Spring 5-20-2019

SENTIMENT ANALYSIS FOR SEARCH ENGINE

Saravana Gunaseelan
San Jose State University

SENTIMENT ANALYSIS FOR SEARCH ENGINE

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Saravana Gunaseelan

May 2019

The Designated Project Committee Approves the Project Titled

SENTIMENT ANALYSIS FOR SEARCH ENGINE

by

Saravana Gunaseelan

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2019

ABSTRACT

SENTIMENT ANALYSIS FOR SEARCH ENGINE

By Saravana Gunaseelan

The chief purpose of this study is to detect and eliminate the sentiment bias in a search engine. Sentiment bias means a bias induced in the search results based on the sentiment of the user's search query. As people increasing depend on search engines for information, it is important to understand the quality of results produced by the search engines. This study does not try to build a search engine but leverage the existing search engines to provide better results to the user. In this study, only the queries that have high sentiment polarity are analyzed and the machine learning models are used to predict the sentiment polarity of the input query, sentiment polarity of the documents produced by the search engine for the given query and also to change the sentiment polarity of the input query to its opposite sentiment. This project proposes an end-to-end system that eliminates the search engine bias by producing results that align with the query sentiment as well as the opposite sentiment. The system comprising of three models for document level sentiment analysis, aspect level sentiment analysis and sentiment style transfer. The document level sentiment analyzer is an LSTM based model that uses GloVe word embeddings to analyze the sentiment of the documents produced by the search engine. The aspect level sentiment analyzer uses deep memory network with attention and auxiliary memory to analyze the sentiment of each search query. In order to obtain the documents of the opposite polarity, the sentiment of the search query is reversed using the sentiment style transfer model that uses a bi-directional LSTM. The results are analyzed to determine the sentiment bias of the search engine based on the input query. In our experiments, we observed that positive sentiment queries yielded 67% documents with positive sentiment and negative sentiment queries yielded 70% documents with negative sentiment. The proposed system eliminates this bias by providing the users with two sets of result, one with positive sentiment and one with negative sentiment.

ACKNOWLEDGMENTS

First, I would like to thank my project adviser Dr. Ching-seh Wu for his continued support and guidance throughout this project. I would like to extend my gratitude to Dr. Robert Chun and Neraj Bodra for being a part of my project committee and for their indispensable feedback which helped me to complete my project. Last but not the least, I would like to thank my family and friends and for their continuous support and guidance throughout the duration of this project.

TABLE OF CONTENTS

List of Tables..viii
List of Figures..ix

Chapter 1: Introduction...1
1.1 Research Objective...1
1.2 Project Motivation...2
Chapter 2: Literature Review...4
2.1 Sentiment Analysis...7
2.2 Aspect-Level Sentiment Analysis...8
2.3 Document-level sentiment analysis...10
2.4 Sentiment Style Transfer...13
2.5 Web Scraper...14
Chapter 3: Model Components and Evaluation...15
3.1 Web Scraper...15
3.1.1 Implementation...16
3.1.2 Result...18
3.2 Text Summarization...19
3.3.1 Implementation...20
3.3.2 Results...22
3.3 Sentiment Analysis...24
3.3.1 Sentiment...24
3.3.2 Problems...25
3.3.3 Levels of Sentiment Analysis...27
3.3.4 Document-level Sentiment Analysis...28
3.3.4.1 Dataset...28
3.3.4.2 Implementation...29
3.3.4.3 Result...35
3.3.5 Aspect Level Sentiment Analysis...38
3.3.5.1 Dataset...39
3.3.5.2 Implementation...40
3.3.5.3 Result...43
3.4 Sentiment Style Transfer...46
3.4.1 Dataset...46
3.4.2 Implementation...48
3.4.3 Result...51
Chapter 4: End-to-End Model Evaluation...53
4.1 Dataset...53
4.2 Implementation...53
4.3 Result...55

Chapter 5: Conclusion and Future Work 60
References 62
Appendix 65

LIST OF TABLES

Table 1: Statistics for IMDB review dataset..28
Table 2: Comparison of different Document-level Sentiment Analyzer.......................35
Table 3: Statistics for two datasets 40
Table 4: Comparison of different Aspect-level Sentiment Analyzer 45
Table 5: Statistics of two datasets..47
Table 6: Comparison of different Sentiment Style Transfer methods 52
Table 7: Summary of results for individual components..59

LIST OF FIGURES

Figure 1: Various Sentiment Analyzer Approaches [30] 11
Figure 2: Representation of Bag of Words [34] 12
Figure 3: Steps involved in Web Scraper 17
Figure 4: List of Queries 18
Figure 5: List of URL obtained from Search Engine Result 18
Figure 6: Sample document scraped from search engine result 19
Figure 7: Sample document provided as input to web scraper 23
Figure 8: Summarized output from the Text Summarizer 23
Figure 9: Problems in Sentiment Analysis [31] 25
Figure 10: Levels of Sentiment Analysis 27
Figure 11: Review length for Document-level Sentiment Analyzer Dataset 29
Figure 12: Working of Recurrent Neural Network 31
Figure 13: Implementation of Document level sentiment analyzer 34
Figure 14: Loss Graph and Accuracy graph for model with no GloVe word embedding 36
Figure 15: Loss Graph and Accuracy graph for model with GloVe word embedding 37
Figure 16: Implementation structure of Aspect-level Sentiment Analyzer [24] 41
Figure 17: F-score for Restaurant and Laptop with increasing Epochs 44
Figure 18: Implementation steps for Sentiment Style Transfer [25] 48
Figure 19: Classifier Accuracy vs BLEU score for DeleteOnly and DeleteAndRetrieve. 51
Figure 20: Overall project structure 54
Figure 21: Search results and its sentiment for a positive search query 56
Figure 22: Search results and its sentiment for a negative search query 57
Figure 23: Final search result produced by the end-to-end model. 58

Chapter 1:
Introduction

1.1Research Objective

Advancement in technology is progressing in an exponential rate. The world wide web is huge with different formats of information such as text, audio and video recordings, etc. Everyone in this world is trying to be constantly connected to the internet for many number of reasons. One of the primary reasons is to get the latest information of the activities that is going on around them. The activities can be anything from local news, information about famous people around them and for several other reasons. Google, Bing, and Yahoo etc., are a common search engine that consumers use when they search for information or services online. The majority or about 90% of the customers select the search results from first page only and out of that 90% more than 80% select the search results from first three or four links. So, it is very important for these search engines to provide relevant information as their top most results. As there is an abundance in information it is becoming increasing difficult to select the results that suits the users.

For instance, a popular person can be viewed as both good and bad by different sets of people. Some users may want to see only the positive content of a person that they are searching for and the search engine must understand this and provide those search results that have positive views for that person. But due to the excess amount of data available for that person the search engines can provide some results which are not exactly what the user is looking for. This problem can be solved by introducing sentiment analysis to both the search query that the user provides as input to the search engine and the search results that the search engine provides as output.

1.2 Project Motivation

This project aims to provide advancements two areas in specific, one is to provide the search query's sentiment that help the user to see how biased their query is and also try to change the sentiment of the query to provide a complete picture of the topic that the user is searching for. Also, the project aims to provide the exact sentiment of the documents provided as a result by the search engine to make it easy for the users to select or understand the content.

In this project, two different kinds of sentiment analysis model are built to provide solution to the above problem. The first sentiment analyzer model tries to understand the sentiment of the search query provided by the user. To do this, a model has to developed that can identify the main aspect of the sentence and provide the sentiment associated with the main aspect. The second sentiment analyzer model is used to summarize the documents into its most important essence without changing the overall meaning and further provide the sentiment score for each of the generated search results. There are several machine learning algorithms such as regressions, deep neural networks that try to predict the exact sentiment of the sentences. This project tries to explore several of those techniques and to identify the approach for sentiment prediction. Apart from developing two sentiment analysis models, this project aims to explore existing web scrapers to develop the dataset. The web scraper must be capable of identifying the URL's present in the search results produced by the search engines. In turn these URLs must be explored to extract the documents that are related to the topic being searched by the user.

The project aims to develop a sentiment style transfer model that takes a positive or negative sentiment query and produces the opposite sentiment of the search query. This would allow the user to see the complete picture of what their search term's sentiment is around the world rather than showing only the positive or negative side of the query.

Chapter 2: Literature Review

People are completely trusting the search engines for information. Most of the queries for a search engine are on controversial topic which are highly opinionated. These topics include searching about a product, a lifestyle, a celebrity and much more. Though the search engine provides appropriate result for the query that the user is searching for, the user never knows how biased or opinionated is their query. Also, the user must search through the results provided by the search engine to find the results which may also be highly opinionated [1]. A sentiment analysis model can present the user with the sentiment of the search results to help the user in navigating those documents. There are several methods proposed for developing the sentiment analyzer. These methods range from simple machine learning algorithms such as Naive-Bayes to deep neural network such as convoluted neural network or recurrent neural network. Some of the related methods are mentioned in this section.

Sergui et al. proposed a novel approach to exploit the web queries provided by the user to the search engine and detect the sentiment for those queries. The authors state that though there are several literatures and researches related to the sentiment analysis as various topics such as product reviews or book reviews, there is an unexplored field where the opinionated queries are used a lot. One such field is the web queries and their sentiments. They clearly list out four different methods in which they can use the sentiment of the queries to gain more information. First advantage is to generate a search query recommendation system, that will try to recommend the users with additional queries that have the same sentiment as the user has provided or try to provide queries that are opposite to that of the user's query sentiment. Another advantage is to change the wordings provided in the query that could be more meaningful to the topic but at the same time does not change the sentiment of the user query. Some other advantages include the Trend Analysis and Target Advertising that can be capitalized by the search engine to gather more information on their users and provide results that are more appropriate for the user. Some interesting finding provided by [2], include identifying the volume of opinionated queries by developing such as sentiment model for analyzing it. Also, the intent of the user can also be identified, and the system can adapt to better suit the user.

As for the dataset creation, they identified the most opinionated topics that are currently trending in the world. They used Google's trend analyzer software to get topmost highly controversial topics and attached questions to them. For example, if the topic was 'abortion', they added questions as prefix such as why, how, when and added opinions as suffix such as good, bad. This allowed them to create a query dataset containing 1200 queries. These queries where then manually tagged to obtain the sentiments.

In [3], the authors tried to use the queries sentiment to effectively remove the search results that are not valuable to the user. They call this method as genre and sentiment classification. The results are analyzed, and they classify the documents into review and non-review documents. They claim that by removing the non-review documents from the search results the user gets to see only the search results that are important and have a more suitable result. The models used for developing the sentiment analysis model include Support Vector Machine and hybrid and heuristic approach that involves understanding the domain knowledge. They try to use only the title of product under review, the URLs associated with the search result and snippets of the actual reviews. In [4], the paper claims that just using SVM would not be suitable for sentiment analysis but a combination of SVMs using the domain knowledge based on unigrams and lemmatized unigrams provide much more satisfying result for developing a sentiment model. Another interesting use case for including sentiment analyzer is to build reputation that help the user by providing beneficial information. According to [5], this can be achieved by using word vectors and paragraph vectors. Word vectors means converting a sentence into vector form and paragraph vectors means preserving the order of words while converting them to a vector. To achieve this, they used morphological analyzer and named entity recognition to convert the sentences into word vectors. Their model consisted of understanding the sentiment polarity of the web queries and providing the search results that align with the user's sentiment polarity. For this, they analyzed the documents provided by the search engine and selected only the documents that are heavily opinionated. Then, the polarities of those documents were analyzed and only the document that align with user's polarity was displayed to the user.

2.1 Sentiment Analysis

Though, sentiment analysis is a highly researched topic, there is not a single model that has good accuracy as compared to other fields. The primary objective of the sentiment analyzer is simple, to check if a given text has a positive or negative sentiment, if so how much polarity score they exhibit. There are several different domains in natural language processing that developing one model will not answer all satisfy all test cases. Some of the challenges in sentiment analysis is domain specific and is difficult to overcome.

Several machine learning techniques have been used separately or in a hybrid form by combining multiple algorithms to analyze and predict the sentiment of a sentence. One of the very first machine learning techniques were developed by Bo Pang et al. They were among the first to analyze the sentiment of the sentences. They used the IMDB movie database which contains around 2000 reviews after being manually labeled as positive, negative or neutral. They used the Naïve Bayes approach as a baseline machine learning algorithm for analyzing the sentiment [17]. For the preprocessing of data, they used unigrams and bigrams. Unigrams are a collection of individual words and bigrams are a collection of pair of words. For their research, they split the movie reviews into individual word vectors forming a bag of words. The words are individually separated for the unigram approach and in pairs for the bigram approach. Finally, a bag of words each with their frequency count and whether they appeared in a positive or negative sentiment is stored.

For actual training of the model, the training dataset was changed into a bag of words and Naïve Bayes approach was applied. The Naïve Bayes approach assigns probability value for each of the words in the vector space.

Once all the probabilities are determined for each word, their collective probabilities are computed and stored for testing phase. Then during the testing phase, sentences with unknown sentiments are tested on the model trained to predict their sentiment. They achieved an accuracy of 80 percent and some of the pitfalls that contributed to this reduced accuracy is because they did not take the relationship among words into account. There are also different levels of sentiment analysis that one must consider in order to properly classify a model. This project uses two levels of sentiment analysis models.

2.2 Aspect-Level Sentiment Analysis

The project considers only the search queries that are related to an aspect or a topic. For example, "Yoga is good for health". In the above sentence the main entity is yoga and user's sentiment about yoga is its good for health, so it's a positive sentiment. For developing aspect-based sentiment analysis three main methods need to be constructed [10]. Identifying the aspect, classifying the sentence and aggregating the result. In [11], the authors clearly mention the steps required for identifying the aspect and classifying the sentiment of the sentence. The most common method includes creating a bag of words with all the aspects. Then the sentences or search queries can be converted into vectors using word to vector models. This process is called as word embeddings. Some popular word embedding models include GLoVe [13], InferSent by Facebook [14], and CBOW [15]. One of the main problems in aspect level sentiment analysis is the ambiguity involved in identifying the aspect to which the sentiment is related to. To resolve this, a binary classification tournament model can be developed [20]. This model takes two aspects at a time and compares them with sentiment words to verify which aspect better matches to the sentiment. A disadvantage with this method is no new additional aspects can be introduced to the model. Deep neural networks like convolutional neural network and recurrent neural networks along with pre-trained word embedding models produce the best accuracy for developing the aspect level sentiment model [12]. The neural network models develop Long Short-Term Memory (LSTM) network from the word vectors and feed them to the training model consisting of multiple layers with different loss functions to create the classifier. But the problem with this model is, it does not try to identify the sentiment towards the aspect but just the general sentiment [11]. To rectify this a joint LSTM model which considers both aspect detection and sentiment analysis have been proposed. One method involves identifying the sentiment words first, since they are easily identifiable and use grammatical relation between the aspect and sentiment words to identify the actual aspect. In a search query, the assumption is that the user would be talking about only one aspect. Hence, it would be easy to identify this single aspect and the sentiment surrounding it.

2.3 Document-level sentiment analysis

This model is used for classifying the sentiment of the search results. Document level sentiment analysis has its own challenges. Some challenges include finding the relationship among sentences, eliminating the outliers i.e. sentences that are neutral, and semantic connections among words [16]. To overcome these challenges, the authors in [16] developed dependency trees that represent the relationship among words and sentences. Then a machine learning algorithm which is a combination of LSTM and GRNN is used to develop the model. The model proposed is a two-layer LSTM model. The first LSTM layer is fed a vector of words, where each word is present in the document. Then the second layer is fed a vector of sentences and finally a softmax function is used for determining the sentiment of the document. Another simple method proposed in [21] involves pre-processing the document to split the document into individual sentences. Then the sentiment of the individual sentences are calculated. Finally, the overall sentiment of the document is derived by calculating the average of sentiments of all sentences. But there is a one important problem in both the above method. The sentiments for the documents are not always same, they tend to change in the same document. For example, a detailed movie review will provide both the positives and negatives for the movie. For this reason, virtual paragraphs are constructed [21]. These paragraphs represent the change in sentiment and are determined by a threshold formula,

$$SD(n, m) = |SV(m) - SV(n)|$$

Here m is the previous sentence and n is the current sentence. If the sentiment polarity change is above the threshold, a virtual barrier is constructed between them. The same virtual barrier can be used to identify only the portion of the document that is same as the user's query sentiment and the document can be summarized with only that sentiment.

