- The product ID is used to retrieve the reviews from the product dataset.
- The reviews are preprocessed and fed into the summarization model to produce summary of each review.
- The preprocessed reviews are fed into sentiment analysis model to obtain the sentiment.
- The most relevant review summaries are returned to the web application along with the sentiment factor.

4.4.6 Output Display

- The summary and sentiment along with the basic details of the product is displayed.
- A sweet alert box is used for the same.

Chapter 5

Data Flow Diagram

5.1 Level 0 DFD

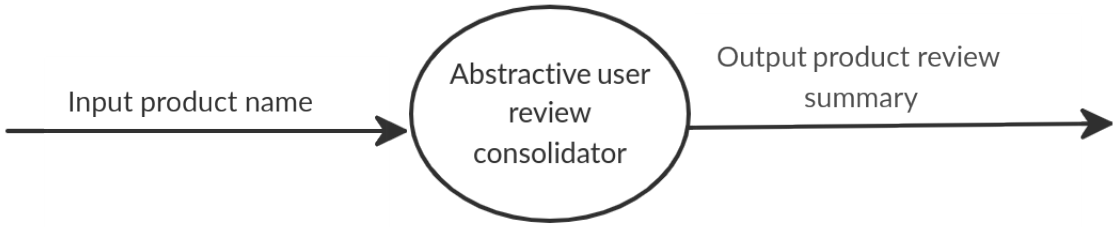

Figure 5.1: DFD Level 0

5.2 Level 1 DFD

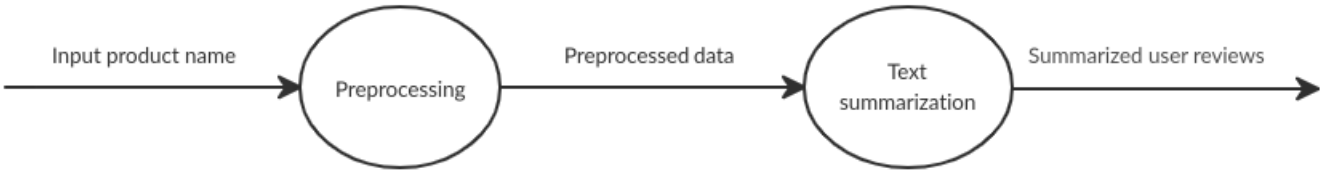

Figure 5.2: DFD Level 1

5.3 Level 2 DFD

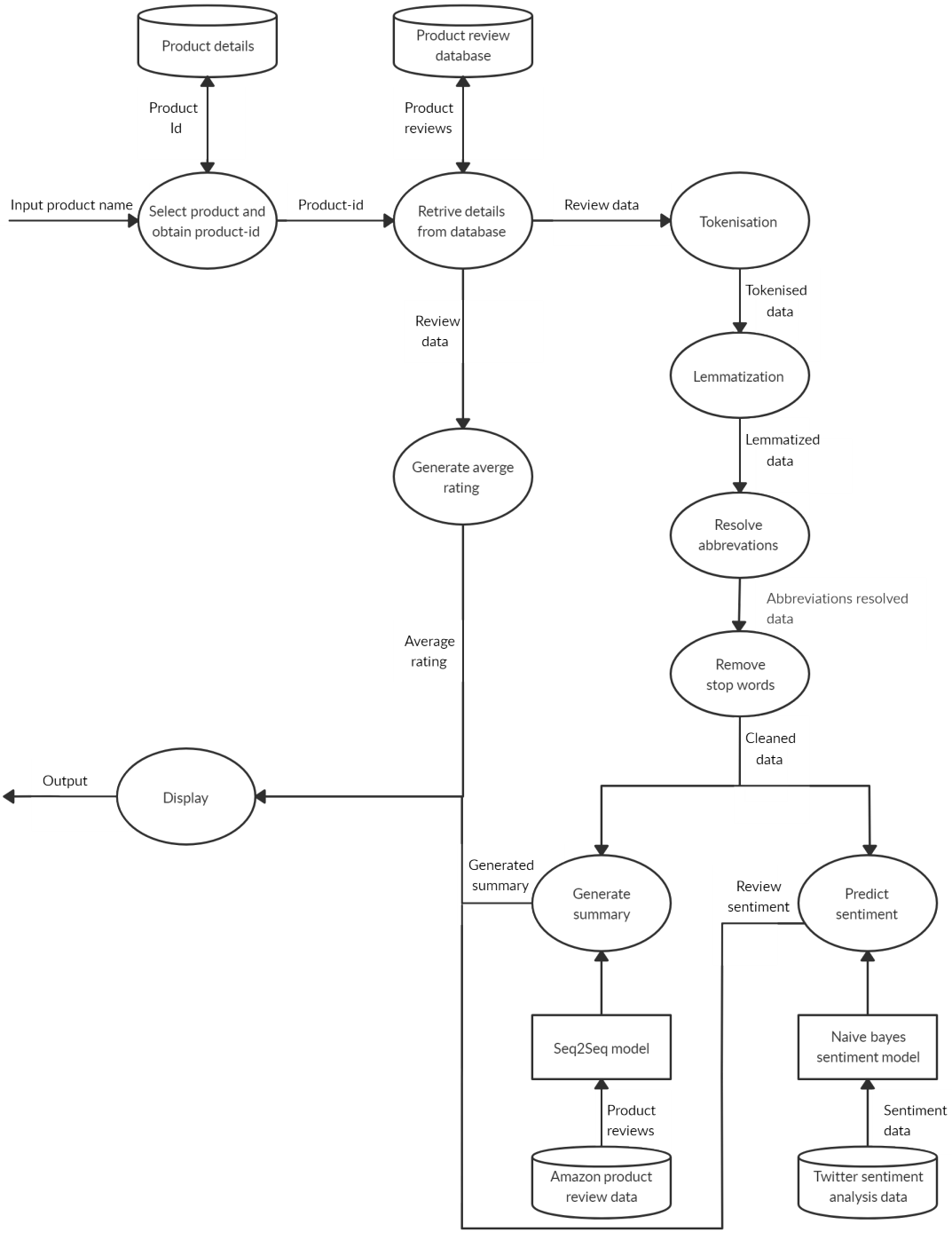

Figure 5.3: DFD Level 2

Chapter 6

Implementation

6.1 Algorithms

6.1.1 Algorithm for Inputting Product Name

Algorithm 1: Algorithm for Inputting Product Name

Input: Product name
Output: Product ID

```
1  Start
2  Accept product name from the drop-down list
3  if user selected x then
4  |  Retrieve the product ID from the database
5  |  Send the product ID to the back-end server
6  else
7  |  Prompt the user to try again
8  |  Go to step 2
9  end
10 Stop
```

6.1.2 Algorithm for Data Collection

Algorithm 2: Algorithm for Data Collection

Input: The product ID
Output: Reviews corresponding to the product ID

```
Start
Initialise pid with product id of the chosen product
Initialise df as product details dataset
Initialise result as an empty dataframe
for every row in df do
    if product-id at the selected row is same as pid then
        Append the row to result
    end
end
Return result
Stop
```

6.1.3 Algorithm for Preprocessing

Algorithm 3: Algorithm for Preprocessing

Input: Consolidated review data
Output: Preprocessed review data

```
Start
Separate the words in the data into tokens
Remove the stop words from the input data
Resolve abbreviations in the data
Perform lemmatization on the data
Stop
```