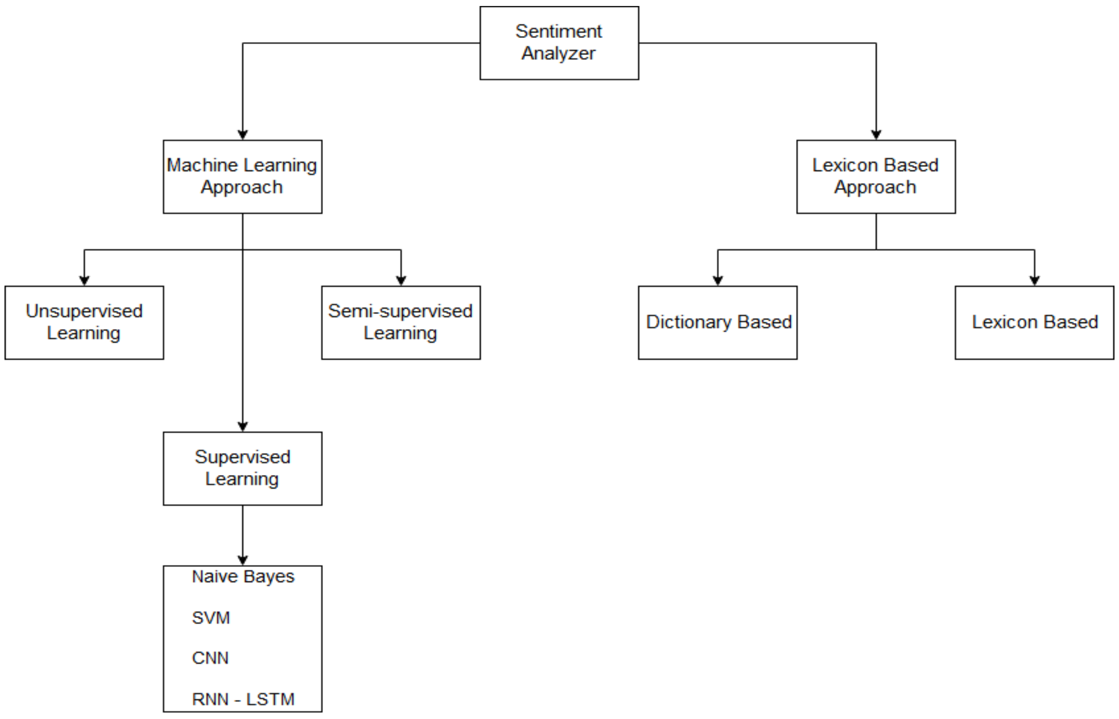
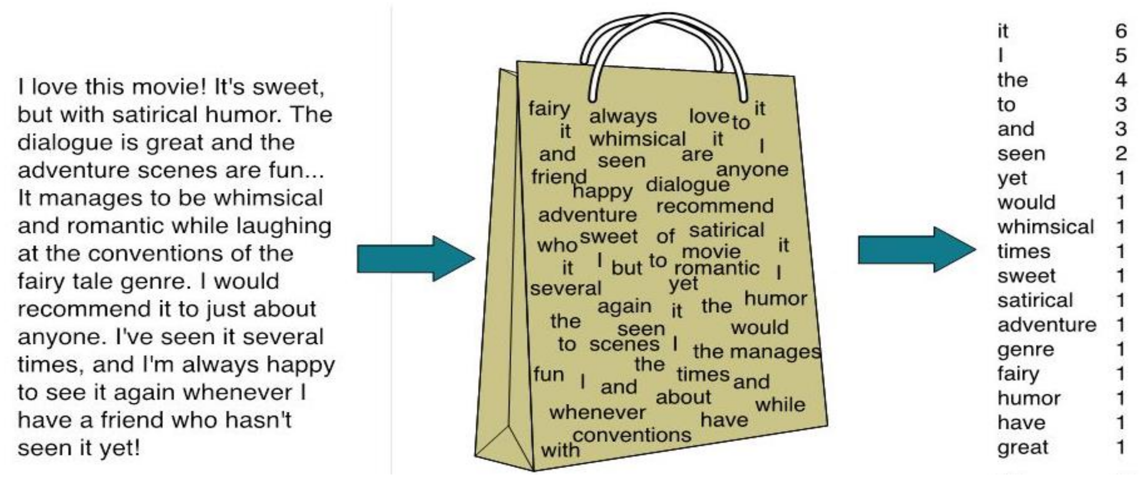


Figure 1: Various Sentiment Analyzer Approaches [30]

There are several approaches to build a document level sentiment analyzer. One of the most common method is to use bag of words. This method is popular because of its simplicity. Even though they are simple, they have the ability to achieve high accuracy. But there is a big downside to the approach implemented by bag-of-words. This approach works by collecting and analyzing each word as an individual element. This makes the approach simple, but they have a hard time analyzing sentences with negation in them. For example, “The book is not great” is intercepted as a positive sentiment because of the presence of word “great”. Since the words are analyzed individually, the word “not” has much less sentiment intensity than the word “great” and leads to misclassification. The approach implemented by Pang. Et al involves using bag of words and naïve Bayes for classification. The dataset used for their experiment is the IMDB movie review dataset. The dataset contained 700 positive reviews and 700 negative reviews. It is very important to have a dataset with equal classification, so that the model developed is not skewed to one particular class.



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1

Figure 2: Representation of Bag of Words [34]

Naïve Bayes works on the assumption that each word present in the class are independent of each other. But as we saw previously, ordering of words is important in determining the sentiment. The first step in their experiment is to divide the reviews into individual words and count them. Then the words are classified based on their individual polarity and their probability is calculated, $P(wk|c)$ = nk+1 n+|$vocabulary$|. Finally, a Naïve Bayes formula is used to combine the probability of each individual words to obtain the final classification. They achieved an accuracy close to 78%.

2.4 Sentiment Style Transfer

For the user to get a complete understanding of the topic, they must be provided with results from both sides of a topic. This means, if the user searches for a positive sentiment then they must be also provided with the negative sentiment for the same topic so that they can get a complete picture. A famous research area for generating such text is called Adversarial text generation. In [8], the authors mention a technique called GANs to generate text that can generate texts with positive or negative meaning. They used a dataset with one billion words to generate semantically correct sentences. Though this project does not require creation of new sentences, one can change the sentiment of the sentences with the model developed in [8]. In [9], the project specifically explains how to change the sentiment of a sentence. The authors have restricted the sentence length to 30 words and have changes the sentiment of a sentence with 80% Accuracy. They used a manifold deep learning model which uses convolutional neural network to change the sentiment of the sentence with minimal changes but retaining the same semantic meaning. Text style transfer is an important NLP field with lot of research. There are various approaches that can be employed for this purpose. Some popular methods include creating complex neural network models using GANs. Generative Adversarial Network consists of an encoder decoder model where the encoder tries to build a new sentence and the decoder tries to find the mistakes in that sentence. In this approach, the output sentences are completely built from scratch. Though this sounds like a promising approach, the problem with this approach other than building the complex GAN model is low-quality of output produced by such models. Another simple approach is to use the same sentences but change only the part which affects the sentiment polarity. For this project, the model developed by Juncen Li et. al is used. This model was chosen because of its simplicity to reproduce and for its better accuracy than other available models.

2.5 Web Scraper

Another field of research that was needed for this project is to build an efficient web scraper. As pointed out by the authors in [6], one must overcome several challenges to build a successful web scraper. Some of the challenges that these authors list out includes collecting the complete document. Sometimes a web document can be spread over various pages. There is also a possibility that the content present in the web pages are loaded dynamically which becomes difficult to mine. Also, some websites try to stop web scraping by employing methods that detect robots, such as CAPTCHAs and reverse Turing tests. The web scrapers must satisfy the legal and ethical requirements. To overcome these challenges several methods are proposed. Popular programming languages provide libraries that perform web scraping. One example is using beautiful soup library in python. Also, there are desktop-based environments that when fed with URLs provide the output in a text or CSV file format [7].

Chapter 3:

Model Components and Evaluation

3.1 Web Scraper

Web scraping, also known as web crawling or web spidering is an algorithm developed to programmatically go over a large collection of web pages, parse them and extract the desired information. It is a process of automatic information and data collection from the internet. This information is stored to the local system in a database and has wide range of applications. Without web scraping one must manually analyze all the web pages and collect and organize the information. This is a very time-consuming process and can be easily automated by Web scraping.

The main application of web scraping is search engines like Google, Bing, Yahoo, and etc... All these major web search engines run web scrapers in the background to provide relevant information to the user based on their search queries. One application of web scraping is news data scraping [22]. There are several news sources that provide articles about the same events. A user can visit all popular news sites and try to obtain those news articles. But if web scraper for such news sources is developed the scraper automatically crawls these web sources and provides all the articles in a single page. Another application is job searching. A user must often visit all the companies that one has interest in and has to constantly check for job positions. This can be an extremely time-consuming task. This can be automated by developing a web scraper that acts as a search engine and scrapes the internet for job positions. Web scraping can automatically collect information about the job vacancies from several websites and provide it to the user [23].

3.1.1 Implementation

The web scraping steps performed for this project include,

1. Collect a list of queries and store them in a csv file.
2. Use these queries in a search engine and fetch the results from it.
3. With a help of a parser, parse the search engine results and extract the URLs
4. Store these URLs in a separate database.
5. Now use an algorithm to access the articles from these URLs
6. Clean the articles thus obtained and store them in a database.

The search engine that was chosen for this project is Google. The first step was to develop the scraping script that can input the search query and retrieve search result from the web. For this, the python library Requests was used. The Requests library acts just like a web browser that can communicate with the web and retrieve the information. Requests library provides several advantages such as passing parameters along with the URL. This helps in creating unique search requests such as searching in particular geographical locations or accessing only news articles, etc... In addition to this, the results retrieved from web are automatically decoded by Requests library by trying out combination of different decoding schemas.

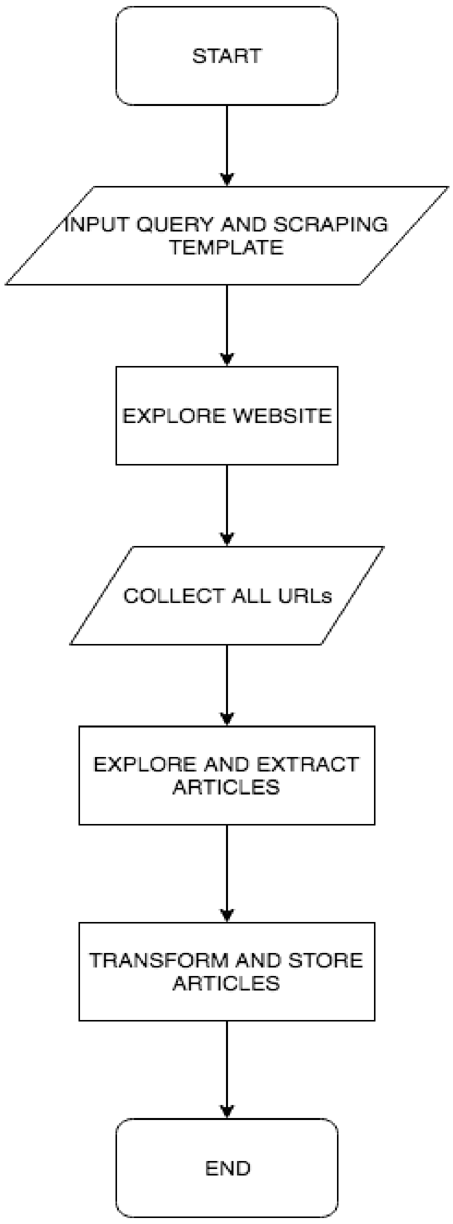
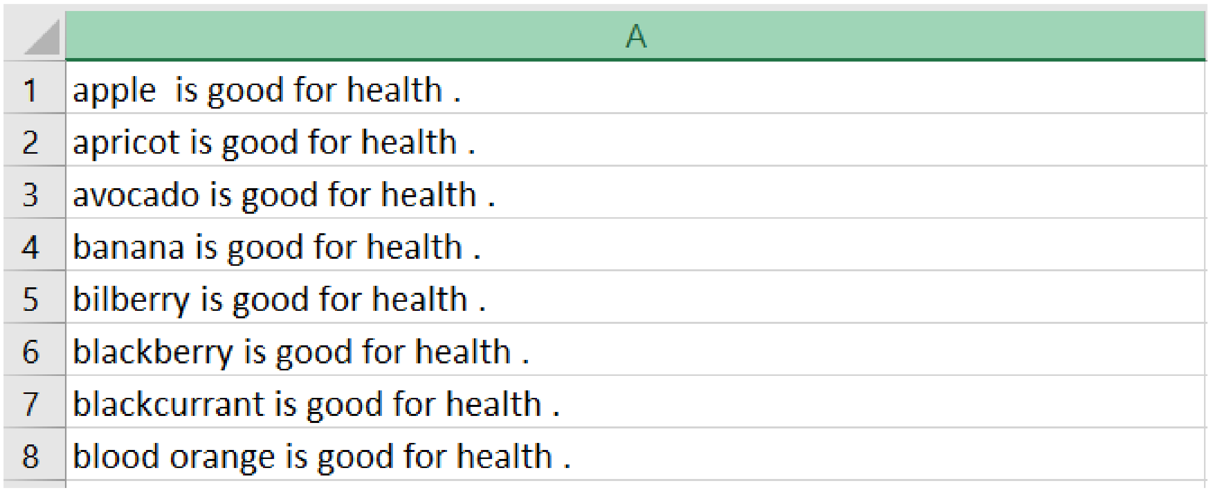

START
INPUT QUERY AND SCRAPING TEMPLATE
EXPLORE WEBSITE
COLLECT ALL URLs
EXPLORE AND EXTRACT ARTICLES
TRANSFORM AND STORE ARTICLES
END

Figure 3: Steps involved in Web Scraper

Once the data is retrieved from the website, pythons nltk library is used to identify the paragraphs that have complete sentences and provide useful meaning. The below diagrams provide the results of web scraping.

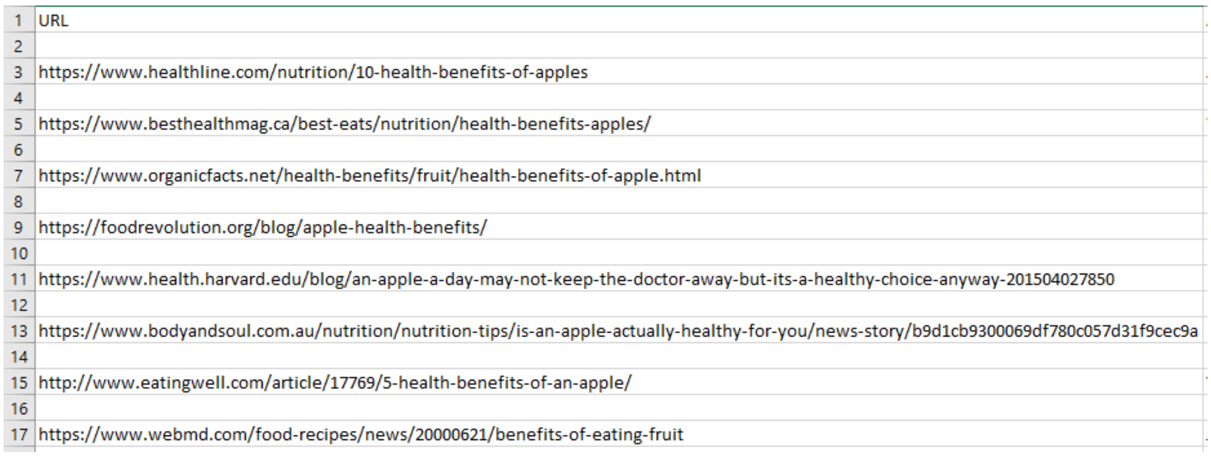
3.1.2 Result

The below figures provide the Queries used and the URL and document extracted.



	A
1	apple is good for health .
2	apricot is good for health .
3	avocado is good for health .
4	banana is good for health .
5	bilberry is good for health .
6	blackberry is good for health .
7	blackcurrant is good for health .
8	blood orange is good for health .

Figure 4: List of Queries



1	URL
2	
3	https://www.healthline.com/nutrition/10-health-benefits-of-apples
4	
5	https://www.besthealthmag.ca/best-eats/nutrition/health-benefits-apples/
6	
7	https://www.organicfacts.net/health-benefits/fruit/health-benefits-of-apple.html
8	
9	https://foodrevolution.org/blog/apple-health-benefits/
10	
11	https://www.health.harvard.edu/blog/an-apple-a-day-may-not-keep-the-doctor-away-but-its-a-healthy-choice-anyway-201504027850
12	
13	https://www.bodyandsoul.com.au/nutrition/nutrition-tips/is-an-apple-actually-healthy-for-you/news-story/b9d1cb9300069df780c057d31f9cec9a
14	
15	http://www.eatingwell.com/article/17769/5-health-benefits-of-an-apple/
16	
17	https://www.webmd.com/food-recipes/news/20000621/benefits-of-eating-fruit

Figure 5: List of URL obtained from Search Engine Result

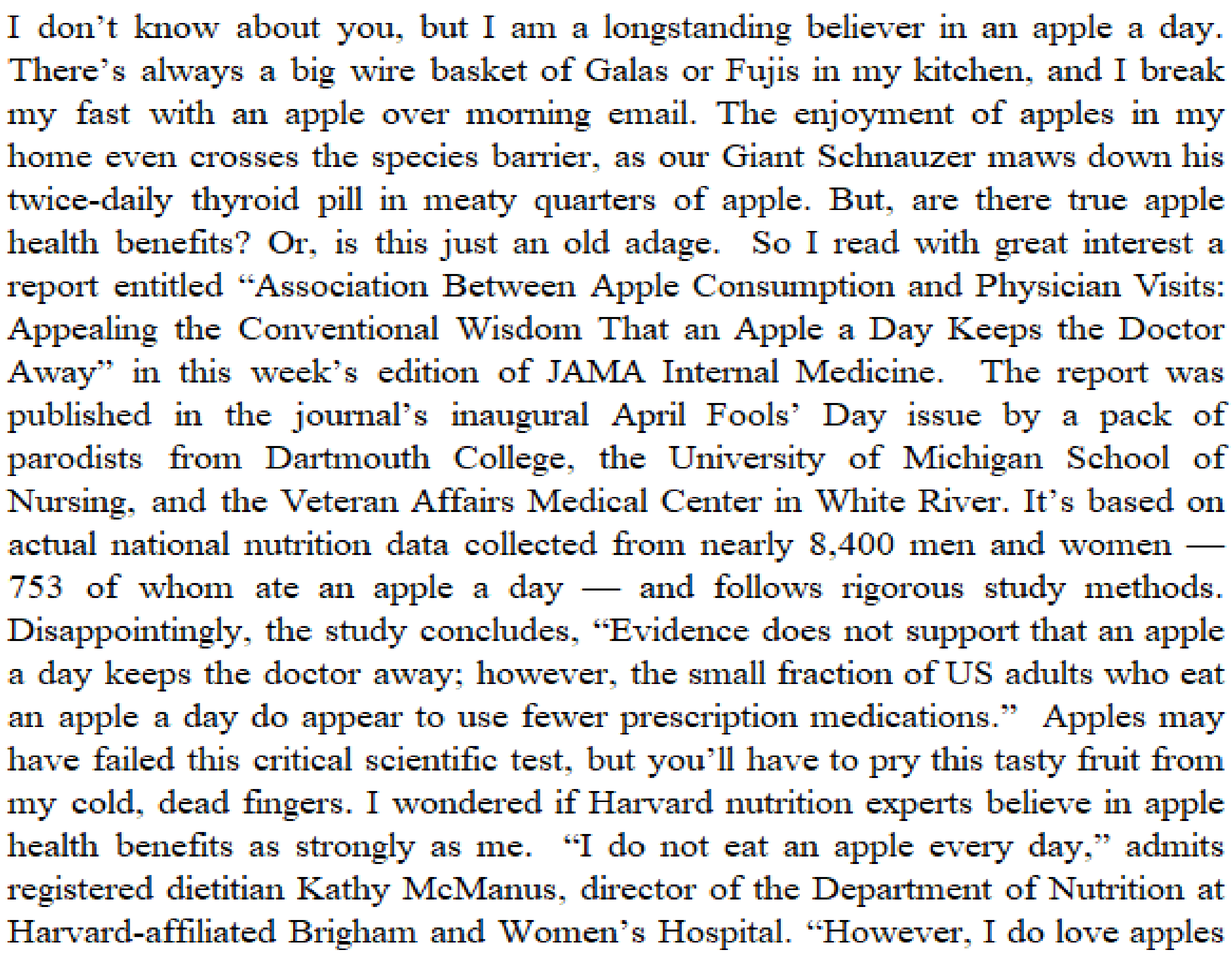

I don't know about you, but I am a longstanding believer in an apple a day. There's always a big wire basket of Galas or Fujis in my kitchen, and I break my fast with an apple over morning email. The enjoyment of apples in my home even crosses the species barrier, as our Giant Schnauzer maws down his twice-daily thyroid pill in meaty quarters of apple. But, are there true apple health benefits? Or, is this just an old adage. So I read with great interest a report entitled "Association Between Apple Consumption and Physician Visits: Appealing the Conventional Wisdom That an Apple a Day Keeps the Doctor Away" in this week's edition of JAMA Internal Medicine. The report was published in the journal's inaugural April Fools' Day issue by a pack of parodists from Dartmouth College, the University of Michigan School of Nursing, and the Veteran Affairs Medical Center in White River. It's based on actual national nutrition data collected from nearly 8,400 men and women — 753 of whom ate an apple a day — and follows rigorous study methods. Disappointingly, the study concludes, "Evidence does not support that an apple a day keeps the doctor away; however, the small fraction of US adults who eat an apple a day do appear to use fewer prescription medications." Apples may have failed this critical scientific test, but you'll have to pry this tasty fruit from my cold, dead fingers. I wondered if Harvard nutrition experts believe in apple health benefits as strongly as me. "I do not eat an apple every day," admits registered dietitian Kathy McManus, director of the Department of Nutrition at Harvard-affiliated Brigham and Women's Hospital. "However, I do love apples

Figure 6: Sample document scraped from search engine result

3.2 Text Summarization

Text Summarization is the process of extracting important content from the original document or documents. There is a huge need for text summarization and the necessity for it has increased many-fold in the recent years due to the explosion of information. In this project text summarization is used for summarizing the documents obtained from web scraper. Web scraper collects these documents from the search engine results generated for the given search query. Text summarization is important to remove unwanted data from the scraped text documents and also to make the input to the sentiment analyzer as close to the input dataset. As the input dataset to the sentiment analyzer has reviews with a average length of 400 characters, the extracted documents are also summarized to this length.

Based on the input and output of the text summarizers they are classified into varies types. Based on the input they are classified as Single document summarization where the input consists of a single document and Multiple document summarization where the input consists of a collection of documents. Also, a query can be passed along with the input documents, this is called Query summarization.

Based on the output summary, they are classified as Indictive summary, where the summary does not give the complete information present in the document but only a partial one, Informative summary, where the summary provides the complete information present in the document, Keyword summary, where only the important keywords present in the document are summarized, and Headline summary, as the name indicates it provides an one line summary.

3.2.1 Implementation

There are basically two important approaches that can be used for text summarization, Extractive and Abstractive.

In extractive summary, important sentences are extracted from the original document and are concatenated to form a single document. The pros of this approach are, it is easy to implement and the output sentences this produced have proper grammatical meaning because they are the same sentences present in the original document. There are several methods that are employed to find the importance of the sentences present in the original document. Features based method extracts the features in a sentence and evaluates its importance. Some of the features include, position and length of the sentence, presence of verb, term frequencies and many more. Text-Rank based method uses a similarity matrix similar to web engines page rank algorithms for ranking the sentences. Some other extractive summary approaches include, Topic based summary and Grammar based summary.

In Abstractive summary, complex machine learning approaches are used to study the documents and provide a summary just like the way a human does. Sometimes the output produced will have sentences that are not present in the original document. These new sentences are created to provide coherency to the output text. The pros of this approach is they can create a more fluent and natural summary of the input text. But it is also much harder to make the model generate coherent phrases and connectors. Some of the most common Abstractive text summarization approach include, Sequence to Sequence model, encoder decoder model and neural attention model.

For this project to extractive summarization approach is used. The text-rank model in combination with input query approach is used for summarizing the documents.

Text-Rank is an extractive and unsupervised text summarization technique which does not need any training data. The flow of the Text-Rank model is given below.

1. The first step is to take the article from the web scraper and split them into sentences.
2. Then they are divided into vectors and the similarities between these vectors are obtained and stored in a similarity graph.
3. The similarity graph is analyzed to include only the vectors sentences with the given aspect term. This would eliminate all the unwanted information present in the input text.
4. The similarity matrix is mapped to a graph with sentences as the vertices and similarity score as the edges.
5. As the last step, only the sentences with higher similarity score is taken and combined to form a single sentence.

3.2.2 Results

One of the articles extracted by the web scraper is given below. The query used to obtain this article is "Apple is bad for health". As one can see, the article contains the information related to the query. But also, there is some comparisons made to other food beverages which is not relevant for this project. The final output from the summarizer only includes sentences that are directly related to the search query.

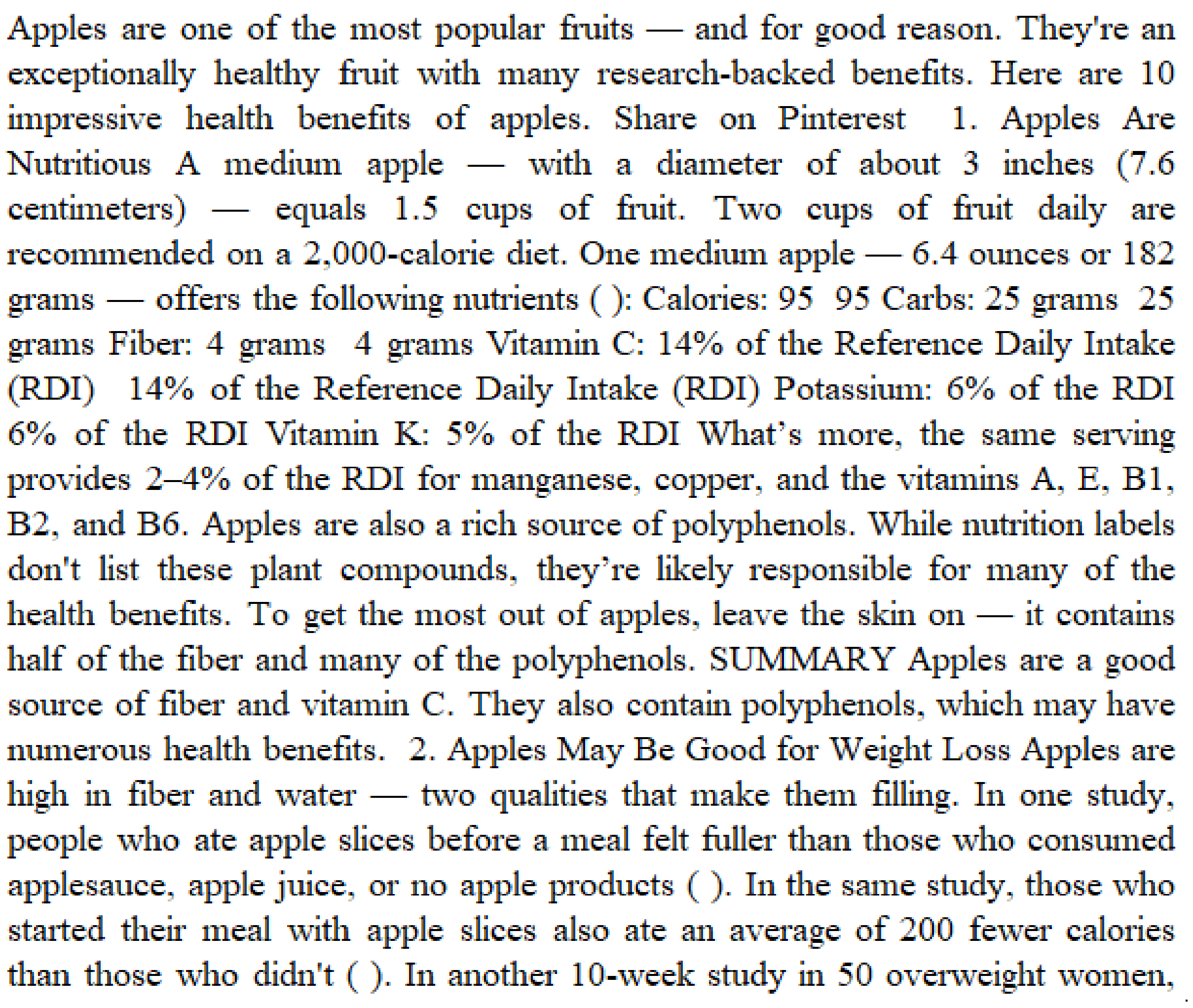

Apples are one of the most popular fruits — and for good reason. They're an exceptionally healthy fruit with many research-backed benefits. Here are 10 impressive health benefits of apples. Share on Pinterest 1. Apples Are Nutritious A medium apple — with a diameter of about 3 inches (7.6 centimeters) — equals 1.5 cups of fruit. Two cups of fruit daily are recommended on a 2,000-calorie diet. One medium apple — 6.4 ounces or 182 grams — offers the following nutrients ( ): Calories: 95 95 Carbs: 25 grams 25 grams Fiber: 4 grams 4 grams Vitamin C: 14% of the Reference Daily Intake (RDI) 14% of the Reference Daily Intake (RDI) Potassium: 6% of the RDI 6% of the RDI Vitamin K: 5% of the RDI What's more, the same serving provides 2–4% of the RDI for manganese, copper, and the vitamins A, E, B1, B2, and B6. Apples are also a rich source of polyphenols. While nutrition labels don't list these plant compounds, they're likely responsible for many of the health benefits. To get the most out of apples, leave the skin on — it contains half of the fiber and many of the polyphenols. SUMMARY Apples are a good source of fiber and vitamin C. They also contain polyphenols, which may have numerous health benefits. 2. Apples May Be Good for Weight Loss Apples are high in fiber and water — two qualities that make them filling. In one study, people who ate apple slices before a meal felt fuller than those who consumed applesauce, apple juice, or no apple products ( ). In the same study, those who started their meal with apple slices also ate an average of 200 fewer calories than those who didn't ( ). In another 10-week study in 50 overweight women,

Figure 7: Sample document provided as input to web scraper

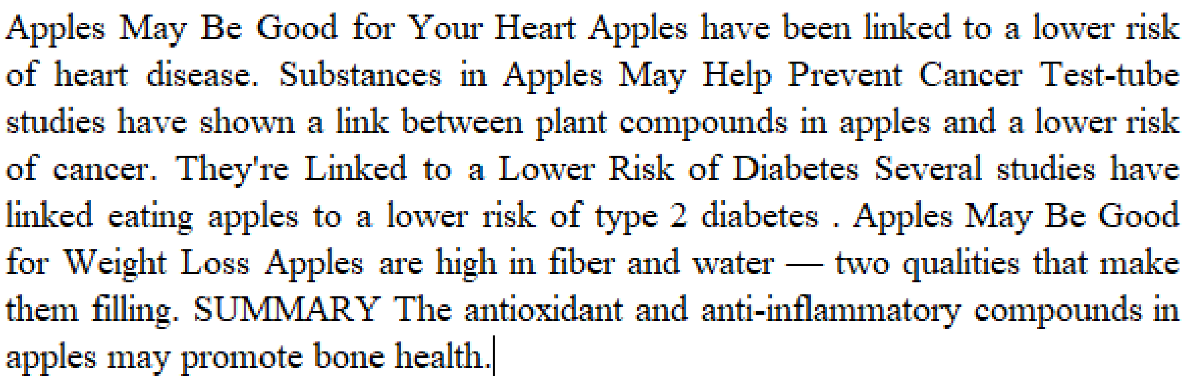

Apples May Be Good for Your Heart Apples have been linked to a lower risk of heart disease. Substances in Apples May Help Prevent Cancer Test-tube studies have shown a link between plant compounds in apples and a lower risk of cancer. They're Linked to a Lower Risk of Diabetes Several studies have linked eating apples to a lower risk of type 2 diabetes . Apples May Be Good for Weight Loss Apples are high in fiber and water — two qualities that make them filling. SUMMARY The antioxidant and anti-inflammatory compounds in apples may promote bone health.

Figure 8: Summarized output from the Text Summarizer

3.3 Sentiment Analysis

Sentiment Analysis is a field in machine learning that comes under the umbrella term Natural Language Processing (NLP). The main objective of sentiment analysis is to analyze the text to understand the sentiment or opinion or emotion behind the text. The main objective of the sentiment analysis to provide automated methods to identify if a text is positive, neutral or negative sentiment. In some cases, one could identify the degree of positive or negative sentiment in a sentence. For example, "The apple is good for health" is a positive sentiment and "The apple is bad for health is a negative sentiment". A neutral sentiment would be "Apple is a fruit". It is just a statement or a fact and does not provide any opinion in the sentence. There are numerous techniques that help us to identify the sentiment of the text.

The applications for sentiment analysis include wide range of fields such as, analyzing the opinions of people during elections, gathering useful information for users reviews on products, using sentiment of financial markets for financial gain in stock markets. The application this project focuses on is using sentiment analysis for analysis of news articles and blogs as well as analysis of search engine queries.

3.3.1 Sentiment

The definition for what a sentiment is wide or varied. The main abstract is sentiment is a opinion, attitude, emotion, subjectivity, or evaluation expressed in a text. The text can be of many forms, they can be reviews, discussions, blogs, news, comments, and feedbacks.

Sentiment can be represented in two values. They are called the sentiment orientation(o) and sentiment intensity(i). Sentiment orientation can be either positive or negative [19]. In some cases, they are neutral because they don't exhibit any polarity. Sentiment intensity is the amount of score that they have towards an orientation, with high score meaning they are extremely opinionated. For example, "The movie is ok" and "The movie is amazing and definitely worth the watch" both express a positive sentiment orientation. But the sentiment intensity of the latter sentence is very high when compared to the former sentence.

3.3.2 Problems

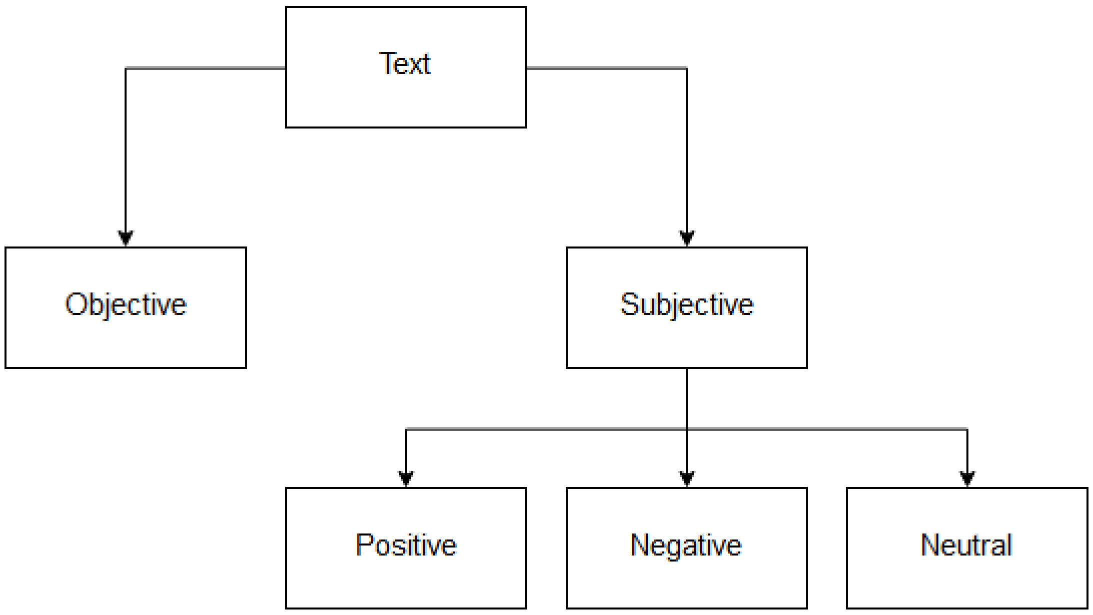


Figure 9: Problems in Sentiment Analysis [31]

As sentiment analysis comes under natural language processing, there are certain aspects that make the accurate prediction of the sentiment very difficult. In some cases, the language is too complex even for a human to exactly predict the sentiment. So, in reality, one can never achieve a 100% accuracy in this field. Sometimes, a sentence said in a positive way can be construed as a negative sentiment and vice versa.

Another aspect of natural language processing that affects sentiment analysis is distinguishing between objective or subjective text. Only if the sentence is considered subjective, further analysis would be needed to determine the polarity of its sentiment. But if a sentiment is an objective sentence, then no further analysis would be needed. This process of classifying the text based on its subjectivity is called subjectivity classification. For example, "Apex Legends is a computer game" is objective, and "Apex Legends is a good computer game" is subjective. All these aspects make sentiment analysis a difficult and an interesting problem to solve.

For this project, sentiment analysis is a part of the whole solution. So, to keep it simple, we have used only the main characteristics of sentiments. For example, the project attempts to only find the polarity of a given sentence and not the intensity towards that polarity.

3.3.3 Levels of Sentiment Analysis

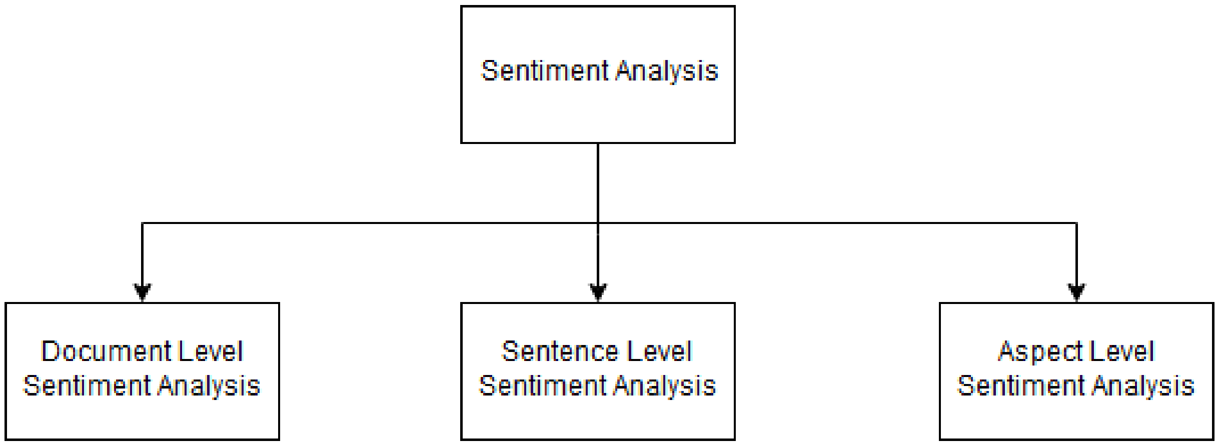

Sentiment Analysis
Document Level Sentiment Analysis
Sentence Level Sentiment Analysis
Aspect Level Sentiment Analysis

Figure 10: Levels of Sentiment Analysis

Sentiment Analysis can be performed on different levels of text. Here, the level mainly indicates the size of the text content on which a sentiment analysis has to be performed,

Document Level: This is the topmost level of sentiment analysis. Here, the entire document is analyzed as a whole. For example, an entire review for a movie or product comes under this category.

Sentence Level: The objective is to classify the sentiment of a single sentence. For example, one-line comments come under this category.