6.1.4 Algorithm for Training of Seq2Seq Model with Attention Mechanism

Algorithm 4: Algorithm for Training Seq2Seq Model with Attention Mechanism

Input: Product review dataset, maximum review length, maximum summary length
Output: Trained abstractive summarization model

Start
Read dataset into a dataframe
Preprocess the reviews and summaries from the dataframe
Remove rows containing null values from dataframe
Remove rows containing reviews or summaries less than the set value
Split the training and testing data
Training phase:
begin
    Prepare a tokenizer for reviews on training data
    Convert review text sequences into integer sequences
    Pad the review sequences with zeroes upto max review length
    Prepare a tokenizer for summaries on training data
    Convert summary text sequences into integer sequences
    Pad the summary sequences with zeroes upto max summary length
    Append <start> and <end> tokens to each sequence
    Delete rows with only <start> and <end> tokens
    Set up the Encoder
    Set up the Attention Layer
    Set up the Decoder
    Train the model till loss values are minimum
    Save the model
end
Inference phase:
begin
    Create dictionary to convert the index to word for target and source vocabulary using the tokenizer
    for *Each review in testing dataset* **do**
        Encode the entire input sequence
        Initialize the decoder with the internal states of the encoder
        Pass <start> token as an input to the decoder
        Run the decoder for one time-step with the internal states
        Select the word with the maximum probability
        Pass the sampled word as an input to the decoder in the next time-step and update the internal states with the current time-step
        Repeat steps 30-32 until <end> token is generated or maximum length of the target sequence is reached
    end
end
Stop

6.1.5 Algorithm for Naive Bayes Sentiment Analysis Model

Algorithm 5: Algorithm for Naive Bayes Sentiment Analysis Model

Input: Twitter sentiment analysis dataset
Output: Trained naive bayes model

1 Start
2 Read dataset into a dataframe
3 Preprocess the reviews and summaries from the dataframe
4 Remove rows containing null values from dataframe
5 Create an instance of Naive Bayes classifier
6 Train the model with cleaned data
7 Save the model
8 Stop

6.1.6 Algorithm for Summary Generation

Algorithm 6: Algorithm for Summary Generation

Input: Preprocessed review data
Output: Product review summary

1 Start
2 Initialise df as preprocessed product review data
3 Initialise result as empty string array
4 Initialize the decoder with internal states of the encoder
5 Initialise reqlen as length of summary required
6 **for** *every row in df* **do**
7 Initialise summary as empty string
8 **while** $len(result) < reqlength$ **do**
9 Encode the review into vector form
10 Run the decoder for one time-step
11 Calculate the probability for the next word
12 Select the word with maximum probability
13 Pass the sampled word as input to the decoder in the next time-step
14 Update the internal states with the current time-step
15 **end**
16 Append summary to result
17 **end**
18 Let word-frequencies be a dictionary containing frequency of each word in result
19 Score each summary using word-frequencies
20 Select required number of sentences in final summary based on maximum sentence scores
21 Return result
22 Stop

6.1.7 Algorithm for Sentiment Analysis

Algorithm 7: Algorithm for Sentiment Analysis

Input: Preprocessed review data
Output: Product review positive and negative sentiment percentages

Start
Initialise df as preprocessed product review data
Initialise sent-pos with 0 as value
Initialise sent-neg with 0 as value
for *Each row in df* **do**
    Call trained Naive Bayes Model on review at row i
    if *sentiment predicted is positive* **then**
        sent-pos = sent-pos + 1
    else
        sent-neg = sent-neg + 1
    end
end
Return

$$\left(\frac{sent - pos \times 100}{n} \quad and \quad \frac{sent - neg \times 100}{n} \right)$$

Stop

6.2 Development Tools

- **Google Colaboratory**
  Google Colaboratory is a free cloud service by Google and supports free GPU. It provides us with a cloud environment consisting of n1-highmem-2 instance machine type, 2 virtual CPU at 2.2GHz, 13GB RAM and 64GB Free Space. It has an idle cut-off 90 minutes and can be used to run code for free maximum for 12 hours.

- **Visual Studio Code**
  Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring.

- **Git**
  Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Compared to other version control systems, Git is responsive, easy to use, and inexpensive (free, actually). Branching allows you to create independent local branches in your code.

- **Postman**
  Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs faster. It makes it easier for developers to create, share, test and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses.

Chapter 7

Testing

7.1 Testing Methodologies

Software testing methodologies are the various strategies or approaches used to test an application to ensure it behaves and looks as expected. The idea of using various testing methodologies in the development process is to ensure that the software can successfully work in multiple environments and different platforms. Broadly, this is broken down into functional and non-functional testing. In functional testing, the application is tested against business requirements. Non-functional testing methods incorporate all test types focused on the operational aspects of a piece of software. We have performed functional testing part which includes unit testing, integration testing and system testing, each of which is described below.

7.2 Unit Testing

Unit testing refers to the testing of individual software modules or components that make up an application or system. It validates that each module of the software performs as designed.

7.2.1 Input Module

User selects the required product from the drop-down list.By mapping the product name to its corresponding product id, all reviews for the selected product are extracted.

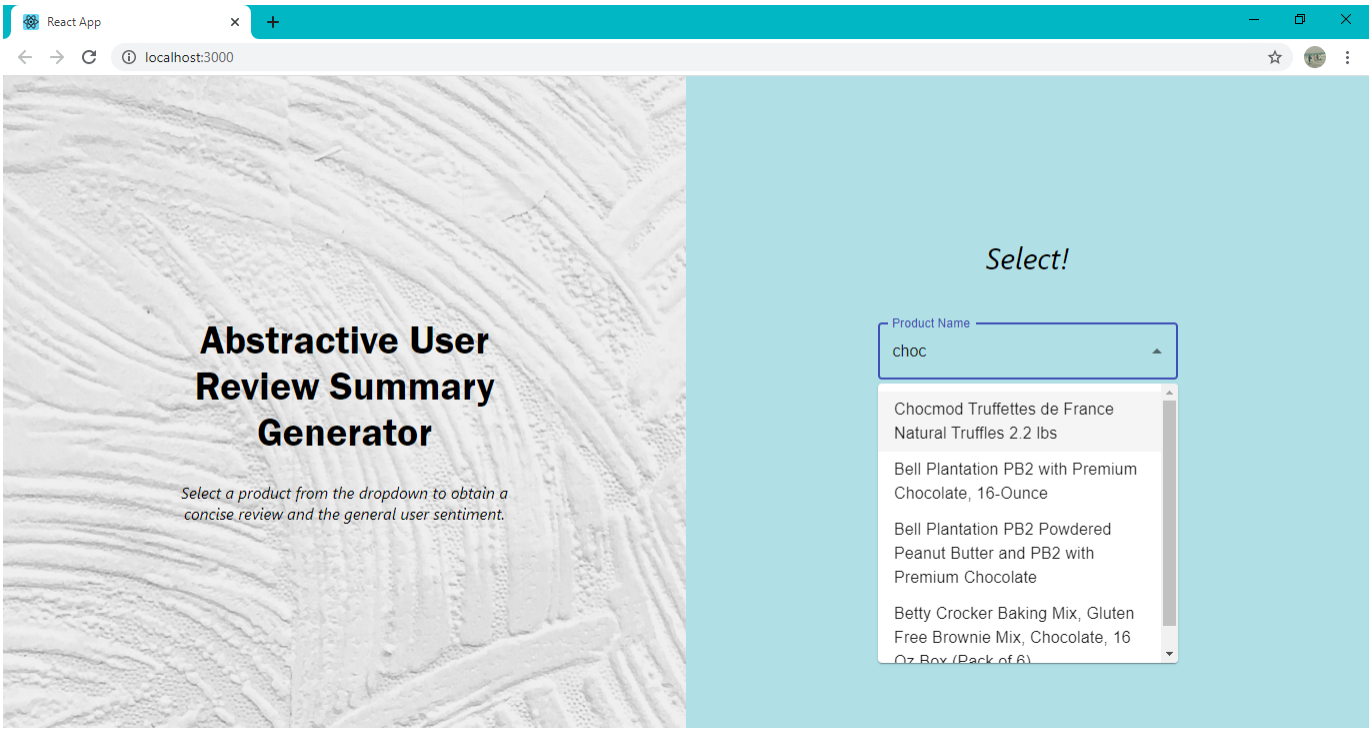

Figure 7.1: Drop-Down Selection

7.2.2 Preprocessing Module

The review data undergoes tokenization, removal of stop words, POS tagging and lemmatization.

```
%%time
import pandas as pd
product_id = "B002QWP89S" #User input

df = pd.read_csv(r"/content/gdrive/My Drive/PROJECT | S7-S8/Data/Reviews.csv", nrows=100000)
df.groupby('ProductId').mean()
df_test = (df.loc[df['ProductId'] == product_id])
print(df_test.columns)

pickle_in = open(r"/content/gdrive/My Drive/PROJECT | S7-S8/Pickle Files/nb_sentiment_analysis_final.pkl","rb")
classifier = pickle.load(pickle_in)
df_test['PreprocessedText'] = df_test.Text.apply(lambda x: remove_noise(word_tokenize(str(x))))
df_test['Sentiment'] = df_test.PreprocessedText.apply(lambda x: classifier.classify(dict([tok, True] for tok in x)))
print(df_test[['PreprocessedText', "Sentiment"]])
```

```
                                                PreprocessedText
 from, these, use, with, caution., bathroom, if, the, cat, get, too, many, she, have, the, run, ..., bathroom, sheltie, do, good, when, i, up, her, ...]
22.37, for, this, 96-pack, and, it, be, by, far, the, best, price, i, have, find, anywhere, i, wish, these, be, part, of, the, subscribe, save, program]
gies, do, n't, i, buy, this, for, my, dashchund, and, minpin, and, it, 's, perfect, a, great, price, for, a, great, product, who, could, ask, for, more]
            [what, can, i, say, dog, love, greenies, they, begg, for, them, all, the, time, they, always, sit, by, the, cupboard, and, ask, for, more]
 lite, for, my, dog, the, package, come, quickly, and, be, package, appropriately, i, be, very, satisfied, with, this, purchase, and, with, the, seller]
                                                                                                                                              ...
 and, they, do, whiten, the, teeth, very, well, see, all, the, great, review, help, your, dog, teeth, if, you, do, n't, want, to, brush, them, like, me]
smart, they, be, 32.99, plus, tax, for, the, exact, same, amount, thanks, for, the, good, buy., bathroom, bathroom, sincerely, bathroom, danny, knowles]
 where, i, get, my, puppy, and, he, love, them, i, like, the, fact, that, they, help, to, clean, his, teeth, as, well, as, satisfy, his, need, to, chew]
, fast, and, convenient., bathroom, i, be, a, huge, fan, of, amazon, they, treat, their, customer, right, and, make, all, purchases/returns, very, easy]
 you, buy, for, a, lot, more, at, other, store, not, much, more, to, say, about, i, love, it, she, love, it, and, i, 've, buy, it, again, numerous, time]
```

Figure 7.2: Preprocessed Text

7.2.3 Model Training

For the sentiment analysis, the in-built sentiment dataset from nltk is used to train and test a Naive Bayes model.

```
[6] positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
    negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)

    positive_dataset = [(tweet_dict, "Positive") for tweet_dict in positive_tokens_for_model]

    negative_dataset = [(tweet_dict, "Negative") for tweet_dict in negative_tokens_for_model]

    dataset = positive_dataset + negative_dataset

    random.shuffle(dataset)

    train_data = dataset[:7000]
    test_data = dataset[7000:]

    classifier = NaiveBayesClassifier.train(train_data)
    print("Score Naive Bayes:", classify.accuracy(classifier, test_data))

    pickle.dump(classifier, open("nb.pkl", 'wb'))
    files.download('nb.pkl')

[(':)', 3691), (':-)', 701), (':d', 658), ('thanks', 391), ('follow', 357), ('love', 337), ('...', 290), ('good', 286), ('get', 263), ('thank', 253)]
Score Naive Bayes: 0.996
CPU times: user 13.1 s, sys: 232 ms, total: 13.4 s
Wall time: 17.2 s
```

Figure 7.3: Naive Bayes Model for Sentiment Analysis

Here, we use a Seq2Seq model with attention mechanism that includes an encoder, decoder and attention layer. Amazon product dataset which contains the product reviews and their summaries are fed into the model to train the model to produce summaries. The model is then tested with a testing dataset of reviews.

```
[ ] model.summary()

WARNING:tensorflow:Layer lstm will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 100, 100)     3149000     input_1[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 100, 300), ( 481200      embedding[0][0]

input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 100, 300), ( 721200      lstm[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 100)    749200      input_2[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 100, 300), ( 721200      lstm_1[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 300),  481200      embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
attention_layer (AttentionLayer ((None, None, 300),  180300      lstm_2[0][0]
                                                                 lstm_3[0][0]
__________________________________________________________________________________________________
concat_layer (Concatenate)      (None, None, 600)    0           lstm_3[0][0]
                                                                 attention_layer[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 7492)   4502692     concat_layer[0][0]
==================================================================================================
Total params: 10,985,992
Trainable params: 10,985,992
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 7.4: Seq2Seq Model with Attention Mechanism Architecture

```
%%time
#compile and train the model
filename = 'model.h5'
model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1,patience=2)
#checkpoint = ModelCheckpoint(filename, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
#history=model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=50,batch_size=128, validation_data=([x_val,y_val[
history=model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=5, callbacks=[es], batch_size=624, validation_data
```

```
Epoch 1/5
529/529 [==============================] - 1479s 3s/step - loss: 1.8264 - accuracy: 0.7355 - val_loss: 1.6098 - val_accuracy: 0.7524
Epoch 2/5
529/529 [==============================] - 1471s 3s/step - loss: 1.5161 - accuracy: 0.7605 - val_loss: 1.4416 - val_accuracy: 0.7665
Epoch 3/5
529/529 [==============================] - 1459s 3s/step - loss: 1.4044 - accuracy: 0.7694 - val_loss: 1.3781 - val_accuracy: 0.7728
Epoch 4/5
529/529 [==============================] - 1451s 3s/step - loss: 1.3415 - accuracy: 0.7742 - val_loss: 1.3263 - val_accuracy: 0.7757
Epoch 5/5
529/529 [==============================] - 1479s 3s/step - loss: 1.2984 - accuracy: 0.7776 - val_loss: 1.2971 - val_accuracy: 0.7782
CPU times: user 2h 23min 17s, sys: 25min 9s, total: 2h 48min 26s
Wall time: 2h 2min 37s
```

Figure 7.5: Model Training