Aspect Level: The objective is to identify the sentiment of the text relative to the aspect term. This involves identifying the aspect term in a sentence and determining the sentiment affecting it. For example, “Apple tastes good and Mango tastes bad”. This sentence has two different sentiments and based on the aspect the sentiment varies.

Apple is an aspect term and has a positive sentiment whereas Mango has a negative sentiment and has a negative sentiment.

This project uses both Document level sentiment analysis and Aspect level sentiment analysis. For the sentiment classification of search engine queries an aspect level sentiment analyzer is used and for the sentiment classification of search engine result like the news articles and blogs, document level sentiment analyzer is used.

3.3.4 Document-level Sentiment Analysis

3.3.4.1 Dataset

Table 1: Statistics for IMDB review dataset

Set	**Total**	**Positive**	**Negative**
Train	15000	7500	7500
Validate	5000	2500	2500
Test	5000	2500	2500

The dataset used for this project is the IMDB movie review dataset. This dataset contains around 50,000 reviews where the amount of positive and negative sentiment reviews is equal. As mentioned, this is very important to remove the bias. The reviews are fully formatted to lower case and special characters like punctuations and numerical values are removed.

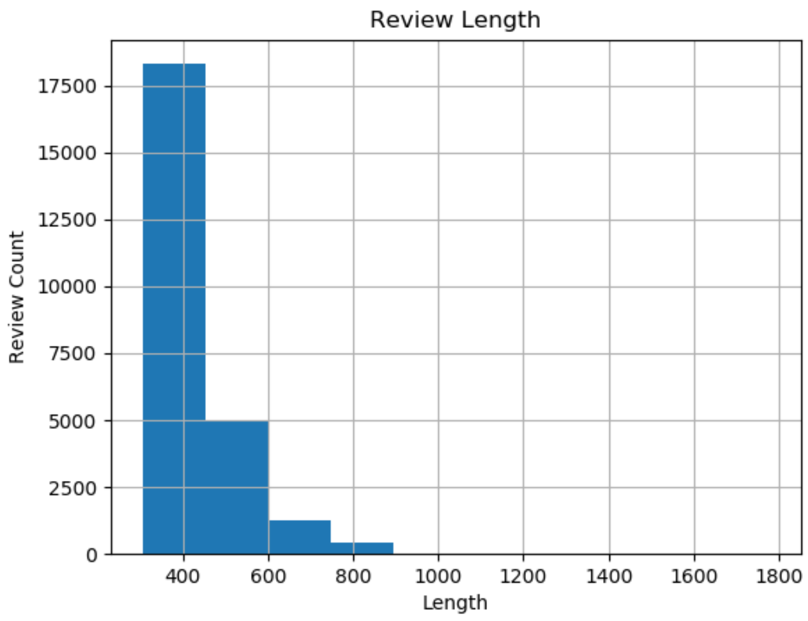


Figure 11: Review length for Document-level Sentiment Analyzer Dataset

Since RNN requires the input data to be of same length, the length of all reviews was analyzed to find the optimal length. The most common length for the reviews were around 400 characters. Hence, the shorter reviews were padded, and the longer reviews were reduced to 400 characters. Finally, the dataset was divided into 60% train data, 20% validation data and 20% test data.

3.3.4.2 Implementation

Word embeddings is a way to represent words in a vector space and allows words with similar meaning to have similar representation in that vector space. For this project, two types of word embedding were used. The first one is a word embedding developed from the words present in the IMDB movie review dataset. The second one was the GloVe [13] word embedding which was developed by understanding the cooccurrence if words over a large training sample. For this project, the GoogleNews GloVe word embedding which was trained on 3 billion words is used. The dataset contains 50-dimensional word vectors.

To understand the working of Recurrent neural networks, one must understand the disadvantages of Feed-Forward Neural Networks. Feed-Forward Neural Networks work only on the current input and does not take sequences or time series data into consideration. As previously mentioned the sequence of words is important for an accurate sentiment analysis. Also, the Feed-Forward Neural Networks do not model memory. They do not carry any room in their model to store the information and pass it along the model. Each state in the model in the model depends only on the current input value.

Recurrent Neural Networks takes into consideration the sequence of data. This means that the information obtained from the previous hidden states are used as inputs to the next hidden states. RNN is deep learning neural network with multiple hidden state each containing multiple neurons and an output state. The inputs are provided at each input state to the neurons and the output gets carried over to other hidden states until it reaches the final state where a loss function is used to determine the output.

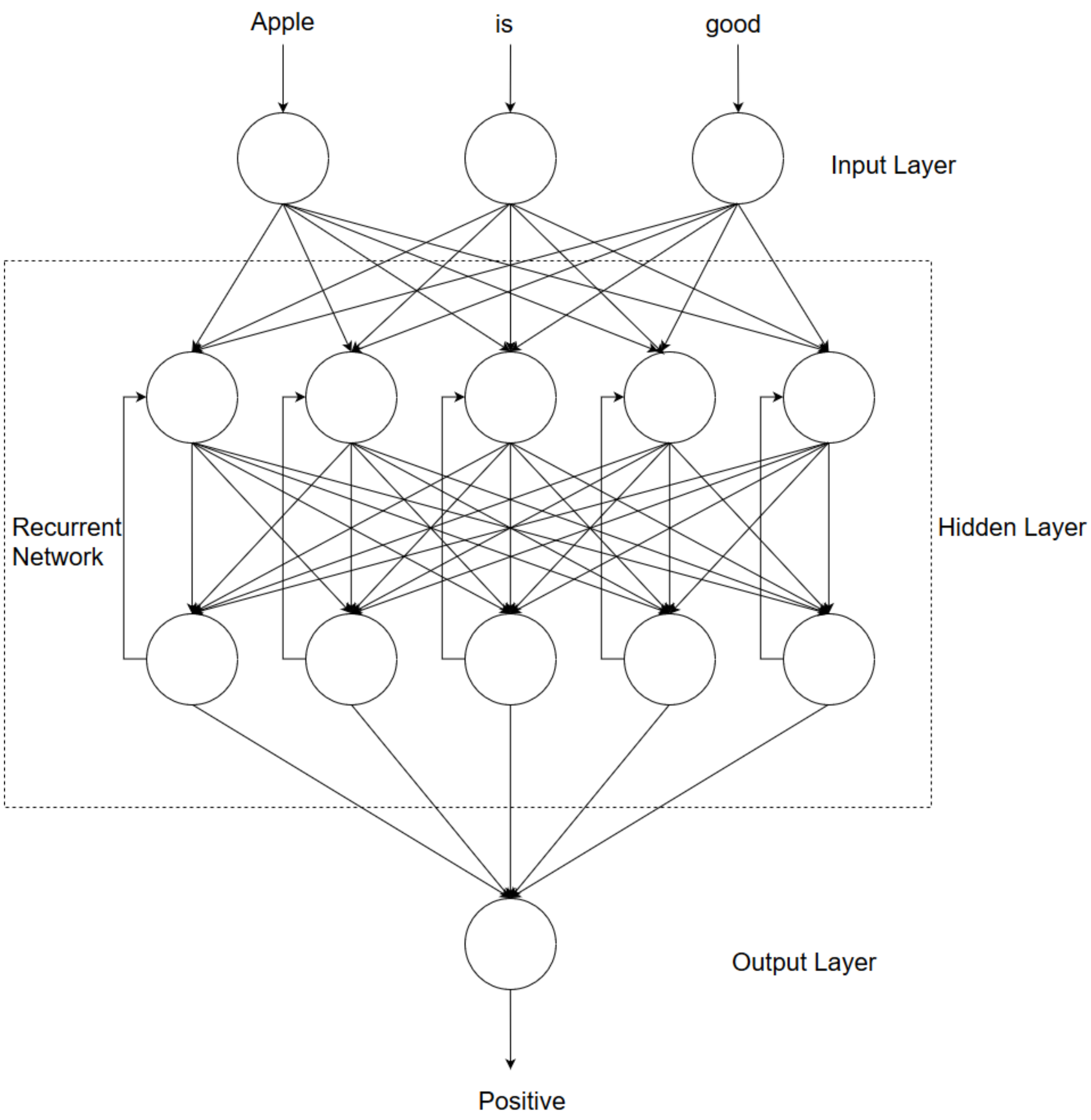


Figure 12: Working of Recurrent Neural Network

A gradient value is used to determine the learning rate of the model. RNN’s have the capability to calculate the cost function or error at any given point. But the problem with this approach is the vanishing gradient problem. For example, an error is found in one hidden state. This means a new cost function is calculated and it needs to back propagate to update previous neurons. But since RNN’s use previous time data to calculate the current state, not only the current neuron but all the previous neuron's weights have to be updated. Here, we multiply the same weight multiple times and when one multiplies the same values with a smaller value multiple times the original value decreases very quickly and vanishes. Hence, the weights of most neurons remain the same and are not updated.

The recursive formula used is,

$$S_t = F_W(S_{t-1}, X_t)$$

Where,

S_t – the current state at time t

F_w – a recursive function

S_{t-1} – The old state at time t-1

X_t – the input value at time t

Long-Short Term Memory (LSTM) solves the vanishing gradient problem and increases the accuracy. An LSTM model has three gates that updater and control the cell states, they are input gate, forget gate and output gate. These gates use the tan function and sigmoid activation functions. The forget gate has the control to make the cell state forget irrelevant information. The input state has the control to make the cell state encode the required information given a new input. The output gate controls what information encoded in the cell state is sent to the network as input in the following time step. The LSTM solves the Vanishing gradient problem by creating a connection the forget gates and the gradients computation and this acts a pathway for information flow without vanishing. In simple terms, this connection acts as a long-term memory for the RNN.

There are several steps involved in the RNN-based LSTM model, they are

1. Tokenize: Covert all the words into the integer tokens
2. Embedding Layer: Map the integer tokens to a embedding of specific dimension
3. LSTM layer: This is defined by the hidden state dimensions and number of layers
4. Fully connected layer: This maps the output of LSTM layer to a desired output size
5. Sigmoid Activation layer: This maps the output values to a binary output class (0 or 1)

A learning rate of 0.001 was chosen to be the optimal value. A BCELoss function and an Adam optimizer is used. The BCELoss function is,

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i . log\big(p(y_i)\big) + (1 - y_i)' log(1 - p(y_i))$$

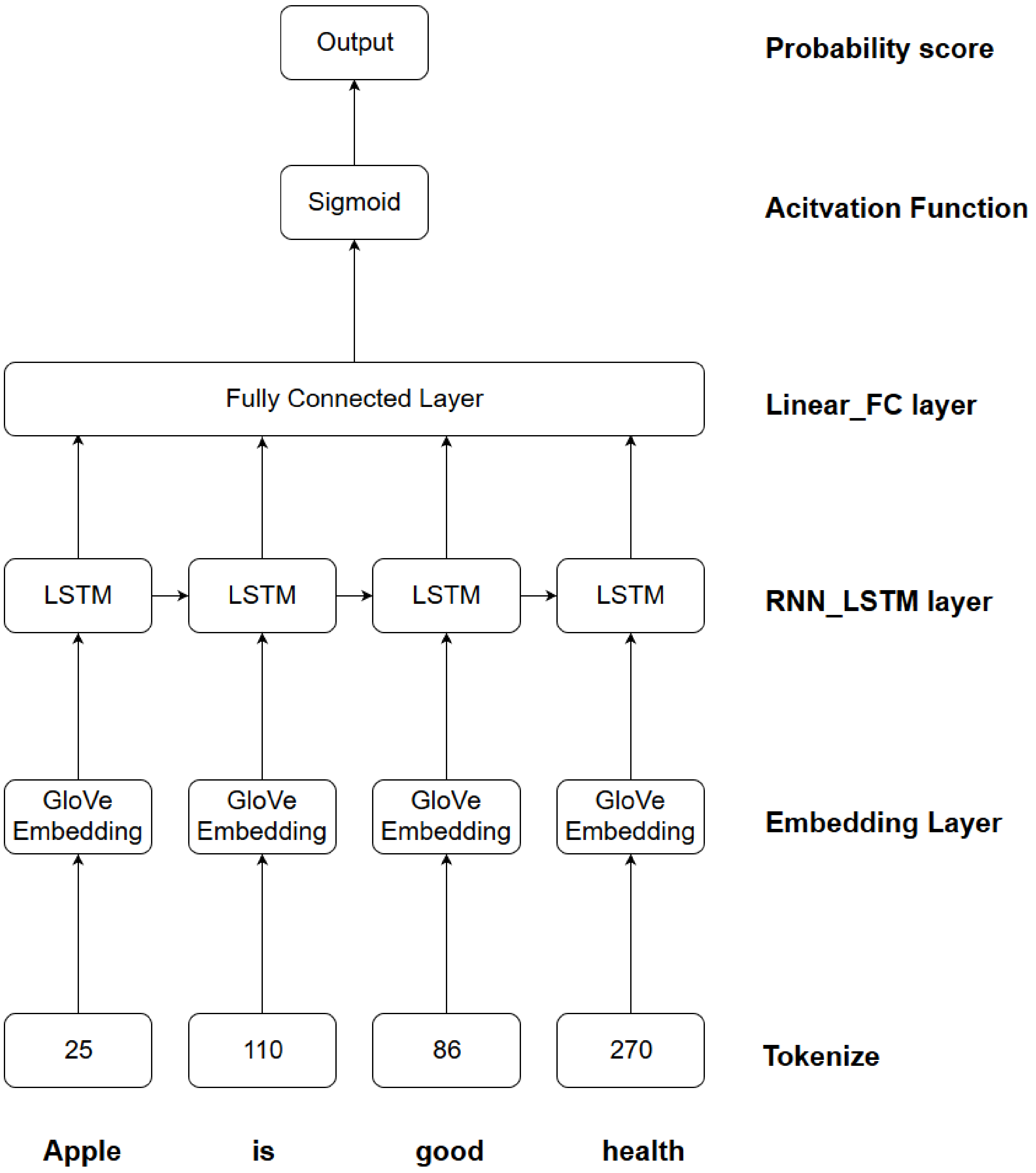


Figure 13: Implementation of Document level sentiment analyzer

3.3.4.3 Result

There were two variations of models tested, one where the embedding is from the input dataset itself and the other where the embedding was provided by the GloVe word2Vec embedding model. For the former, an accuracy of 86% was obtained and for the GloVe embedding an accuracy of 91% was obtained.

Table 2: Comparison of different Document-level Sentiment Analyzer

Method	**Accuracy**
Bag of Words SVM [32]	87.15
Add'l Unlabeled + Bag of Words [32]	88.89
n-grams [32]	90.20
RNN + LSTM	**91**

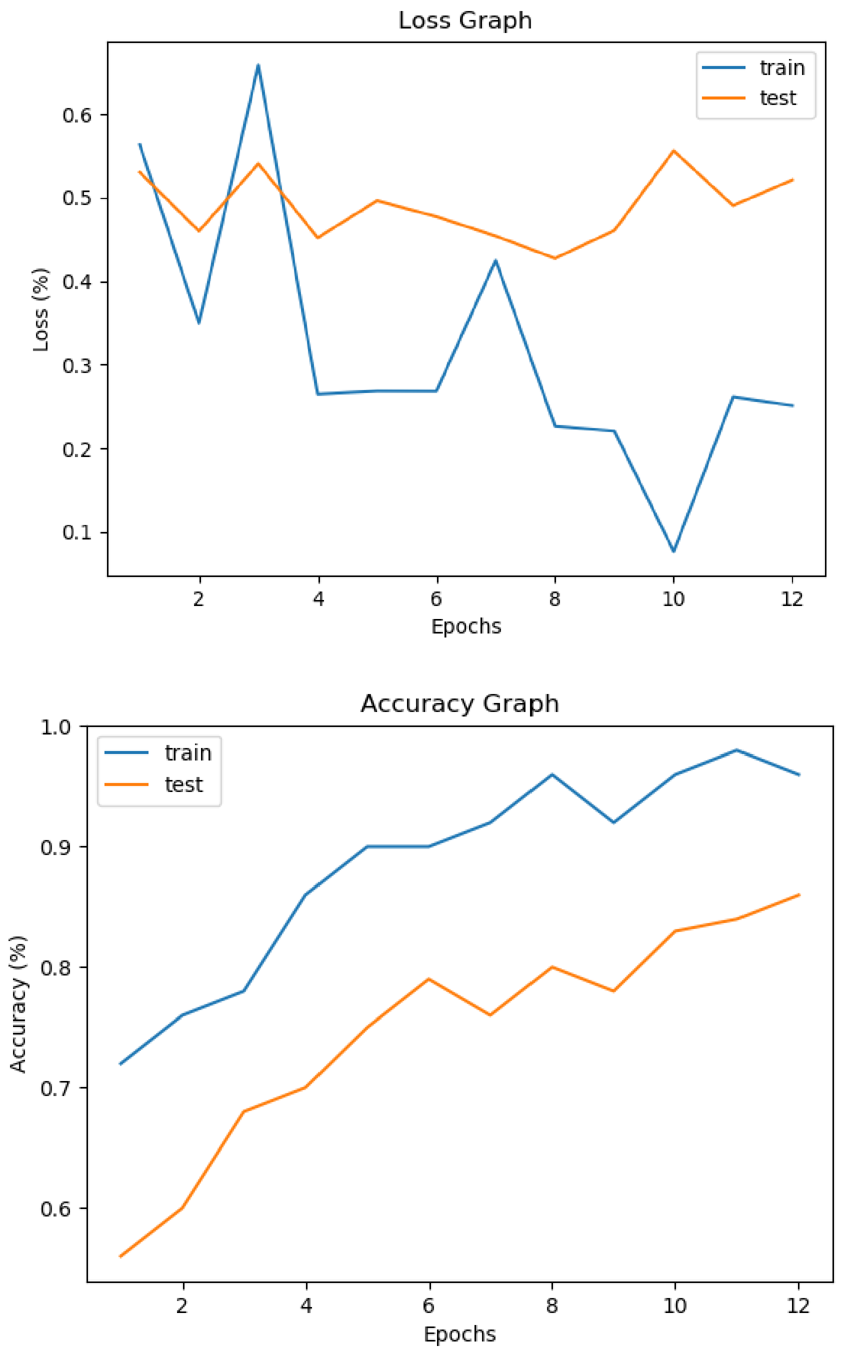
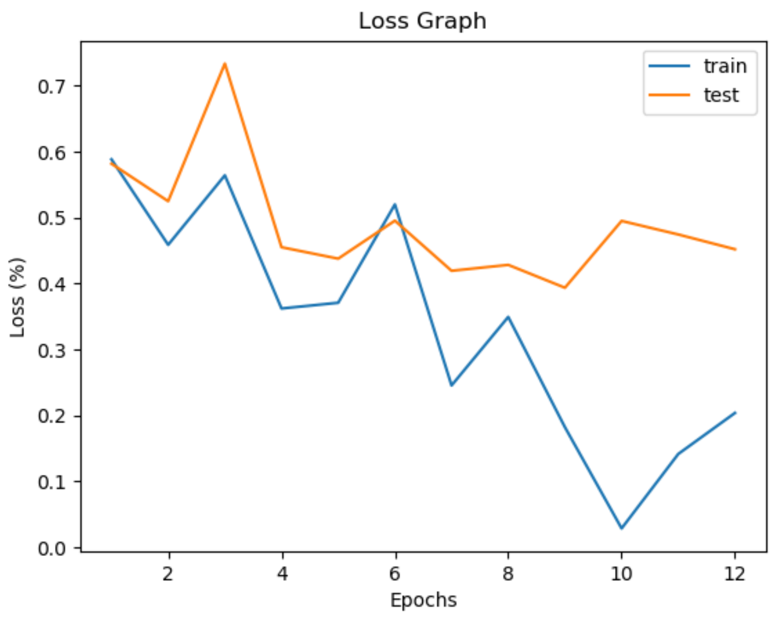