```
[ ] #Testing
    for i in range(0,100):
        print("Review:",seq2text(x_tr[i]))
        print("Original summary:",seq2summary(y_tr[i]))
        print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,max_text_len)))
        print("\n")
```

```
Review: like coffee best many tried amazon normally buy costco found price ordered delivered door amazon nice full bodied flavor
Original summary: good coffee
Predicted summary:  best coffee ever


Review: received march pictured got sweet pops wanted kind would ordered box cherry stawberry orange grape blue razz berry assortment flavors huh disappointed
Original summary: ordered sweet and sour pops got sweet pops
Predicted summary:  not what ordered


Review: really like indian food find kitchens india dishes fairly authentic tasting meals variety pack good range fairly mild spicy provides something everyone
Original summary: great indian sampler set highly recommended
Predicted summary:  very good indian food highly recommended


Review: ever since tried coffee absolutely loved drink cups coffee per day picky coffee drink anything green moutain columbian get costco went away weekend actu
Original summary: dont drink anything else
Predicted summary:  best coffee ever


Review: breakfast cookies favorite young children friend one ounce cookies right size young children individually wrapped used home take along snack item also u
Original summary: good snack for children
Predicted summary:  great for kids
```

Figure 7.6: Summaries Predicted by Trained Model

7.2.4 Summary Generation & Sentiment Analysis

Preprocessed data is fed into the trained seq2seq model and trained naive bayes model. The models generate an abstractive summary and the predict the sentiment respectively, which is then given to the web application. The web application displays the generated summary along with the sentiment and review rating for the user to view.

	pid	title	summaries	avg_rating	sentiment	price	image
0	B001L4JH5I	Pamela's Products Gluten-free Bread Mix, 4-Pound Bags (Pack of 3)	good but not great. great tasting and low carb. great tasting snack. great tasting and healthy. great tasting. great for cats. great for gluten free. great pizza crust. great product. great snack.	4.618497	1	$37.71	https://images-na.ssl-images-amazon.com/images/I/71crjrvBCFL._SL1500_.jpg
1	B000DZFMEQ	Pamela's Products Gluten Free, Bread Mix, 19-Ounce Packages (Pack of 6)	good but not great. great tasting and easy to make. great taste and texture. great tasting and healthy. great for cats. great gluten free cake. great for my cats. great cat food. great taste. gre...	4.627660	1	$28.68	https://images-na.ssl-images-amazon.com/images/I/71aqMwcuqEL._SL1500_.jpg
2	B003QNJYXM	5 Hour Energy Extra Strength Energy Shots, Berry, 12 pk	good but not great. great tasting and easy to make. great tasting and healthy. great tasting snack. great taste and texture. great taste but not so much. great for kids with allergies. great for k...	4.229167	1	$27.89	https://images-na.ssl-images-amazon.com/images/I/71u0bax1-IL._SL1300_.jpg
3	B0039ZOZ86	Gourmet Basics Smart Fries Original KC BBQ, 3-Ounce Bags (Pack of 12)	good but not great. great taste. great for kids and adults. great flavor. great product. great for the price. great taste but not. great gum. great cookies. great for the office	4.141892	1	$32.91	https://images-na.ssl-images-amazon.com/images/I/81fXAtPvYHL._SL1500_.jpg

Figure 7.7: Final Summary, Sentiment and Average Rating

7.3 Integration Testing

Integration testing refers to the testing of the different modules/components that have been successfully unit tested when integrated together to perform specific tasks and activities. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together.

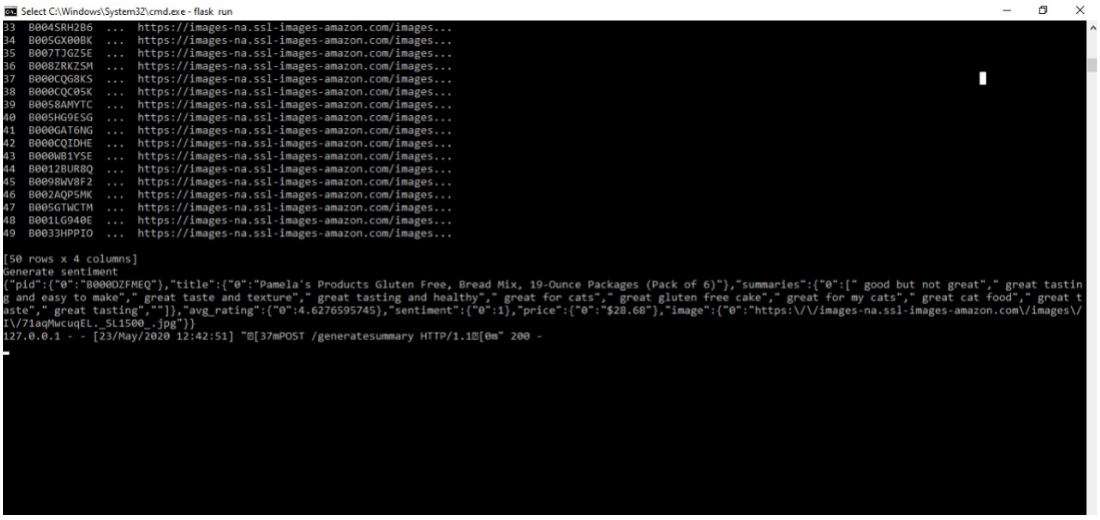

Figure 7.8: Generated Summary

7.4 System Testing

The system testing part of a testing methodology involves testing the complete system for errors and bugs. All modules were integrated at the end of integration testing and the entire system is tested here.

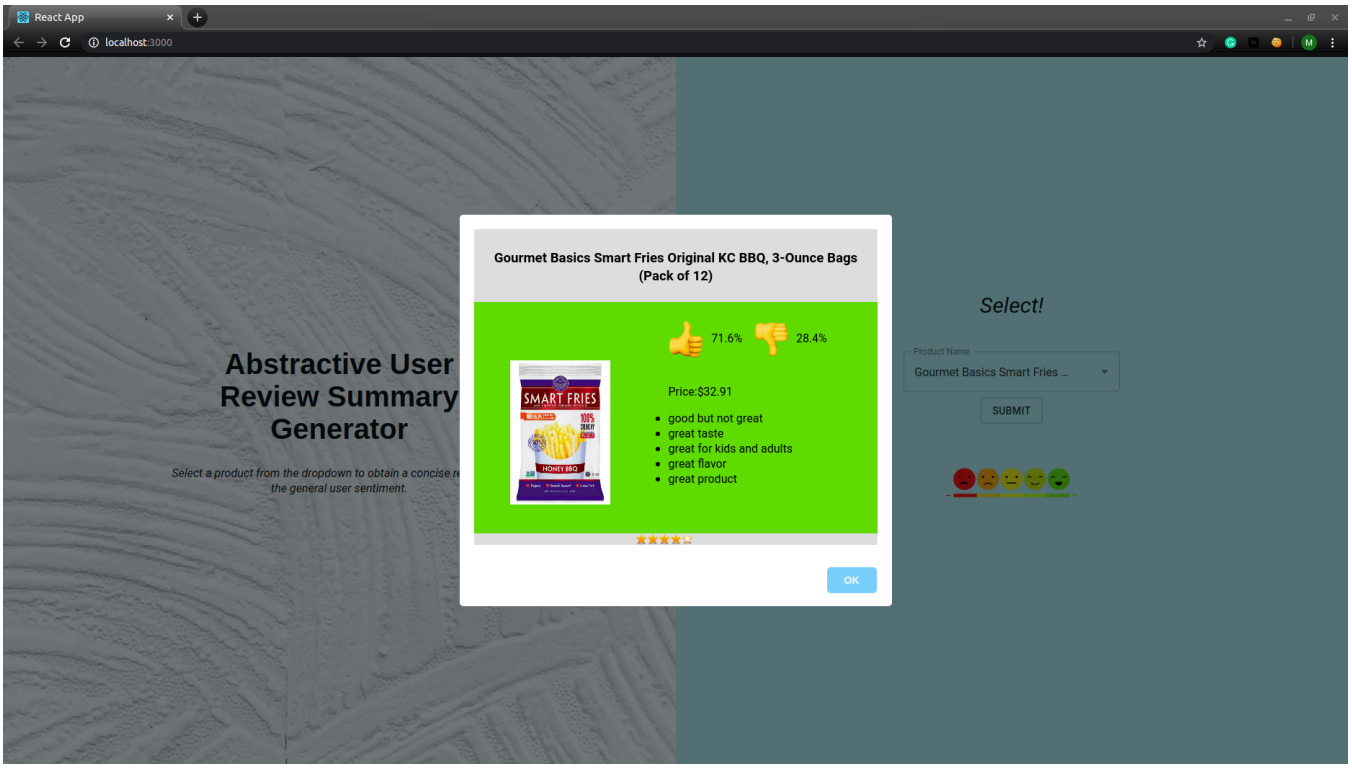

Figure 7.9: Abstractive Summarization System

7.5 Validation Testing

The system is compared to Text Compactor (https://www.textcompactor.com/) which is a free online automatic text summarization tool. The tool has an input box to enter the text. It provides an option to specify the percentage of the text to be retained within the summary.

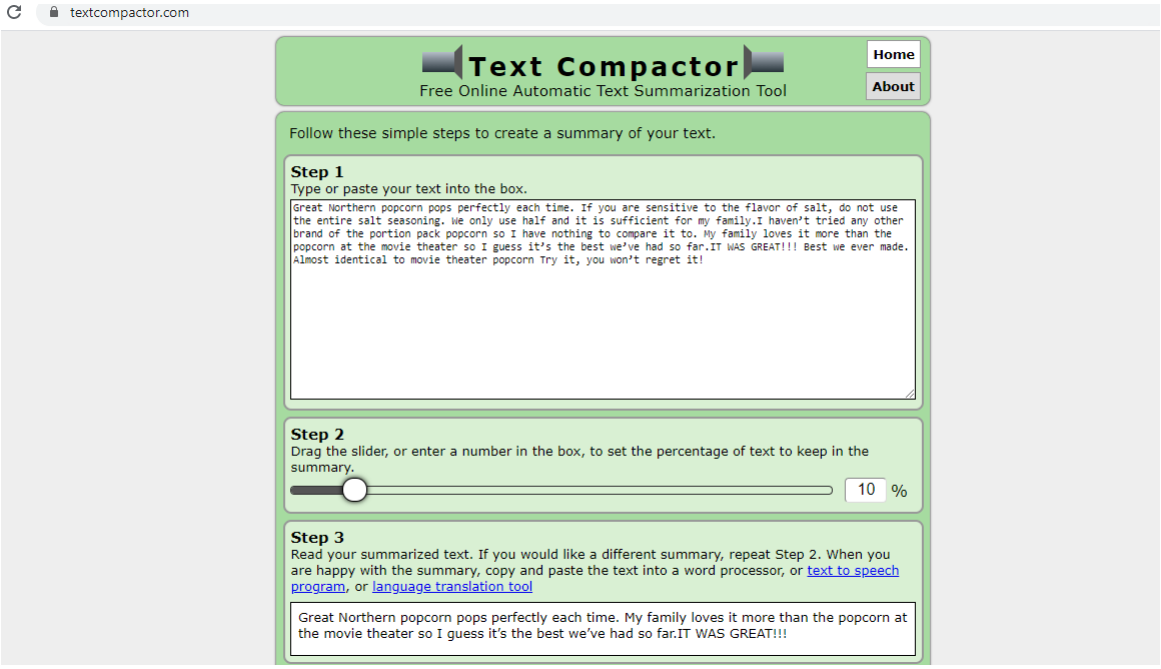

Figure 7.10: Summarization using Text Compactor

In comparison to the Text Compactor which uses extractive summarization, the developed system uses abstractive summarization, thereby producing new sentences. These sentences contain the gist of the whole of the content, represented as points, in contrast to limiting the content by a percentage.The software used for testing only provides selected sentences from the input text based on the frequency of the repeated words, without knowing the relevance of the same in the given text.

The developed system understands the context of the text and prepares a sentence on its own. It also provides the overall sentiment of the reviews, as a percentage of positive and negative review along with the overall rating.

Chapter 8

Graphical User Interface

8.1 GUI Overview

The user interface is quite simple and straight-forward. The home page gives you the option to select the product from an auto-complete drop-down list. On selecting the product, a sweet alert appears with the summary of the reviews, image, sentiment and rating of the product that was selected.

8.2 Main GUI Components

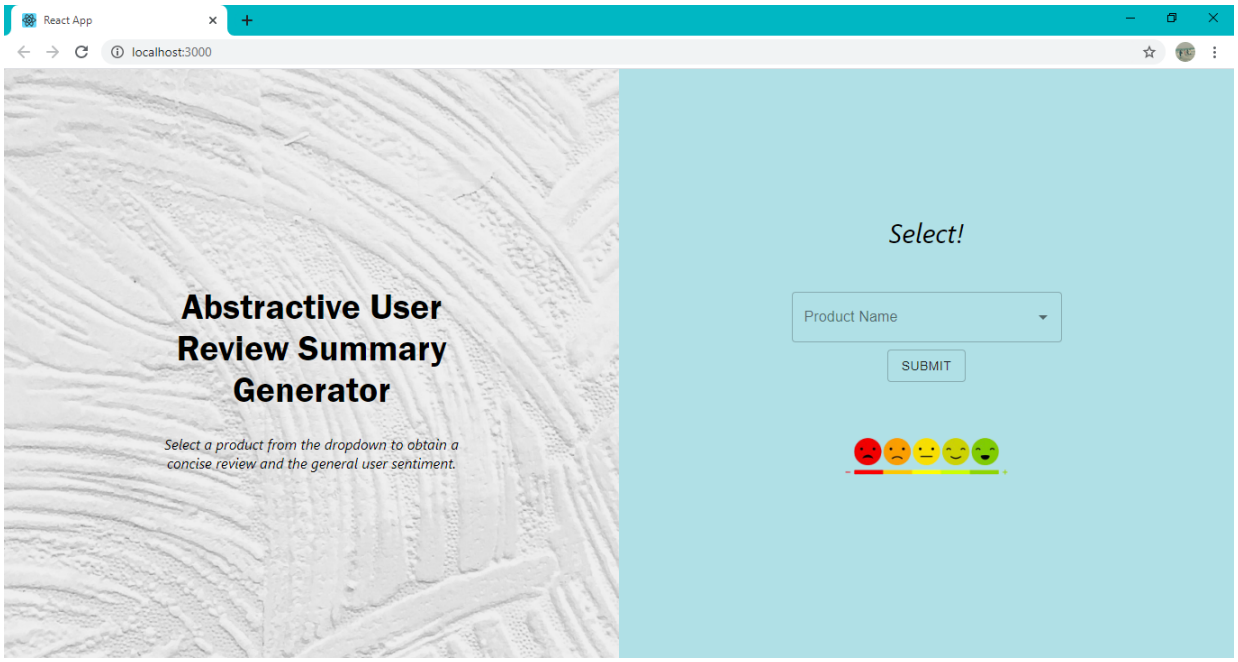

Figure 8.1: GUI Interface

The user interface gives you the options to select the product from the auto complete drop-down list.

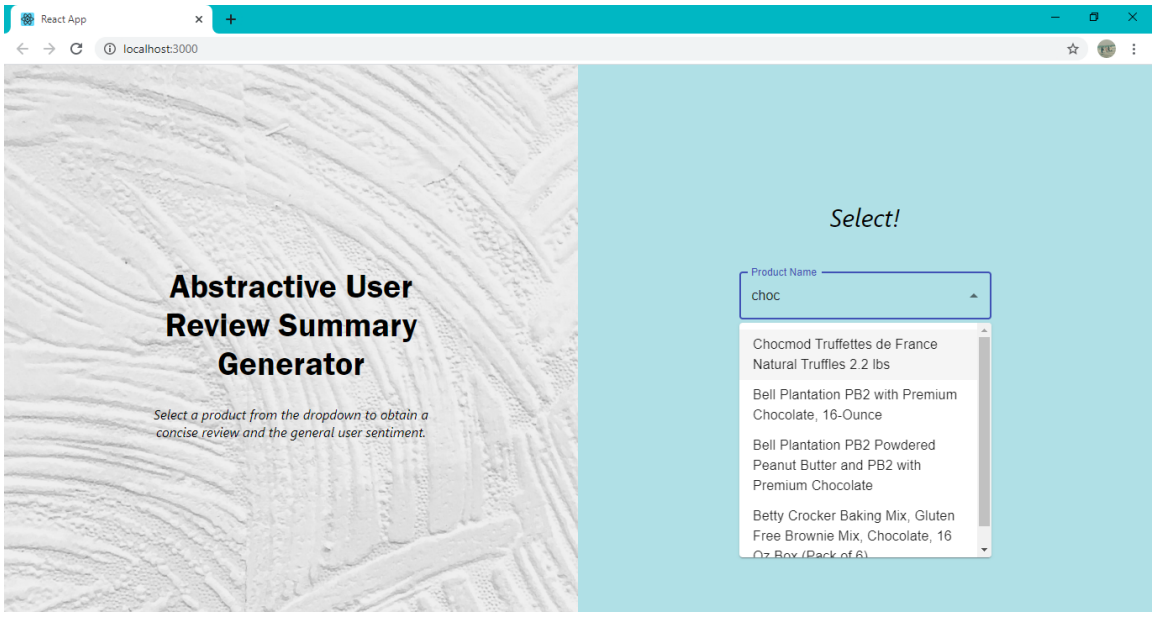

Figure 8.2: Drop-Down Selection

After selecting the product, the summary is generated along with the rating, sentiment and the image of the product in a sweet alert.

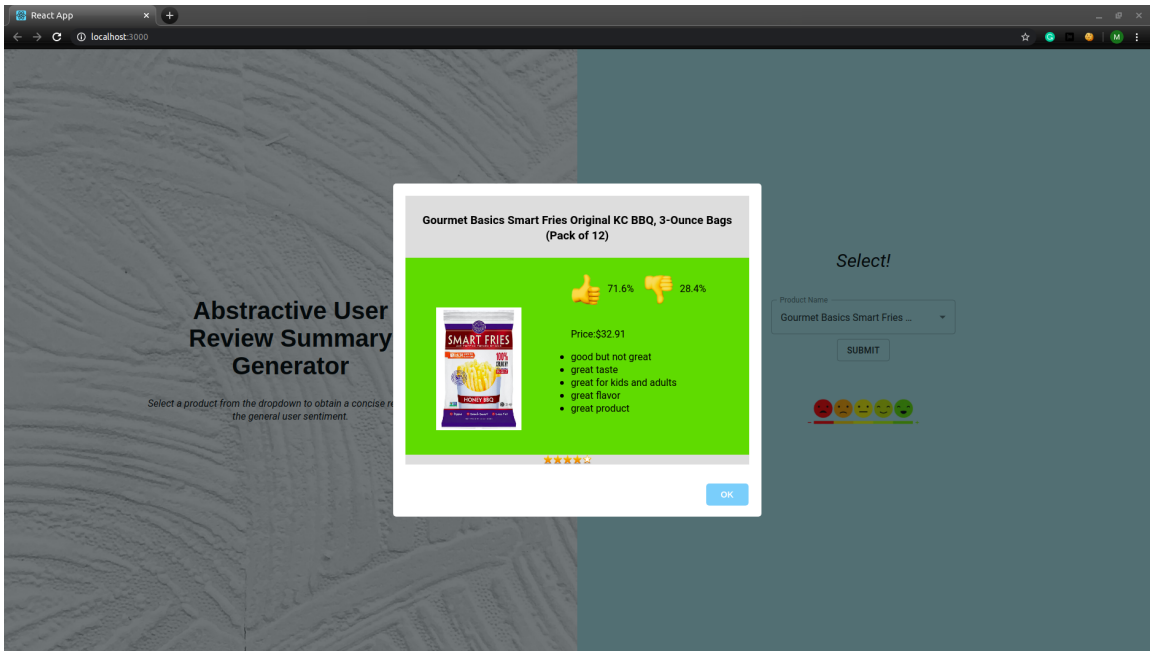

Figure 8.3: SweetAlert

Chapter 9