Figure 14: Loss Graph and Accuracy graph for model with no GloVe word embedding



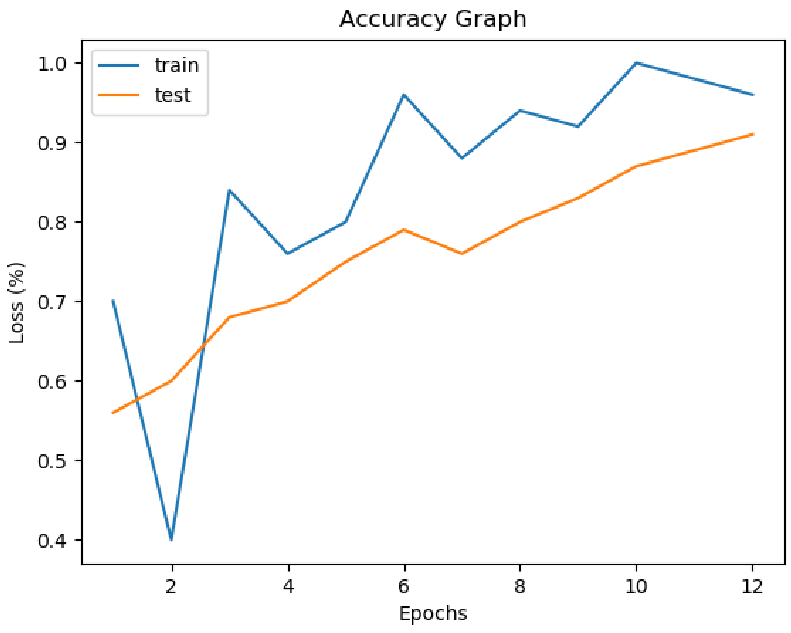


Figure 15: Loss Graph and Accuracy graph for model with GloVe word embedding

3.3.5 Aspect Level Sentiment Analysis

Aspect Level Sentiment Analysis is used for identifying the sentiment of a particular aspect present in a sentence. Usually a sentence can contain two different sentiments related to two different aspects. For example, "A coffee was good, but the ambience was bad". The above sentence has both positive and negative sentiments around them. If we consider coffee as the aspect term, then it is a positive sentence but if we consider ambience, then it has a negative sentiment. This distinction is needed when accurately predicting the sentiment of a query in search engines. There are several applications for this kind of sentiment analysis namely, reviews that talk about different entities at the same time, documents that cover wide range of topics and much more.

There are relatively two problems that need to be solved in order to achieve Aspect Level Sentiment Analysis. First, one has to identify the aspect of a given sentence. Second, the actual sentiment classification based on the chosen aspect. For simplicity, we only develop a model for the second problem. We assume that the aspect term for a sentence is already known. There are relatively two types of aspect level sentiment classification. One is to find the general aspects in the sentence and find the sentiment polarity around them and the other is to find the sentiment polarity of the actual terms present in the sentence.

Even though this project considers only the classification of sentences based on their aspects, there are several different methods that can be employed to achieve this. These methods include dictionary-based, supervised and unsupervised machine learning models. Dictionary based method uses a dictionary to find the sentiment values of each words, which is followed by an association step to assign the sentiment of the surrounding words to the aspect itself. Supervised and unsupervised machine learning models uses sophisticated methods such as sentiment lexicons and word embeddings in combination with LSTMs and other models to provide an accurate prediction.

3.3.5.1 Dataset

The dataset used for this experiment includes reviews for Laptops and Restaurants. Each of these datasets have multiple sentiment polarities, they are '0' is negative, '1' is neutral and '2' is positive. They have adopted F-score as their final accuracy score. As a preprocessing step, the words present in the input sentences are converted to word embeddings. The word embedding used for this model is a 300-dimension pre-trained word vectors from GloVe.

To make this model detect queries with more accuracy, query sentences were included to the existing review dataset. This helped the model get a better accuracy for certain type of queries. For example, queries like "A *fruit name* is good/bad for health" was used in the training dataset to get better accuracy.

Table 3: Statistics for two datasets

Dataset	Set	Total	Positive	Negative	Neutral
Restaurant	Train	3017	1806	669	542
	Validate	1120	728	196	196
	Test	591	358	138	95
Laptop	Train	1934	823	730	381
	Validate	638	341	128	169
	Test	394	171	140	63

3.3.5.2 Implementation

The aspect level sentiment analysis for this project was primarily derived from the paper by Zhu et. al [24]. This paper uses a novel approach based on the of aspect level sentiment classification with the help of Auxiliary memory.

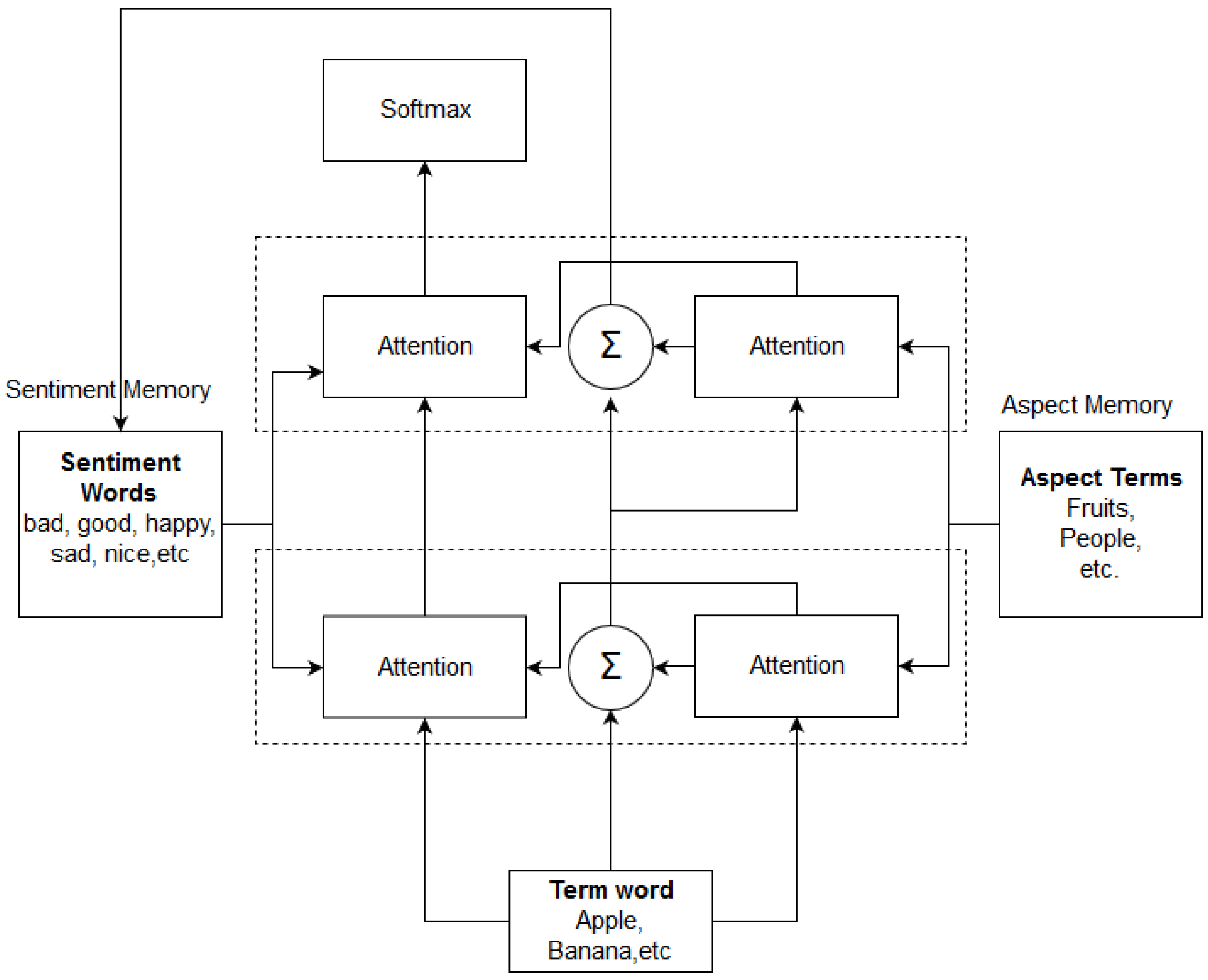


Figure 16: Implementation structure of Aspect-level Sentiment Analyzer [24]

There are two memory units involved in their process, the sentiment memory and the aspect memory. The aspect memory holds all the aspect representations and the sentiment memory for sentiment classifications.

The aspect memory takes as input the aspect term and produces an output representation which is a combination of aspect term and aspect information. The weight for each piece of memory is calculated by the following formula,

$$\alpha_i = \frac{exp(u_i)}{\sum_{j=1}^{k} exp(u_j)}$$

Here, α_i is the weight of memory m_i and u_i is the sematic relatedness between the term and each aspect.

The first step in this model is to convert all the words into a multidimensional vector called word embeddings. So, for example given a sentence with multiple words, each word 'w' goes through a lookup layer to get their corresponding embedding vectors. Then an aspect memory is built with stacks of aspect information. Hence, when a term is given as input to the memory, it can generate an output which is a combination of aspect information and the term information. Similarly, a sentiment memory is built where its main goal is to identify the sentiment of the given terms. Here, the output from the aspect memory is fed into the sentiment memory and along with a feed forward neural network, the sentiment polarity is calculated for the given term.

This represents one layer of aspect and sentiment memory. They have postulated that multiple layers produce better accuracy than a single layer. Hence, the output from the above layer is passed as input to another set of aspect and sentiment memory layer. For example, the output of aspect memory is fed to the next aspect memory and the output of the sentiment memory and the aspect memory is fed to the next sentiment memory.

There are multiple loss functions present in [24], for different predictions such as term sentiment, term prediction and aspect regularization. For this project, only the term sentiment classification is used. For example, in "Kale is good for health", the term is kale (w_t) and the input sentence (s) is the query

$$L = -\sum_{(s,w_t)\in T}\sum_{c\in C} y_c(s, w_t).\log P_c(s, w_t)$$

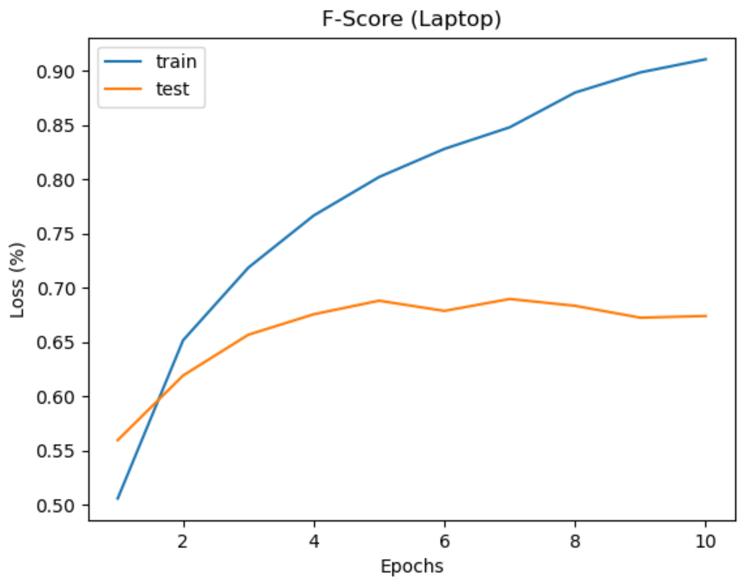
Here, the (s, w_t) denotes the sentence-term pair. And the $P_c(s,w_t)$ is the probability function for predicting category c. The ground truth or the actual value is provided by $y_c(s, w_t)$ and T is the set of training data and C is the set of sentiment categories (positive, negative and neutral).

3.3.5.3 Result

The experiment was conducted with different parameter variations. Parameters like aspect ratios, number of layers where varied and the result is present in the graph below. A maximum accuracy of 70 was obtained for Laptop review dataset and an accuracy of 79 was obtained for the Restaurant review dataset. The F-score is calculated by the following formula,

$$F\text{-}score = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

The table 4 shows the comparison with other models. The restaurant dataset is used for obtaining the accuracy and the F-score.



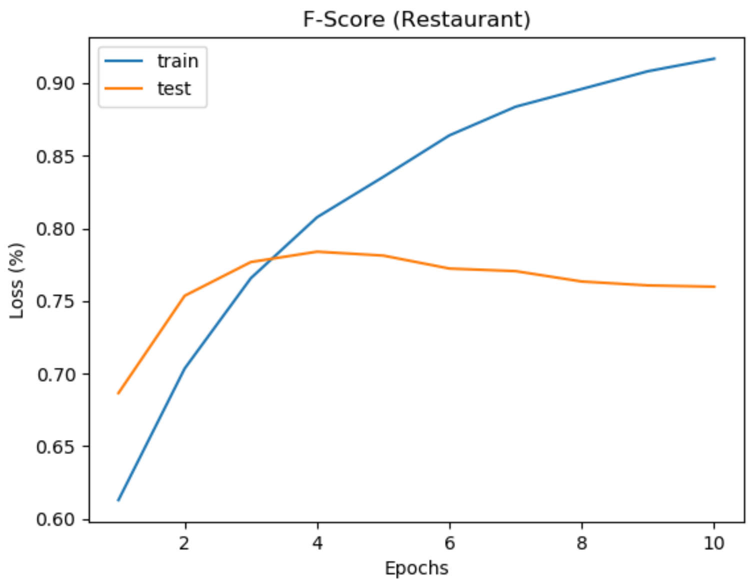


Figure 17: F-score for Restaurant and Laptop with increasing Epochs

Table 4: Comparison of different Aspect-level Sentiment Analyzer

Methods	**Accuracy**	**F-Score**
ContextAVG	75.09	63.96
LSTM [27]	75.37	64.51
LSTM + ATT [28]	76.83	66.48
MemNN [29]	80.09	72.10
DAuM	**82.32**	**77.45**

The methods used include ContextAVG which averages the vectors for every context word and then concatenated with aspect vectors. Tang et al. uses considers both the information before and after the aspect term for better prediction. In addition to [27], Wang et al. uses attention mechanism to capture important terms in the initial stages of model development and pass them to the final stage for improvement. MemNN uses an end-to-end deep memory network with an external memory to hold important context words.

3.4 Sentiment Style Transfer

One of the main objectives of this project is to provide the user with complete information. This means converting a positive sentiment query into negative and vice versa. For example, if a user query is "MacBook is a great laptop", this query has a positive sentiment and the main aspect of this query is MacBook. This has to be converted into a negative sentiment and the output should be "MacBook is a bad laptop". The concept of changing the style of a sentence without changing its original sematic meaning is called Text Style transfer. This is not restricted only to sentiments but also to other styles such as removing sarcasm, introducing comedy, and much more.

3.4.1 Dataset

The dataset plays an important role in style transfer. The dataset for such transfers can either be parallel data or non-parallel data. Parallel data means the dataset is made of sentences which have both sentiment polarities. For example, the below table shows the sentences with positive sentiment and the same sentences with negative sentiment.

Table 5: Statistics of two datasets

Dataset	**Set**	**Positive**	**Negative**
Yelp	Train	27000	18000
	Validate	2000	2000
	Test	500	500
IMDB	Train	27700	27800
	Validate	985	1015
	Test	500	500

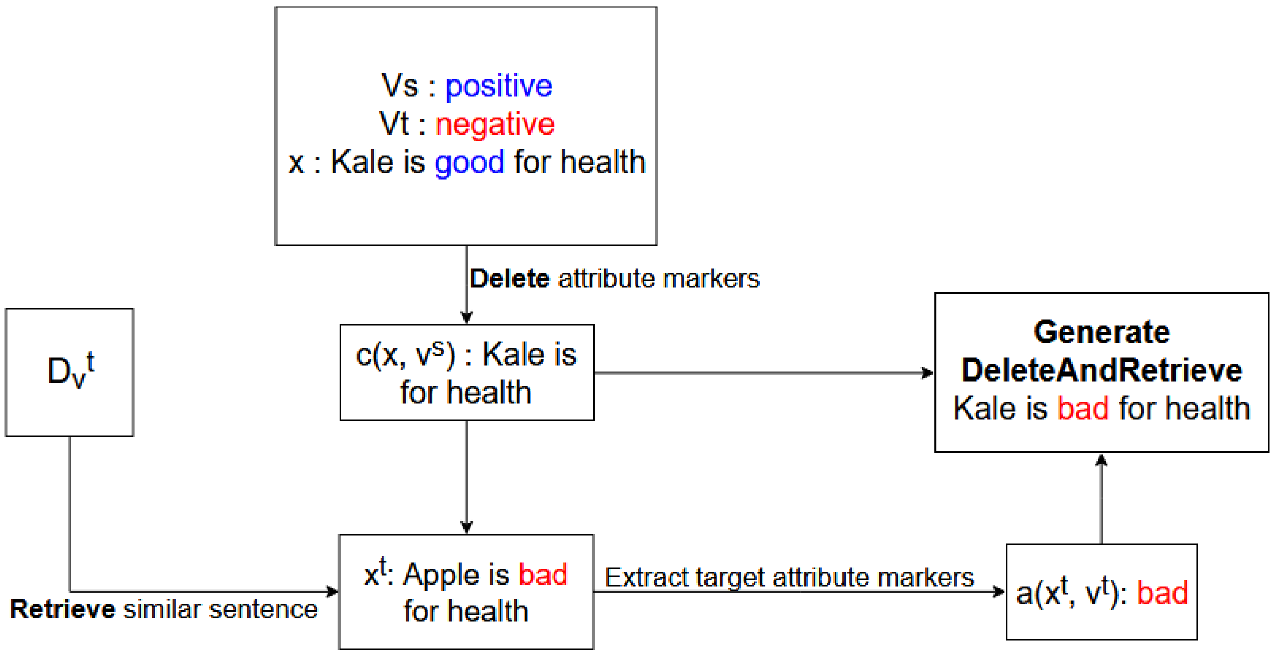
Non-parallel data means the dataset is made of sentences that have a particular sentiment and does not have the same sentences with opposite sentiment. For example, the below table shows the sentences with positive sentiment and different sentence with negative sentiment.

It is difficult to find a dataset which has parallel data, so this project uses both the non-parallel dataset and parallel dataset. The dataset used is Yelp restaurant review and IMDB movie review dataset. During the delete and retrieve phase only the nonparallel dataset is used. But while building the final encoder-decoder RNN model parallel data is used.

3.4.2 Implementation

Li. et al proposed a simple technique for sentiment style transfer. The main motivation for their project is, to transfer the style of a text it is not necessary to completely change the text but identify only the crucial keyword and change them. This means to identify the sentiment from the sematic meaning and change only the sentiment of the sentences without changing the semantic meaning.

So, for example, if the given sentence is "The movie is awesome", here the keyword is "awesome". The first step is to identify and remove such words. Then to replace the void with the most appropriate word of opposite sentiment.



Vs : positive
Vt : negative
x : Kale is good for health
Delete attribute markers
D_v^t
$c(x, v^s)$: Kale is for health
Generate DeleteAndRetrieve Kale is bad for health
Retrieve similar sentence
x^t: Apple is bad for health
Extract target attribute markers
$a(x^t, v^t)$: bad

Figure 18: Implementation steps for Sentiment Style Transfer [25]

The three main components involved in their project include,

1. **Delete** – identify and delete the keywords which provide sentiment to the sentence.
2. **Retrieve** – Search through the corpus to find target sentence with same context but opposite sentiment.
3. **Generate** – Build the final output sentence with the help of input content and target sentence.

Delete:

They have used a salience smoothing approach to delete the n-grams that have the most sentiment polarity. They term these words as the attribute markers. When the n-grams for a given sentence crosses a threshold γ, they are termed as the attribute markers for that sentence and are marked to be removed from the sentence.

$$s(u, v) = \frac{count(u, D_v) + \lambda}{(\sum_{v' \in V, v' \neq v} count(u, D_{v'})) + \lambda}$$

Here, *u* is the n-gram with respect to *v*, *v* is the current sentence and D_v is the entire dataset. Finally, all the attribute markers are denoted as $a(x, v^{src})$ and the sentences that remain after the deletion operation are denoted as $c(x, v^{src})$.

Retrieve:

The main strategy employed to retrieve the words that need to be replaced for the words that were deleted is to find sentence similar to the input sentence. For example, if the input sentence was "The macbook is …", the word that is going to fill the gap would most likely be "great" than "delicious". To find sentences with similar meaning that of the input sentence, the Euclidean distance of all the sentences present in the word embedding is compared.

$$x^{tgt} = \underset{x' \in D_v tgt}{argmin}\ d(c(x, v^{src}), c(x', v^{tgt}))$$

Generate:

In the generate phase, four different approaches are used to obtain the final result, they are RetrieveOnly, TemplateBased, DeleteOnly, DeleteAndRetrieve. For the scope of my project, DeleteAndRetrieve method provided the best results.

The DeleteAndRetrieve method first deletes the attribute markers. Then, it finds the target sentence with least Euclidean distance and retrieves the attribute markers. Finally a encoder decoder model is used for inserting the attribute markers a(xtgt,vtgt) into the input sentence c(x,vsrc) . The RNN encoder encodes the attribute markers and sequence of attribute markers. The encoder uses a bidirectional LSTM model which is similar to the model developed for document level sentiment analysis.

The RNN decoder decodes these two values to generate the output y. The decoder is made of Stacked Attention LSTM model (SANs). SANs use multi-step reasoning to predict the answer. This helps the model to eliminate noise and pinpoint the regions that are highly relevant to the answer [26]. These encoders and decoders are trained with a parallel dataset where the sentence has both positive and negative sentiment.

3.4.3 Result

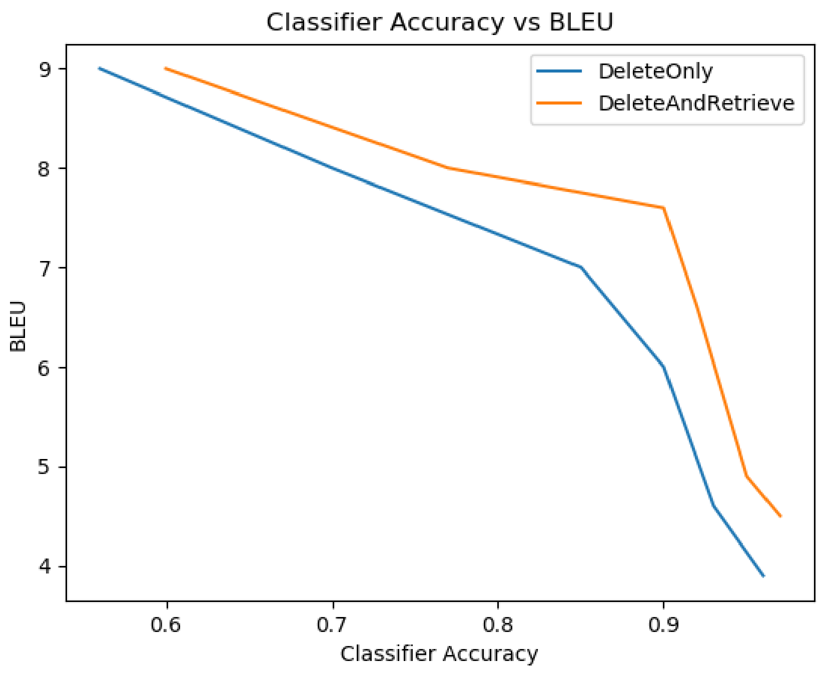


Figure 19: Classifier Accuracy vs BLEU score for DeleteOnly and DeleteAndRetrieve.

There are two main criteria on which this model is tested. First criteria is the sentiment classifier model to identify if the sentiment is successfully changed and the second criteria is through BLEU scores which checks the degree to which semantic meaning of the sentence is changed. These values are given in the table below.

Table 6: Comparison of different Sentiment Style Transfer methods

Method	**Accuracy (Sentiment Classifier)**	**BLEU**
CrossAligned [29]	73.7	3.1
Style Embedding [33]	8.7	11.8
Multi Decoder [33]	47.6	7.1
DeleteOnly [25]	**85.7**	**7.5**
DeleteAndRetrieve [25]	**88.7**	**8**

The cross-aligned autoencoder [29] and the multi-decoder model loses the meaning of the sentence. Also, the sentiment classifier accuracy is very poor for both Multi-Decoder and Style-Embedding.

Chapter 4:
End-to-End Model Evaluation

4.1 Dataset

The dataset for this project was built by analyzing the web for most controversial food items. The list of food items was then appended with statements having different sentiment polarities. For example, one of the controversial food items is kale. The statement appended to this food item is "Kale is good for health" and "Kale is bad for health". By doing this, a positive sentiment and a negative sentiment query is obtained. In total 250 such items were obtained, and 500 queries were used.

4.2 Implementation

The overall process involved in the project is given in the above diagram. First, a query is entered by the user. The query is analyzed to find its sentiment polarity. This is done through aspect-level sentiment analysis. Then the sentiment of the query is changed to its opposite polarity by Sentiment style transfer model. Consider the query entered by user is "Orange is good for health". First the aspect-level sentiment analyzer will analyze the sentence and provide its sentiment. In this case it is positive. Next, the sentiment style transfer model will change the sentiment of the given query. The result is be "Orange is bad for health" which is a negative sentiment. Next, the two queries, with positive sentiment and negative sentiment is used to query the search engine and retrieve relevant documents by the web scraper.

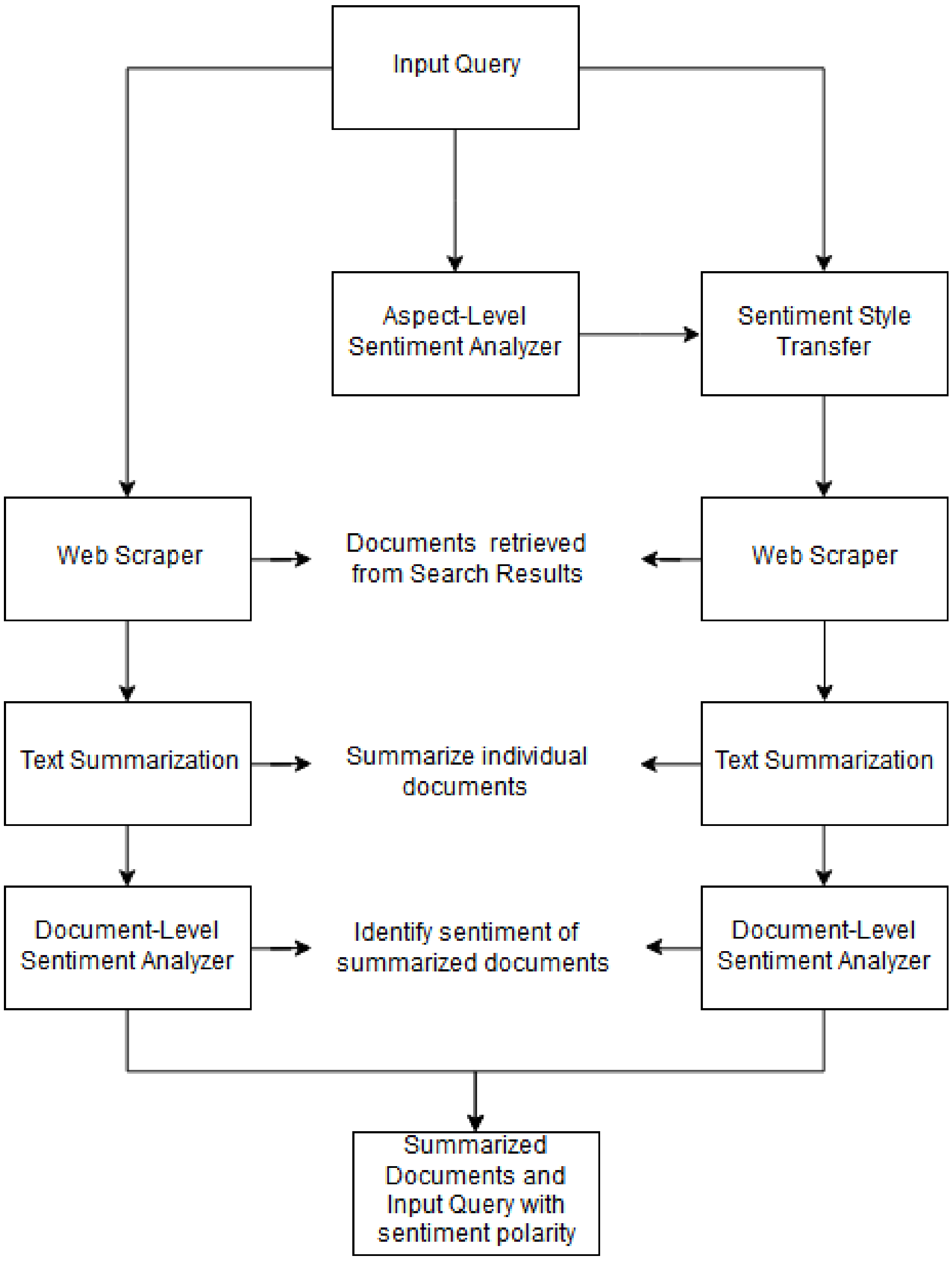

Input Query
Aspect-Level Sentiment Analyzer
Sentiment Style Transfer
Web Scraper
Documents retrieved from Search Results
Web Scraper
Text Summarization
Summarize individual documents
Text Summarization
Document-Level Sentiment Analyzer
Identify sentiment of summarized documents
Document-Level Sentiment Analyzer
Summarized Documents and Input Query with sentiment polarity

Figure 20: Overall project structure

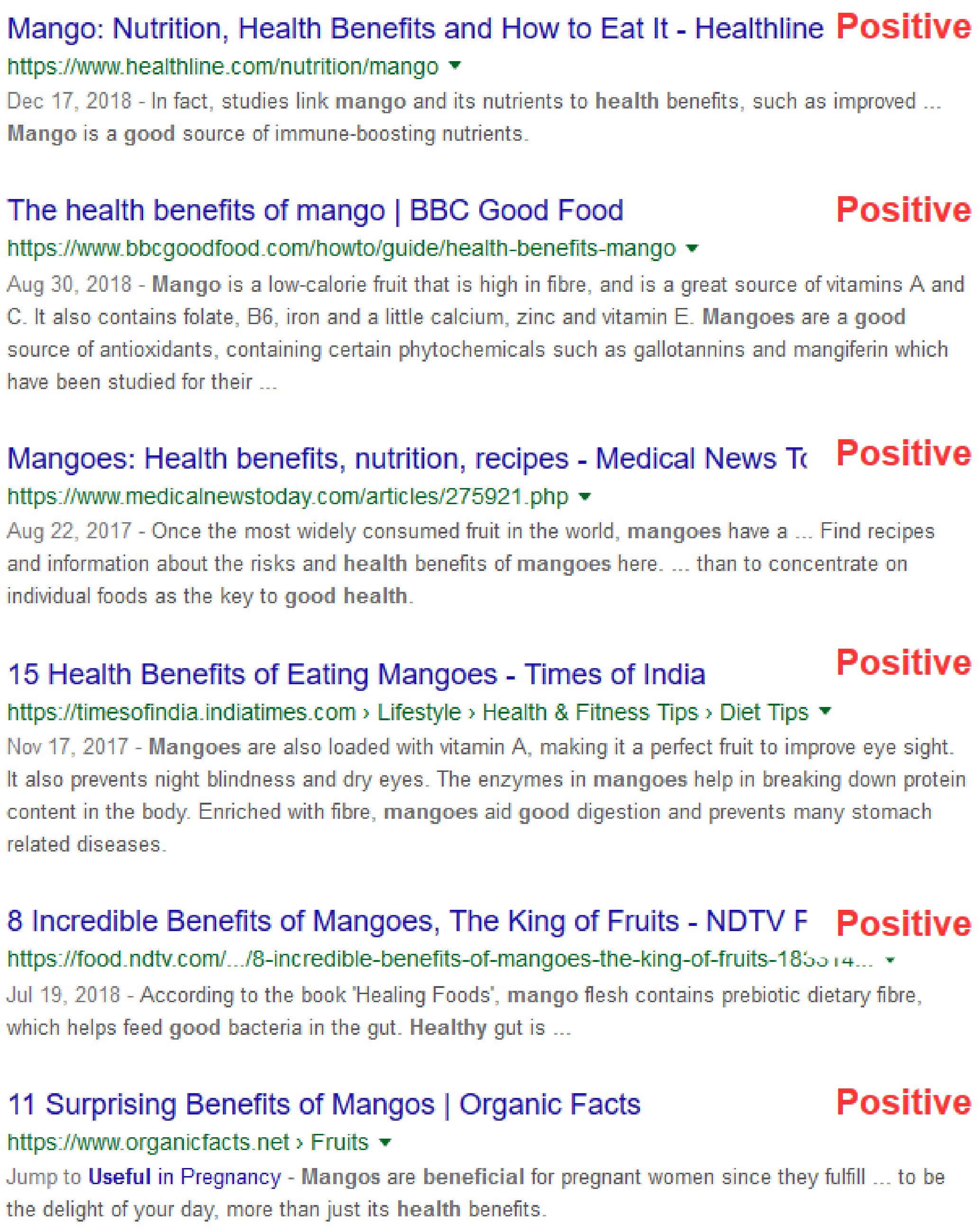
The obtained documents are summarized into a shorter version through text summarization model. Next, the sentiment of these summarized documents is predicted by the document-level sentiment analyzer. Finally, the user is provided with the following information.

1. The sentiment of the query they entered.
2. The modified query with opposite sentiment.
3. The documents that are retrieved from the web.
4. The summarized documents with only the relevant information.
5. Sentiment of each summarized document.

4.3 Result

In total, 500 user queries with 250 positive and 250 negative queries were used. Each Query (Q) will generate a minimum of 10 news articles (A). These articles are analyzed by the sentiment analyzer for their sentiment polarity (S). The final average is calculated by taking the count of the sentiment polarity for these articles. For the positive queries an average of 67% of the documents are positive and for negative queries an average of 70% of the documents are negative.

To remove the sentiment bias in the search engine, the search results for both positive and negative query is collected, and each individual result is analyzed. All the positive search results are placed in one side and the negative search results are placed in the other side. For example, “Mango is good for health” is a positive search query. The document analyzer will tag all the search results obtained for this query.



Mango: Nutrition, Health Benefits and How to Eat It - Healthline **Positive**
https://www.healthline.com/nutrition/mango ▾
Dec 17, 2018 - In fact, studies link **mango** and its nutrients to **health** benefits, such as improved ... **Mango** is a **good** source of immune-boosting nutrients.

The health benefits of mango | BBC Good Food **Positive**
https://www.bbcgoodfood.com/howto/guide/health-benefits-mango ▾
Aug 30, 2018 - **Mango** is a low-calorie fruit that is high in fibre, and is a great source of vitamins A and C. It also contains folate, B6, iron and a little calcium, zinc and vitamin E. **Mangoes** are a **good** source of antioxidants, containing certain phytochemicals such as gallotannins and mangiferin which have been studied for their ...

Mangoes: Health benefits, nutrition, recipes - Medical News Tc **Positive**
https://www.medicalnewstoday.com/articles/275921.php ▾
Aug 22, 2017 - Once the most widely consumed fruit in the world, **mangoes** have a ... Find recipes and information about the risks and **health** benefits of **mangoes** here. ... than to concentrate on individual foods as the key to **good health**.