Results

The web application for the system displays an auto-complete drop-down list in which users can type in the product name. On hitting the submit button, the product id of the selected product is passed onto the back-end of the application.

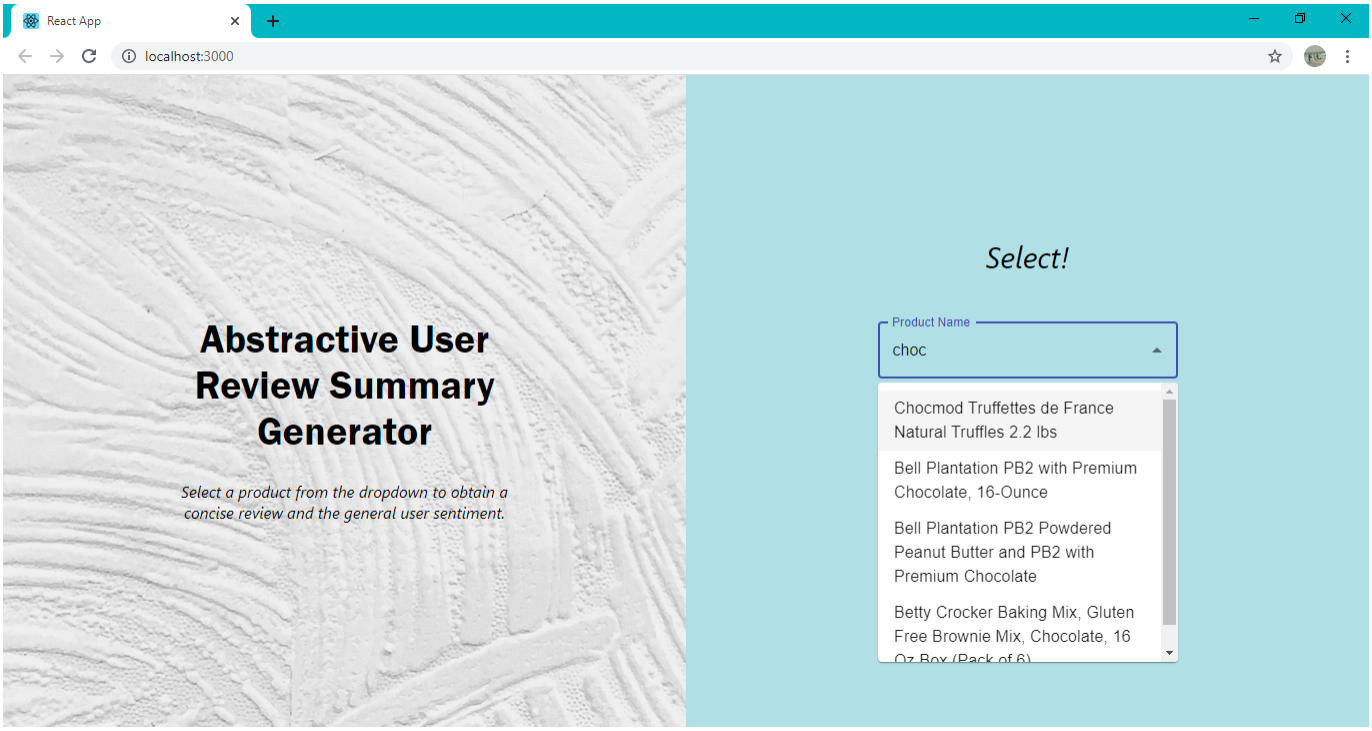

Figure 9.1: Input Selection

At the back-end, the reviews of the product are selected and preprocessed. This is then to the trained Naive Bayes model to generate the sentiment of each review. The percentage of positive and negative reviews is calculated. The reviews are also passed into the trained deep learning model for summary generation. The summaries, along with the sentiment percentage and the average rating is returned to the front-end to be viewed by the user.

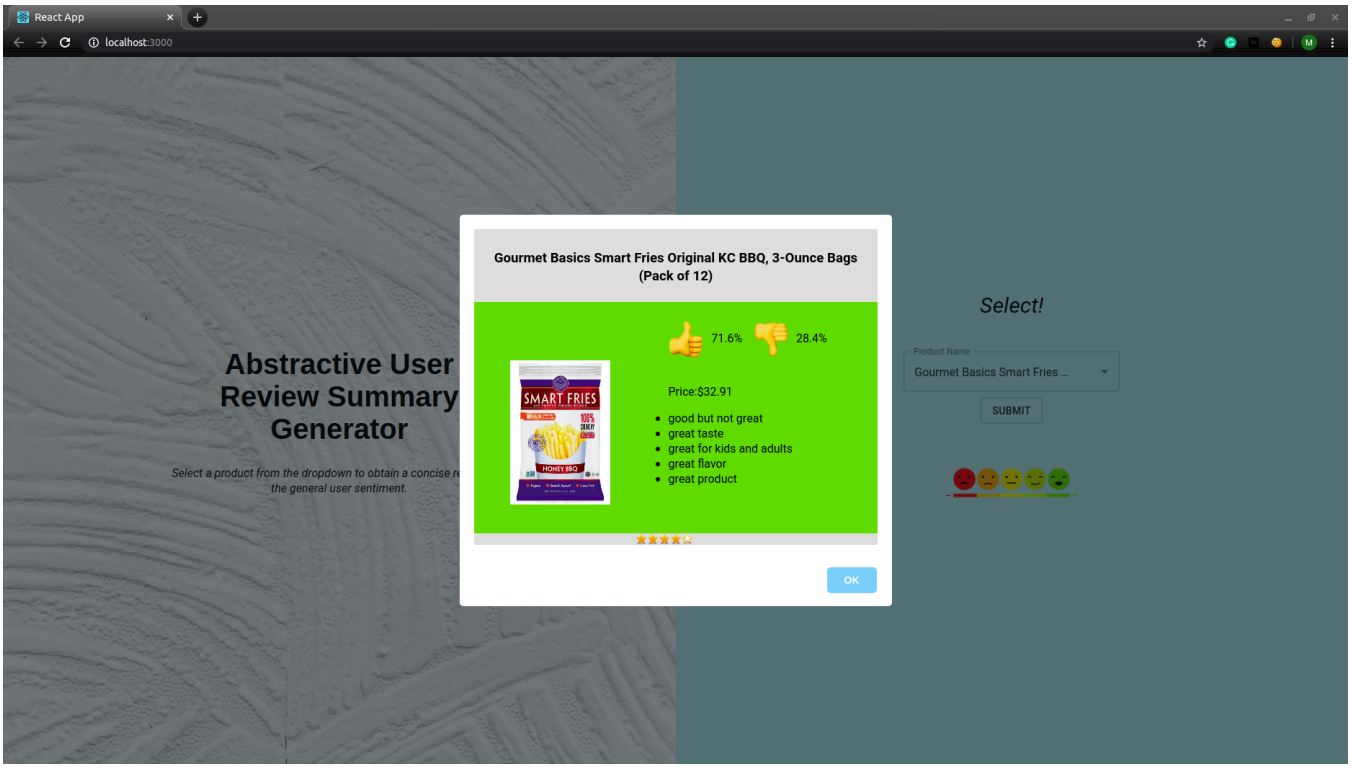

Figure 9.2: Generated Abstractive Summary

The response from the back-end application which includes the summaries, sentiment percentage and the average rating is displayed using a sweet alert box. The alert box features the product name, image and price and the summaries as a bulleted list. The thumbs-up and thumbs-down images indicate the percentage of positive and negative reviews respectively. The filled stars on the bottom represent the average rating out of 5 and the background colour also depicts the range of the product rating.

Chapter 10

Conclusion

A user review consolidation system has been developed using the method of abstractive summarizing to obtain a concise form of the lengthy product reviews available on e-commerce websites along with additional features such as overall rating and sentiment analysis. The system uses a dataset of fine food reviews from E-commerce websites to generate the summaries. It also generates the general sentiment of the customers towards each product. The average rating based on the numerous reviews available for each product is also calculated.

The sentiment analysis has been performed by training a model based on supervised learning. The Naive Bayes classifier has been chosen as the machine learning model, trained on social-media sentiment analysis data, which provides an accuracy of more than 95%. The abstractive summarizer has been modelled as a Sequence-to-Sequence model which uses attention mechanism to improve accuracy. The model gives fairly good summaries of the reviews.

Chapter 11

Future Scope

The model is trained using reviews for a selected dataset. It can be extended to include other categories of items available online, thereby generating better summaries due to a larger dataset. It can also be improved to integrate reviews from various other online shopping websites to provide an unbiased summary and rating.

The generalization capability of a deep learning model enhances with an increase in the training dataset size. Web scraping can be introduced to perform summarization on real-time data.