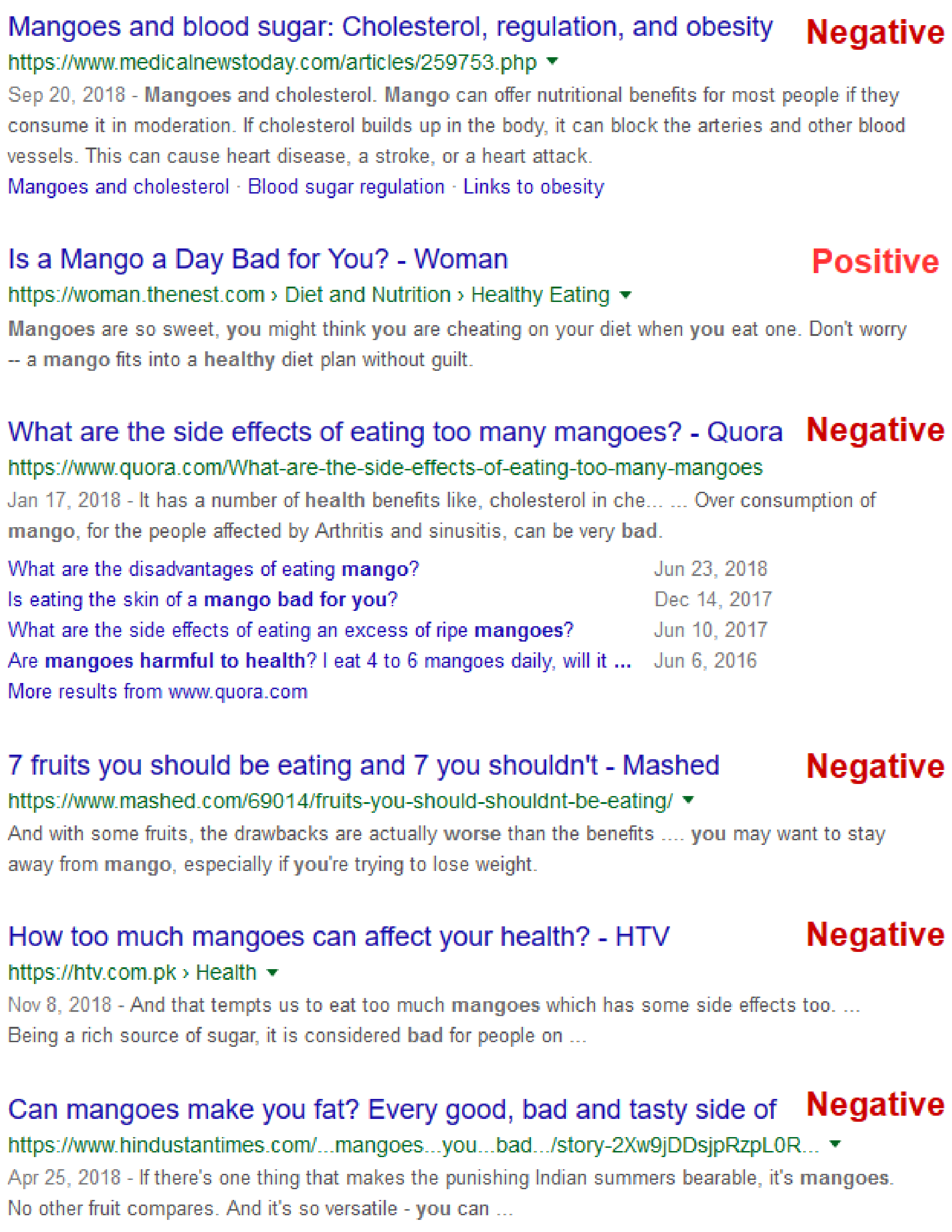
15 Health Benefits of Eating Mangoes - Times of India **Positive**
https://timesofindia.indiatimes.com › Lifestyle › Health & Fitness Tips › Diet Tips ▾
Nov 17, 2017 - **Mangoes** are also loaded with vitamin A, making it a perfect fruit to improve eye sight. It also prevents night blindness and dry eyes. The enzymes in **mangoes** help in breaking down protein content in the body. Enriched with fibre, **mangoes** aid **good** digestion and prevents many stomach related diseases.

8 Incredible Benefits of Mangoes, The King of Fruits - NDTV F **Positive**
https://food.ndtv.com/.../8-incredible-benefits-of-mangoes-the-king-of-fruits-1835514... ▾
Jul 19, 2018 - According to the book 'Healing Foods', **mango** flesh contains prebiotic dietary fibre, which helps feed **good** bacteria in the gut. **Healthy** gut is ...

11 Surprising Benefits of Mangos | Organic Facts **Positive**
https://www.organicfacts.net › Fruits ▾
Jump to **Useful** in Pregnancy - **Mangos** are **beneficial** for pregnant women since they fulfill ... to be the delight of your day, more than just its **health** benefits.

Figure 21: Search results and its sentiment for a positive search query

From figure 21, one could deduce that a positive query produces documents that are mostly positive. Next, the sentiment style transfer model will transform this positive query into a negative one. The output from the style transfer model is “Mango is bad for health”.



Mangoes and blood sugar: Cholesterol, regulation, and obesity **Negative**
https://www.medicalnewstoday.com/articles/259753.php ▾
Sep 20, 2018 - **Mangoes** and cholesterol. **Mango** can offer nutritional benefits for most people if they consume it in moderation. If cholesterol builds up in the body, it can block the arteries and other blood vessels. This can cause heart disease, a stroke, or a heart attack.
Mangoes and cholesterol · Blood sugar regulation · Links to obesity

Is a Mango a Day Bad for You? - Woman **Positive**
https://woman.thenest.com › Diet and Nutrition › Healthy Eating ▾
Mangoes are so sweet, **you** might think **you** are cheating on your diet when **you** eat one. Don't worry -- a **mango** fits into a **healthy** diet plan without guilt.

What are the side effects of eating too many mangoes? - Quora **Negative**
https://www.quora.com/What-are-the-side-effects-of-eating-too-many-mangoes
Jan 17, 2018 - It has a number of **health** benefits like, cholesterol in che... ... Over consumption of **mango**, for the people affected by Arthritis and sinusitis, can be very **bad**.

What are the disadvantages of eating **mango**?	Jun 23, 2018
Is eating the skin of a **mango bad for you**?	Dec 14, 2017
What are the side effects of eating an excess of ripe **mangoes**?	Jun 10, 2017
Are **mangoes harmful to health**? I eat 4 to 6 mangoes daily, will it **...**	Jun 6, 2016

More results from www.quora.com

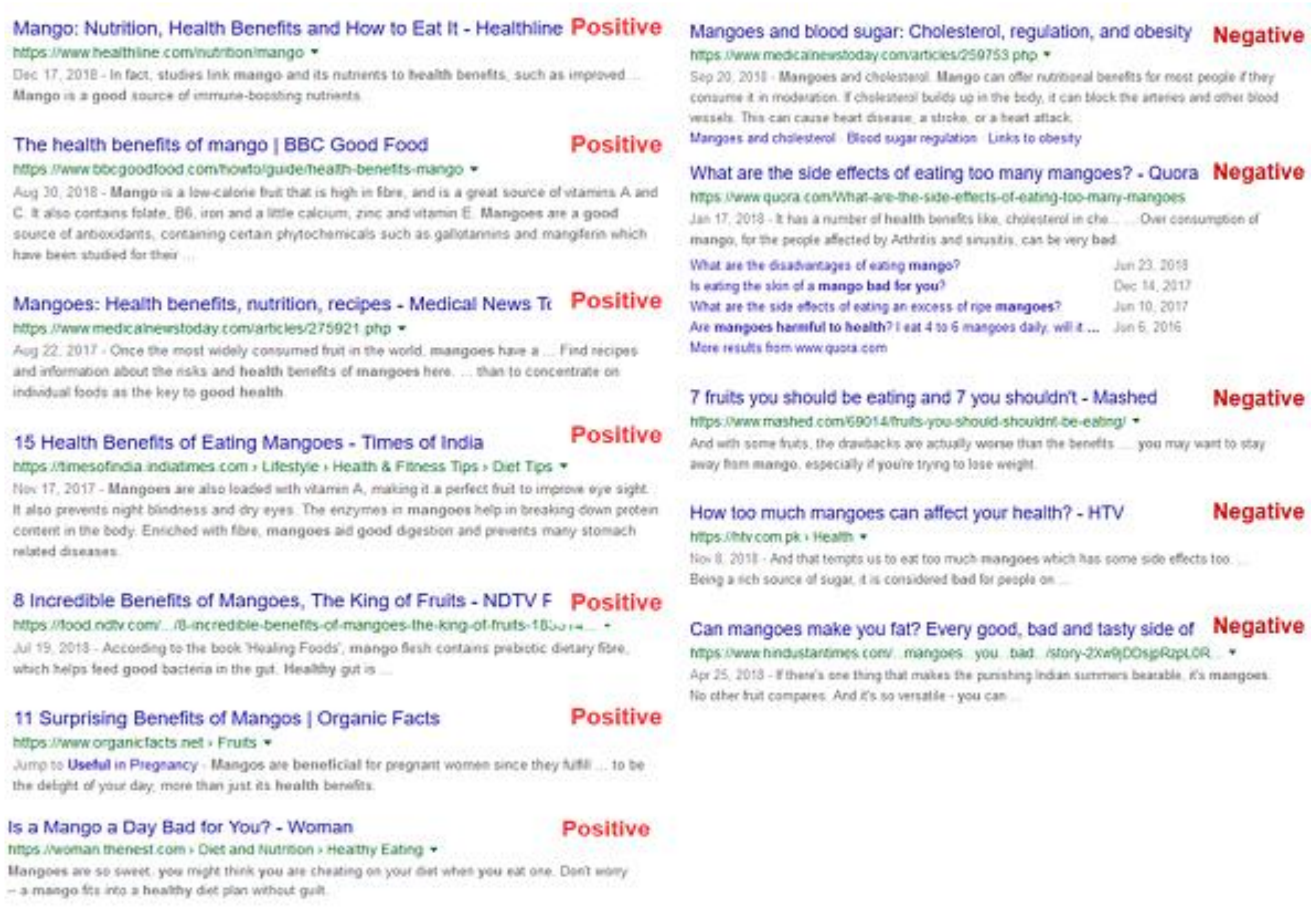
7 fruits you should be eating and 7 you shouldn't - Mashed **Negative**
https://www.mashed.com/69014/fruits-you-should-shouldnt-be-eating/ ▾
And with some fruits, the drawbacks are actually **worse** than the benefits .... **you** may want to stay away from **mango**, especially if **you**'re trying to lose weight.

How too much mangoes can affect your health? - HTV **Negative**
https://htv.com.pk › Health ▾
Nov 8, 2018 - And that tempts us to eat too much **mangoes** which has some side effects too. ... Being a rich source of sugar, it is considered **bad** for people on ...

Can mangoes make you fat? Every good, bad and tasty side of **Negative**
https://www.hindustantimes.com/...mangoes...you...bad.../story-2Xw9jDDsjpRzpL0R... ▾
Apr 25, 2018 - If there's one thing that makes the punishing Indian summers bearable, it's **mangoes**. No other fruit compares. And it's so versatile - **you** can ...

Figure 22: Search results and its sentiment for a negative search query

The search results produced by this negative sentiment is mostly negative. Now, the positive and negative search results are separated and provided side by side to the user. This will remove the sentiment bias present in the search engine. Now, the user is provided with both the positive and negative aspects for the fruit Mango.



Mango: Nutrition, Health Benefits and How to Eat It - Healthline **Positive**
https://www.healthline.com/nutrition/mango ▾
Dec 17, 2018 - In fact, studies link **mango** and its nutrients to health benefits, such as improved ... **Mango** is a **good** source of immune-boosting nutrients.

The health benefits of mango | BBC Good Food **Positive**
https://www.bbcgoodfood.com/howto/guide/health-benefits-mango ▾
Aug 30, 2018 - **Mango** is a low-calorie fruit that is high in fibre, and is a great source of vitamins A and C. It also contains folate, B6, iron and a little calcium, zinc and vitamin E. **Mangoes** are a **good** source of antioxidants, containing certain phytochemicals such as gallotannins and mangiferin which have been studied for their ...

Mangoes: Health benefits, nutrition, recipes - Medical News To **Positive**
https://www.medicalnewstoday.com/articles/275921.php ▾
Aug 22, 2017 - Once the most widely consumed fruit in the world, **mangoes** have a ... Find recipes and information about the risks and **health** benefits of **mangoes** here. ... than to concentrate on individual foods as the key to **good health**.

15 Health Benefits of Eating Mangoes - Times of India **Positive**
https://timesofindia.indiatimes.com › Lifestyle › Health & Fitness Tips › Diet Tips ▾
Nov 17, 2017 - **Mangoes** are also loaded with vitamin A, making it a perfect fruit to improve eye sight. It also prevents night blindness and dry eyes. The enzymes in **mangoes** help in breaking down protein content in the body. Enriched with fibre, **mangoes** aid good digestion and prevents many stomach related diseases.

8 Incredible Benefits of Mangoes, The King of Fruits - NDTV F **Positive**
https://food.ndtv.com/.../8-incredible-benefits-of-mangoes-the-king-of-fruits-18... ▾
Jul 19, 2018 - According to the book 'Healing Foods', mango flesh contains prebiotic dietary fibre, which helps feed **good** bacteria in the gut. **Healthy** gut is ...

11 Surprising Benefits of Mangos | Organic Facts **Positive**
https://www.organicfacts.net › Fruits ▾
Jump to **Useful** in Pregnancy - **Mangos** are **beneficial** for pregnant women since they fulfill ... to be the delight of your day, more than just its **health** benefits.

Is a Mango a Day Bad for You? - Woman **Positive**
https://woman.thenest.com › Diet and Nutrition › Healthy Eating ▾
Mangoes are so sweet, **you** might think **you** are cheating on your diet when **you** eat one. Don't worry – a **mango** fits into a **healthy** diet plan without guilt.

Mangoes and blood sugar: Cholesterol, regulation, and obesity **Negative**
https://www.medicalnewstoday.com/articles/259753.php ▾
Sep 20, 2018 - **Mangoes** and cholesterol. **Mango** can offer nutritional benefits for most people if they consume it in moderation. If cholesterol builds up in the body, it can block the arteries and other blood vessels. This can cause heart disease, a stroke, or a heart attack.
Mangoes and cholesterol · Blood sugar regulation · Links to obesity

What are the side effects of eating too many mangoes? - Quora **Negative**
https://www.quora.com/What-are-the-side-effects-of-eating-too-many-mangoes
Jan 17, 2018 - It has a number of health benefits like, cholesterol in che... ... Over consumption of mango, for the people affected by Arthritis and sinusitis, can be very bad.
What are the disadvantages of eating **mango**? Jun 23, 2018
Is eating the skin of a **mango bad for you**? Dec 14, 2017
What are the side effects of eating an excess of ripe **mangoes**? Jun 10, 2017
Are **mangoes harmful to health**? I eat 4 to 6 mangoes daily, will it ... Jun 6, 2016
More results from www.quora.com

7 fruits you should be eating and 7 you shouldn't - Mashed **Negative**
https://www.mashed.com/69014/fruits-you-should-shouldnt-be-eating/ ▾
And with some fruits, the drawbacks are actually worse than the benefits. ... **you** may want to stay away from **mango**, especially if you're trying to lose weight.

How too much mangoes can affect your health? - HTV **Negative**
https://htv.com.pk › Health ▾
Nov 8, 2018 - And that tempts us to eat too much **mangoes** which has some side effects too. ... Being a rich source of sugar, it is considered **bad** for people on ...

Can mangoes make you fat? Every good, bad and tasty side of **Negative**
https://www.hindustantimes.com/...mangoes...you...bad.../story-2Xw9jDOsjpRzpL0R... ▾
Apr 25, 2018 - If there's one thing that makes the punishing Indian summers bearable, it's **mangoes**. No other fruit compares. And it's so versatile - you can ...

Figure 23: Final search result produced by the end-to-end model.

Table 7: Summary of results for individual components

Method	Use	Dataset	Result
Document-Level Sentiment Analyzer	Identify sentiment polarity of summarized documents	IMDB Movie Review Dataset	Accuracy: 91
Aspect-Level Sentiment Analyzer	Identify sentiment polarity of input query	Yelp Restaurant and Amazon Laptop Review Dataset	Accuracy: 82.32 F-Score: 77
Sentiment Style Transfer	Change the sentiment polarity of input sentence	Amazon Review Dataset	Sentiment classification: 88.7 BLEU: 8

Chapter 5:
Conclusion and Future Work

In this project, a novel end-to-end deep learning system has been developed to eliminate the sentiment bias present in the search engine results. This system comprises of three models. First is the document level sentiment analyzer that classifies the search query results based on their sentiment polarity. This is done in parallel with the aspect level sentiment analyzer that generates the sentiment polarity of the search query itself. Using the polarity of the search query another query with the same semantic meaning but an opposite sentiment polarity is generated by the sentiment style transfer model. Therefore, by using queries of both positive and negative sentiment the system produces two sets of search results one containing documents only with an overall positive polarity and the other containing documents with an overall negative polarity.

The document level sentiment analyzer is a deep learning model based on Long-short term memory (LSTM), trained on the IMDB movie reviews dataset. This model has been used to generate sentiment polarity for the documents taken from the search query results. This polarity is used to demonstrate sentiment bias present in the search engine results. The document level sentiment analyzer achieves an accuracy of 91% in the IMDB test set.

The aspect-level sentiment analyser is a deep memory network made of auxiliary memory and attention trained on the Yelp restaurant review dataset. This model has been used to generate the sentiment polarity of the user search query. The polarity is then used to inform the user about the sentiment of the search query. The search query along with its polarity is fed into the sentiment style-transfer model which generates a similar query, but with an opposite sentiment. This new query is used to obtain documents of the opposite sentiment. The sentiment style-transfer is a bi-directional LSTM model trained on IMDB movie review dataset.

In the future, the overall system can be improved by enhancing the individual components because of its modular design. The document level sentiment analyzer can be improved by incorporating attention mechanism to retain the relevant information throughout the system. Moreover, the model can be trained on a dataset made of search engine results instead of IMDB movie reviews for better results. Currently, the sentiment style-transfer model is trained only on unparallel data, which results in lower BLEU score, which can be alleviate by using a dataset made of parallel data. Also, the dataset used for this project restricts it only to food items and this can be extended to include wide range of queries related to famous personalities, products, political agendas and much more.

References

[1] Caroline M. Eastman, Bernard J. Jansen, Coverage, relevance, and ranking: The impact of query operators on Web search engine results, *ACM Transactions on Information Systems (TOIS)*, v.21 n.4, p.383-411, October 2003

[2] Sergiu Chelaru, Ismail Sengor Altingovde, Stefan Siersdorfer, Wolfgang Nejdl, Analyzing, Detecting, and Exploiting Sentiment in Web Queries, *ACM Transactions on the Web (TWEB)*, v.8 n.1, p.1-28, December 2013

[3] Na, J., & Thet, T. Effectiveness of web search results for genre and sentiment classification. *Journal of Information Science.*, 35(6), 709-726. 2009

[4] Kaur, Mangat, & Nidhi. A Survey of Sentiment Analysis techniques. *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)). IEEE*. 2017

[5] Terazawa, Y., Shiramatsu, S., Ozono, T., & Shintani, T. Sentiment Polarity Analysis for Generating Search Result Snippets based on Paragraph Vector *2015 IIAI 4th International Congress on Advanced Applied Informatics (IIAI-AAI). IEEE*. 2015

[6] Upadhyay, Shreya, Pant, Vishal, Bhasin, Shivansh, Pattanshetti, Mahantesh K, Articulating the Construction of a Web Scraper for Massive Data Extraction, *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). Piscataway, NJ*. 2017

[7] Glez-Peña, Daniel, Anália Lourenço, Hugo López-Fernández, Miguel Reboiro-Jato, and Florentino Fdez-Riverola. "Web scraping technologies in an API world." *Briefings in bioinformatics 15*, no. 5 (2014): 788-797

[8] S. Subramanian, S. Rajeswar, F. Dutil, C. Pal, and A. Courville, "Adversarial Generation of Natural Language," *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017.

[9] Maria Larsson, Amanda Nilsson "Manifold traversal for reversing the sentiment of text", *University of Gothenburg, Gothenburg, Sweden*. 2017

[10] M. Tsytsarau, and T. Palpanas, "Survey on mining subjective data on the web," *Data Mining Knowledge Discovery*, vol. 24, no. 3, pp. 478–514, 2012.

[11] Schouten, K., & Frasincar, F. Survey on Aspect-Level Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering*., 28(3), 813-830, 2016.

[12] Madhoushi, Zohreh, Abdul Razak Hamdan, and Suhaila Zainudin. Sentiment analysis techniques in recent works. *2015 Science and Information Conference (SAI). 2015 Science and Information Conference (SAI). IEEE*, 2015.

[13] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: *EMNLP*. Volume 14. (2014) 1532–154

[14] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, Antoine Bordes "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data", *Conference on Empirical Methods in Natural Language Processing.* 2017.

[15] Bansal, B., & Srivastava, S. (2018). Sentiment classification of online consumer reviews using word vector representations. *Procedia Computer Science*, 132, 1147-1153.

[16] Rao, G., Huang, W., Feng, Z., & Cong, Q. (2018). LSTM with sentence representations for document-level sentiment classification. *Neurocomputing*, 308, 49-57.

[17] Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. *ArXiv.org*, May 28, 2002.

[19] Hemmatian, F., & Mohammad, K. S. (2017). A survey on classification techniques for opinion mining and sentiment analysis. *The Artificial Intelligence Review*, 1-51.

[20] Chen, Gangbao, Zhang, Qinglin, Di Chen, Xiao, Jing, Mao, Zhi-Hong, Suzumura, Toyotaro, & Zhang, Liang-Jie. (2018). A Pair-Wise Method for Aspect-Based Sentiment Analysis. *Lecture Notes in Computer Science*., 10971, 2111-29.

[21] Dabholkar, Salil, Patadia, Yuvraj, Dsilva, Prajyoti, & Akila, V. Automatic Document Summarization using Sentiment Analysis (2016). *Proceedings of the International Conference on Informatics and Analytics - ICIA-16 (Vol. 25, The International Conference). New York NY: ACM Press.*

[22] K. Sundaramoorthy, R. Durga, S. Nagadarshini, "NewsOne—An Aggregation System for News Using Web Scraping Method", *Technical Advancements in Computers and Communications (ICTACC) 2017 International Conference on*, pp. 136-140, 2017.

[23] Slamet, Andrian, Maylawati, Suhendar, Darmalaksana, & Ramdhani. (2018). Web Scraping and Naïve Bayes Classification for Job Search Engine. *IOP Conference Series*., 288, 2018.

[24] Zhu, P., & Qian, T. Enhanced Aspect Level Sentiment Classification with Auxiliary Memory. *Conference on Computational Linguistics COLING (2018).*

[25] J. Li, R. Jia, H. He, and P. Liang. 2018. Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer. *ArXiv e-prints*

[26] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, Alexander J. Smola. Stacked Attention Networks for Image Question Answering. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016).*

[27] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu.2016a. Effective LSTMs for target-dependent sentiment classification. *Conference on Computational Linguistics (COLING 2016).*

[28] Yequan Wang, Minlie Huang, Li Zhao, and Xiaoyan Zhu. 2016. Attention-based LSTM for aspect-level sentiment classification. *Empirical Methods in Natural Language Processing (EMNLP 2016).*

[30] Zohreh Madhoushi, Abdul Razak Hamdan, Suhaila Zainudin. Sentiment analysis techniques in recent works. *2015 Science and Information Conference (SAI)*, pp. 288-291, 2015.

[31] Dong, Li & Wei, Furu & Yin, Yichun & Zhou, Ming & Xu, Ke. Splusplus: A Feature-Rich Two-stage Classifier for Sentiment Analysis of Tweets. *Proceedings of the 9th International Workshop on Semantic Evaluation* pp. 515-519, 2015.

[32] Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. Learning Word Vectors for Sentiment Analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* pp. 142-150, 2011.

[33] Hu, Zhiting, Zichao Yang, Xiaodan Liang, Ruslan R. Salakhutdinov and Eric P. Xing. Toward Controlled Generation of Text. *International Conference on Machine Learning,* 2017.

[34] D. Jurafsky, J. H. Martin, Speech and Language Processing 3rd edition. *Prentice Hall publications,* pp. 63-79, 2018.

Appendix

Web Scraper:

```
from textblob import TextBlob
import requests
import csv
from bs4 import BeautifulSoup
from urllib.request import urlopen
from newspaper import Article
import nltk
import re
from string import digits

nltk.download('punkt')


class Analysis:
        def __init__(self, term):
            self.term = term

            self.url =
'https://www.google.com/search?q={0}'.format(self.term)
            response = requests.get(self.url)
            return response

        def run(self):
            response = requests.get(self.url)
            soup = BeautifulSoup(response.text, 'html.parser')
            get_details = soup.find_all("div", attrs={"class":
"g"})
            final_data = []
            for details in get_details:
                link = details.find_all("h3")
                # links = ""
                for mdetails in link:
                    links = mdetails.find_all("a")
                    lmk = ""
                    for lnk in links:
                        lmk = lnk.get("href")[7:].split("&")
                        final_data.append(lmk[0])
            with open('article_file.csv', mode='w') as
article_file:
                article_writer = csv.writer(article_file,
delimiter=',', quotechar = '"', quoting=csv.QUOTE_MINIMAL)
                article_writer.writerow(['Query', 'URL',
'Article'])
                for article_link in final_data:
                    try:
                        page_data = Article(article_link,
language="en")
```

```
                        # To download the Article
                        page_data.download()
                        # To parse the article
                        page_data.parse()
                        # To perform natural language
processing ie..nlp
                        page_data.nlp()
                        clean_text = page_data.text
                        # clean_text =
clean_text.translate(None, digits)
                        clean_text = clean_text.replace('\n',
' ')
                        print(clean_text)
                        article_writer.writerow([self.term,
article_link, clean_text])
                    except:
                        continue

a = Analysis('apple is good for health')
a.run()
```

Text Summarization

```
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx
import nltk
import time


def read_article(file_name):
    file = open(file_name,  encoding="utf8")
    filedata = file.readlines()
    article = filedata[0].split(". ")
    sentences = []

    for sentence in article:
        print(sentence)
        sentences.append(sentence.replace("[^a-zA-Z]", "
").split(" "))
    sentences.pop()

    return sentences


def sentence_similarity(sent1, sent2, stopwords=None):
    if stopwords is None:
        stopwords = []
    sent1 = [w.lower() for w in sent1]
    sent2 = [w.lower() for w in sent2]

    all_words = list(set(sent1 + sent2))

    vector1 = [0] * len(all_words)
    vector2 = [0] * len(all_words)

    # build the vector for the first sentence
    for w in sent1:
        if w in stopwords:
            continue
        vector1[all_words.index(w)] += 1

    # build the vector for the second sentence
    for w in sent2:
        if w in stopwords:
            continue
        vector2[all_words.index(w)] += 1
    return 1 - cosine_distance(vector1, vector2)


def build_similarity_matrix(sentences, stop_words):
    # Create an empty similarity matrix
    similarity_matrix = np.zeros((len(sentences),
len(sentences)))

    for idx1 in range(len(sentences)):
        for idx2 in range(len(sentences)):
            if idx1 == idx2:  # ignore if both are same
sentences
                continue
            similarity_matrix[idx1][idx2] =
sentence_similarity(sentences[idx1], sentences[idx2],
stop_words)
    return similarity_matrix


def generate_summary(file_name, top_n=5, aspect_term =
'apple'):
    stop_words = stopwords.words('english')
    summarize_text = []

    # Step 1 - Read text and tokenize
    sentences = read_article(file_name)

    # Step 2 - Generate Similarity Matrix across sentences
    sentence_similarity_matrix =
build_similarity_matrix(sentences, stop_words)

    # Step 3 - Rank sentences in similarity matrix
    sentence_similarity_graph =
nx.from_numpy_array(sentence_similarity_matrix)
    scores = nx.pagerank(sentence_similarity_graph)

    # Step 4 - Sort the rank and pick top sentences
    ranked_sentence = sorted(((scores[i], s) for i, s in
enumerate(sentences)), reverse=True)
    print("Indexes of top ranked_sentence order are ",
ranked_sentence)

    for i in range(top_n):
        summarize_text.append(" ".join(ranked_sentence[i][1]))

    final_sentence = []
    for sen in ranked_sentence:
        if aspect_term in sen:
            final_sentence.append(sen)


    # Step 5 - Off course, output the summarize text
    print("Summarize Text: \n", ". ".join(summarize_text))


def main():
    nltk.download('stopwords')
    time.sleep(3)
    generate_summary("article.txt")


if __name__ == '__main__':
    main()
```

Document Level Sentiment Analyzer

```
import pandas as pd
import torch.nn as nn
import matplotlib.pyplot as plt
import numpy as np
from sentiment_model_define_class import SentimentLSTM

from sklearn.model_selection import train_test_split

from string import punctuation
from collections import Counter
import torch
from torch.utils.data import DataLoader, TensorDataset
import bcolz
import pickle
```

```
def pad_features(reviews_int, seq_length):
    ''' Return features of review_ints, where each review is
padded with 0's or truncated to the input seq_length.
    '''
    features = np.zeros((len(reviews_int), seq_length),
dtype=int)

    for i, review in enumerate(reviews_int):
        review_len = len(review)

        if review_len <= seq_length:
            zeroes = list(np.zeros(seq_length - review_len))
            new = zeroes + review
        elif review_len > seq_length:
            new = review[0:seq_length]

        features[i, :] = np.array(new)
    return features


def accuracy(output_values, size, correct_labels):
    num_correct_values = 0
    # convert output probabilities to predicted class (0 or 1)
    prediction = torch.round(output_values.squeeze())  #
rounds to the nearest integer

    # compare predictions to true label
    correct_values =
prediction.eq(correct_labels.float().view_as(prediction))
    # print(correct_tensor)
    correct_values = np.squeeze(correct_values.numpy()) if not
train_on_gpu else np.squeeze(
        correct_values.cpu().numpy())
    num_correct_values += np.sum(correct_values)
    # accuracy over all test data
    return num_correct_values / size


data = pd.read_csv('imdb_tr.csv',names=['text', 'polarity'],
encoding='latin-1')
objects = ('Positive', 'Negative')
train_on_gpu = True
data_int = data['polarity'].value_counts().tolist()
x = np.arange(2)
print(data_int)
Gender=['Train','Validation','Test']

print(data['text'].shape)
reviews = data['text'].tolist()
```

```
# print(reviews[1])
for index, item in enumerate(reviews):
    reviews[index] = item.lower()
    reviews[index] = ''.join([c for c in reviews[index] if c
not in punctuation])
# print(reviews[1])
print ('Number of reviews :', len(reviews))

all_text2 = ' '.join(reviews)
print(len(all_text2))
# create a list of words
words = all_text2.split()
# Count all the words using Counter Method
count_words = Counter(words)


total_words = len(words)
sorted_words = count_words.most_common(total_words)
print (sorted_words[1])
vectors = bcolz.open(f'6B.50.dat')[:]
words1 = pickle.load(open(f'6B.50_words.pkl', 'rb'))
word2idx = pickle.load(open(f'6B.50_idx.pkl', 'rb'))

glove = {w: vectors[word2idx[w]] for w in words1}

matrix_len = len(words)
weights_matrix = np.zeros((matrix_len, 50))
words_found = 0

for i, word in enumerate(words):
    try:
        weights_matrix[i] = glove[word]
        words_found += 1
    except KeyError:
        weights_matrix[i] = np.random.normal(scale=0.6,
size=(50, ))
vocab_to_int = {w:i+1 for i, (w,c) in enumerate(sorted_words)}
reviews_int = []
for review in reviews:
    r = [vocab_to_int[w] for w in review.split()]
    reviews_int.append(r)
encoded_labels = np.asarray(data['polarity'])
print(type(encoded_labels))
reviews_len = [len(x)+300 for x in reviews_int]
reviews_int = [reviews_int[i] for i, l in
enumerate(reviews_len) if l>0 ]
encoded_labels = [encoded_labels[i] for i, l in
enumerate(reviews_len) if l> 0 ]
encoded_labels = np.asarray(encoded_labels)
print(type(encoded_labels))
len_feat = 25000
```

```
features = pad_features(reviews_int, 400)
split_frac = 0.6
train_x = features[0:int(split_frac*len_feat)]
train_y = encoded_labels[0:int(split_frac*len_feat)]
print(type(train_y[0]))
remaining_x = features[int(split_frac*len_feat):]
remaining_y = encoded_labels[int(split_frac*len_feat):]
valid_x = remaining_x[0:int(len(remaining_x)*0.5)]
valid_y = remaining_y[0:int(len(remaining_y)*0.5)]
test_x = remaining_x[int(len(remaining_x)*0.5):]
test_y = remaining_y[int(len(remaining_y)*0.5):]
# create Tensor datasets
train_data = TensorDataset(torch.from_numpy(train_x),
torch.from_numpy(train_y))
valid_data = TensorDataset(torch.from_numpy(valid_x),
torch.from_numpy(valid_y))
test_data = TensorDataset(torch.from_numpy(test_x),
torch.from_numpy(test_y))
# dataloaders
batch_size = 50
# make sure to SHUFFLE your data
train_loader = DataLoader(train_data, shuffle=True,
batch_size=batch_size)
valid_loader = DataLoader(valid_data, shuffle=True,
batch_size=batch_size)
test_loader = DataLoader(test_data, shuffle=True,
batch_size=batch_size)
# obtain one batch of training data
dataiter = iter(train_loader)
sample_x, sample_y = dataiter.next()
vocab_size = len(vocab_to_int)+1 # +1 for the 0 padding
output_size = 1
embedding_dim = 400
hidden_dim = 256
n_layers = 2
model = SentimentLSTM(vocab_size, output_size, embedding_dim,
hidden_dim, n_layers)
print(model)

# loss and optimization functions
lr=0.001

criterion = nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=lr)


# training params

epochs = 4 # 3-4 is approx where I noticed the validation loss
stop decreasing
```

```
counter = 0
print_every = 100
clip=5 # gradient clipping

# move model to GPU, if available
if(train_on_gpu):
    model.cuda()
    criterion.cuda()
train_accuracy = []
train_losses = []
val_lossess = []
model.train()
# train for some number of epochs
for e in range(epochs):
    # initialize hidden state
    h = model.init_hidden(batch_size)
    # batch loop
    for inputs, labels in train_loader:
        num_correct = 0
        counter += 1
        inputs = inputs.type(torch.LongTensor)
        if(train_on_gpu):
            inputs, labels = inputs.cuda(), labels.cuda()

        # Creating new variables for the hidden state, 
otherwise
        # we'd backprop through the entire training history
        h = tuple([each.data for each in h])
        # zero accumulated gradients
        model.zero_grad()

        # get the output from the model
        output, h = model(inputs, h)

        # calculate the loss and perform backprop
        loss = criterion(output.squeeze(), labels.float())
        loss.backward()
        # `clip_grad_norm` helps prevent the exploding 
gradient problem in RNNs / LSTMs.
        nn.utils.clip_grad_norm_(model.parameters(), clip)
        optimizer.step()

        # loss stats
        if counter % print_every == 0:
            train_acc = accuracy(output, batch_size, labels)
            train_accuracy.append(train_acc)
            print("Train accuracy: {:.3f}".format(train_acc))
            # Get validation loss
            val_h = model.init_hidden(batch_size)
            val_losses = []
            model.eval()
            for inputs, labels in valid_loader:
                # Creating new variables for the hidden state,
otherwise
                # we'd backprop through the entire training
history
                val_h = tuple([each.data for each in val_h])
                inputs = inputs.type(torch.LongTensor)
                if(train_on_gpu):
                    inputs, labels = inputs.cuda(),
labels.cuda()

                output, val_h = model(inputs, val_h)
                val_loss = criterion(output, labels.float())

                val_losses.append(val_loss.item())

            model.train()
            #validation_acc = accuracy(output, batch_size,
labels)
           # print("Train accuracy:
{:.3f}".format(validation_acc))
            print("Epoch: {}/{}...".format(e+1, epochs),
                  "Step: {}...".format(counter),
                  "Loss: {:.6f}...".format(loss.item()),
                  "Val Loss:
{:.6f}".format(np.mean(val_losses)))
            train_losses.append(loss.item())
            val_lossess.append(np.mean(val_losses))
            # bar_list = plt.plot(list(range(1,
len(train_losses)+1)), train_losses)
            # # plt.xticks(x, objects)
            # plt.title('Sentiment Polarity count')
            # plt.xlabel('Sentiment Polarity')
            # plt.ylabel('Review Count')
            # plt.show()
# Get test data loss and accuracy
plt.plot(list(range(1, len(train_accuracy)+1)),
train_accuracy, label='train')
plt.plot(list(range(1, len(train_accuracy)+1)), y_values,
label='test')
#plt.xticks(x, objects)
plt.title('Accuracy Graph')
plt.xlabel('Epochs')
plt.ylabel('Loss (%)')
plt.legend()
plt.show()
test_losses = []  # track loss
num_correct = 0

# init hidden state
```

```
h = model.init_hidden(batch_size)
torch.save(model.state_dict(), 'model.ckpt')
model.eval()
# iterate over test data
for inputs, labels in test_loader:

    # Creating new variables for the hidden state, otherwise
    # we'd backprop through the entire training history
    h = tuple([each.data for each in h])

    inputs = inputs.type(torch.LongTensor)
    if (train_on_gpu):
        inputs, labels = inputs.cuda(), labels.cuda()

    # get predicted outputs
    output, h = model(inputs, h)


    # calculate loss
    test_loss = criterion(output.squeeze(), labels.float())
    test_losses.append(test_loss.item())

    # convert output probabilities to predicted class (0 or 1)
    pred = torch.round(output.squeeze())  # rounds to the
nearest integer

    # compare predictions to true label
    correct_tensor = pred.eq(labels.float().view_as(pred))
    correct = np.squeeze(correct_tensor.numpy()) if not
train_on_gpu else np.squeeze(correct_tensor.cpu().numpy())
    num_correct += np.sum(correct)

# -- stats! -- ##
# avg test loss
print("Test loss: {:.3f}".format(np.mean(test_losses)))

# accuracy over all test data
test_acc = num_correct / len(test_loader.dataset)
print("Test accuracy: {:.3f}".format(test_acc))
```