Implementing Bi-Directional LSTM which is capable of capturing the context from both the directions could result in a better context vector. The beam search strategy can be applied for decoding the test sequence instead of using the greedy approach (argmax). Pointer-generator networks and coverage mechanisms can be implemented in the model to further improve the summary generation capability of the model.

References

[1] Chetana Badgujar, Vimla Jethani and Tushar Ghorpade "Abstractive Summarization using Graph Based Methods" Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018), IEEE 2018

[2] Moye Chen, Lei Li, Wei Liu "A Multi-View Abstractive Summarization Model Jointly Considering Semantics and Sentiment", Proceedings of CCIS2018

[3] Tooba Siddiqui, Jawwad Ahmed Shamsi "Generating Abstractive Summaries Using Sequence to Sequence Attention Model", 2018 International Conference on Frontiers of Information Technology (FIT)

[4] Ranjitha N S, Dr. Jagadish S Kallimani "Abstractive Multi-Document Summarization"

[5] A.Jeyapriya, C.S.Kanimozhi Selvi "Extracting Aspects and Mining Opinions inProduct Reviews using Supervised Learning Algorithm", IEEE Sponsored 2nd International Conference On Electronics And Communication Systems(ICECS 2015)

[6] Wenbo Nie, Wei Zhang, Xinle Li, Yao Yu "An Abstractive Summarizer Based on Improved Pointer-Generator Network"

[7] Kaichun Yao, Libo Zhang , Dawei Du , Tiejian Luo, Lili Tao, and Yanjun Wu "Dual Encoding for Abstractive Text Summarization", IEEE Transactions on Cybernetics

[8] Alshaina S, Ansamma John, Aneesh G Nath "Multi-document Abstractive Summarization Based on Predicate Argument Structure"

[9] Farshad Kiyoumarsi "Evaluation Of Automatic Text Summarizations Based On Human Summaries", 2nd Global Conference on Linguistics and Foreign Language Teaching LINELT-2014

[10] Atif Khan, Naomie Salim, Haleem Farman "Clustered Genetic Semantic Graph Approach for Multi-document Abstractive Summarization"

[11] Shuai Wang, Xiang Zhao, Bo Li, Bin Ge, Daquan Tang "Integrating Extractive and Abstractive Models for Long Text Summarization", 2017 IEEE 6th International Congress on Big Data

[12] Siddhartha Banerjee, Prasenjit Mitra, Kazunari Sugiyama "Multi Document Abstractive summarization using ILP Based Multi Sentence Compression"

[13] Alexander M. Rush, Sumit Chopra, Jason Weston "A Neural Attention Model for Abstractive Sentence Summarization", Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-2015)

[14] Jianwei Niu, Huan Chen, Qingjuan Zhao, Limin Sun†, Mohammed Atiquzzaman "Multi-Document Abstractive Summarization using Chunk-graph and Recurrent Neural Network", IEEE ICC 2017 SAC Symposium Big Data Networking Track

[15] Hao Xu, Yanan Cao, Ruipeng Jia, Yanbing Liu, Jianlong Tan "Sequence Generative Adversarial Network for Long Text Summarization", 2018 IEEE 30th International Conference on Tools with Artificial Intelligence

[16] Harsha Dave, Shree Jaswal "Multiple Text Document Summarization System using Hybrid Summarization Technique", 2015 1st International Conference on Next Generation Computing Technologies

[17] Jagbir Kaur, Meenakshi Bansal "Multi-Layered Sentiment Analytical Model for Product Review Mining", 2016 Fourth International Conference on Parallel, Distributed and Grid Computing

[18] Allahyari, Mehdi, et al. "Text Summarization Techniques: A Brief Survey." arXiv preprint arXiv:1707.02268 2017.

[19] Yao, Jin-ge, Xiaojun Wan, and Jianguo Xiao. "Recent advances in document summarization" Knowledge and Information Systems: 140 2017.

[20] https://www.kaggle.com/snap/amazon-fine-food-reviews/home

[21] Khan, Atif, and Naomie Salim. "A review on abstractive summarization methods."Journal of Theoretical and Applied Information Technology 59.1: 64-72, 2014

[22] Dalal, Vipul, and Latesh G. Malik. "A survey of extractive and abstractive text summarization techniques." 2013 6th International Conference on Emerging Trends in Engineering and Technology (ICETET), . IEEE, 2013.

[23] Vilca, Gregory Csar Valderrama, and Marco Antonio Sobrevilla Cabezudo. "A Study of Abstractive Summarization Using Semantic Representations and Discourse Level Information." International Conference on Text, Speech, and Dialogue. Springer, Cham 2017.

[24] Moratanch, N., and S. Chitrakala. "A survey on abstractive text summarization" Circuit, International Conference on Power and Computing Technologies(ICCPCT), IEEE 2016.

[25] Shimpikar, Sheetal, and Sharvari Govilkar "A Survey of Text Summarization Techniques for Indian Regional Languages" International Journal of Computer Applications 165.112017

[26] Ganesan, Kavita, ChengXiang Zhai, and Jiawei Han. "Opinosis: a graph based approach to abstractive summarization of highly redundant opinions" Proceedings of the 23rd international conference on computational linguistics. Association for Computational Linguistics 2010

[27] Lloret, Elena and Manuel Palomar. "Analyzing the use of word graphs for abstractive text summarization" Proceedings of the First International Conference on Advances in Information Mining and Management, Barcelona, Spain 2011.

[28] Kumar, Niraj, Kannan Srinathan and Vasudeva Varma. "A knowledge induced graph-theoretical model for extract and abstract single document summarization" International Conference on Intelligent Text Processing and Computational Linguistics. Springer, Berlin, Heidelberg 2013.

[29] Subramaniam, Manjula and Vipul Dalal."Test Model for Rich Semantic Graph Representation for Hindi Text using Abstractive Method" 2015.

[30] Liu, Fei, et al. "Toward abstractive summarization using semantic representa-tions."1077,2015

[31] Bhargava, Rupal, Yashvardhan Sharma, and Gargi Sharma. "ATSSI: Abstractive Text Summarization Using Sentiment Infusion." Procedia Computer Science 89,404-411 2016

[32] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralpheischedel. OntoNotes: The 90 solution. In Proceedings of NAACL,2006

[33] Sunitha C, Dr. A Jaya and Amal Ganesh "Abstractive Summarization Techniques in Indian Languages" Peer-review under responsibility of the Organizing Committee of ICRTCSE 2016 doi: 10.1016/j.procs.2016.05.121, International Conference of recent trends in computer science., 2016