AN INTERACTIVE VISUALIZATION SYSTEM FOR A MULTI-LEVEL VIEW OF OPINION MINING RESULTS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Karan Chitkara

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Software Engineering

October 2014

Fargo, North Dakota

North Dakota State University
Graduate School

Title

An interactive visualization system for a multi-level view of opinion mining results

By

Karan Chitkara

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Wei Jin
Chair

Gursimran Walia

Na Gong

Approved:

10/24/2014	Brian Slator
Date	Department Chair

ABSTRACT

Products are purchased and sold on a daily basis and people tend to critique on products they purchase. Those who want to buy a product will read reviews on that product given by others before buying; likewise those who have already bought a product will write a review on it. This paper presents a technique for visualizing data that comes from reviews given online for different products. My contribution to this project is to create a tool and process the tagged files generated with the help of machine learning.

This project also focuses on the implementation of Semantic matching which reduces redundancy by grouping similar data together. Semantic matching helps put all the synonyms of the data together. Implementation of Semantic matching is supported by the implementation of error correction technique. Error correction improves data quality by correcting spelling mistakes made by people while writing reviews.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Wei Jin for the opportunity to work with her. She taught me everything about opinion mining and gave me real data which helped me a lot to test my application. Under her guidance and support I was able to improve my tool both qualitatively and quantitatively. I also express my sincere regards and gratitude to the committee members Dr. Na Gong and Dr. Gursimran Walia.

Lastly I would like to thank my friends Aakanksha, Akshay and Saumya for their motivation and inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1. Overview

From needle to a house, products are bought and sold on daily basis. With an increasing number of reviews given by different people on different products it becomes hard to judge which product is best rated. Sometimes a rating of four or 5 star for a product is just not enough and we need to know more than that.

While buying a product one can go online and read all the reviews given by different people for a particular product and make a decision of whether to buy it or not. But this process of reading all the reviews is very time consuming and you might not get the information you need or the information would be later in the reviews that it would take hours to get to that piece of information that you desire. This was the biggest motivation of opinion mining.

My contribution to this paper includes: (1) creating a tool that parses all the XML tagged review files and provide the user with a visual representation of all the reviews in a tree and grid format; (2) applying semantic matching technique to group all the similar keywords along with the review sentences together; (3) complementing Semantic Matching by implementing Error Correction technique which will correct the misspelled keywords so that all the relevant information stays together.

In their work on OpinionMiner, Wei et al. used machine learning to create unique XML tags for the reviews [23]. This information was used as a base to create this tool which can effectively extract all the information from these XML tags and segregate them as positive and negative reviews. It can organize the product reviews of any given product in turn helping the customer make a decision while making a purchase. This application follows basic client-server

communication where the client is the WFA user interface; server is the code which runs it and the database.

Figure 1 shows the interaction among various components showing how client interacts with interface and database.

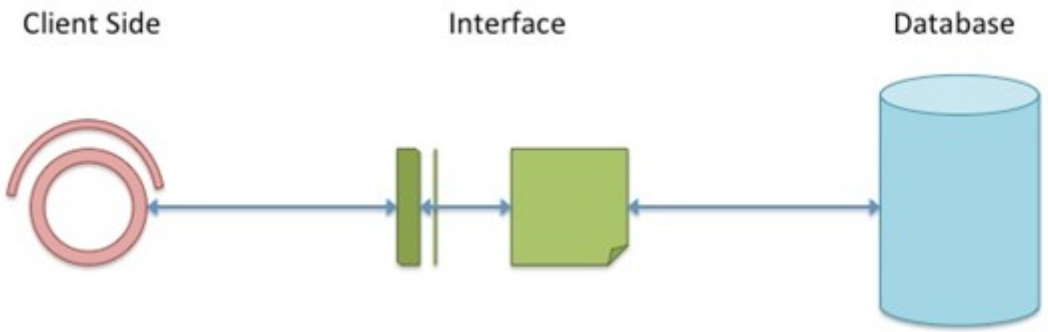

Figure 1. Interaction among various components

The rest of this paper is organized as follows: First chapter gives an overview and description of my contribution to this paper and the problem statement. Second chapter discusses the background of why the application is developed and the existing work which has been done in regards to this application. Third chapter discusses the Application Design which is explained through Unified Modeling Language (UML) which creates visual models to aid understanding of process flow. This chapter also discusses the implementation of database used to store the application data and the way it increases the application's performance. Fourth chapter discusses the implementation of the application by giving an overview of how it was implemented using WFA and how semantic matching was implemented in the application using an external website. Fifth chapter discusses the performance of the application and how it increases as the application runs increase. Sixth chapter concludes with why this application is needed and how is it helpful to many around us. This chapter also discusses some future work.

1.2. Problem Statement

With the help of machine learning all the reviews given for a particular product were processed and tagged files were generated out of them, each file representing an individual review that have XML tags in them. Each tag is responsible for carrying a particular kind of information including, if a review was a positive one or a negative one. After each review is tagged using machine learning the next step was to process all the tags in a review for all the reviews for a particular product and represent them visually in a display that is easy to understand. After looking at it anyone should be able to generate a conclusion about the product being sold or purchased.

The tool serves three major purposes:

1. The user won't have to read all the reviews in order to come to a conclusion whether to buy a product or not as it is very time consuming.
2. The user will be able to see all the positive and negative aspects for every feature mentioned in the reviews and, the user does not have to remember them while comparing with other features of the same product.
3. This tool eliminates information that is not required i.e. the user can instantly see the total number of positive and negative reviews given for any feature for that product and if wanted can also read the review sentences pertaining to that feature.

Most of the times when the user goes online to read the reviews for a particular product given by different people, the user can get confused after reading all the reviews as many talk about the same thing but with different words. This application also takes care of that by implementing Semantic Matching (SM) which will be explained later in the Application Logic section.

The most important issue that was addressed while creating this application was to save user's time as much as we can by visually representing the data in such a way that the user does not have to read all the reviews before drawing a conclusion.

In order to achieve this, after processing the XML tagged files the data is represented in a tree format which can be expanded or collapsed to its roots/features thereby providing the user with necessary and important information.

The objective of creating a client side WF was to allow the user to be able to upload all the processed XML tagged review files for a particular product and get qualitative results. These results can help the user decide whether the product was a success or a failure.

CHAPTER 2. BACKGROUND

2.1. Overview

Over the years people have been buying products online or by going to the store. Because of many different companies selling same kind of products it can be hard to decide which company's product is better. Most of the companies give star rating on a scale of one to five for their products, five being excellent and one being poor. This does not help a consumer in deciding certain functionality i.e. if a certain feature of that product is good or bad.

Most of the companies allow people to write descriptive comments on the product, their likes and dislikes. Anyone can write comments about the product and each comment is considered as a review. For this project, each review written by customers of that product is tagged through XML tags with the help of machine learning.

Table 1 shows the basic tag set and its corresponding entities.[23]

Table 1. Basic tag set and its corresponding entities

Tag set	Corresponding Entities
<PROD_FEAT>	Feature entity
<PROD_PARTS>	Component entity
<PROD_FUNCTION>	Function entity
<OPINION_POS_EXP>	Explicit Positive Opinion Entities
<OPINION_NEG_EXP>	Explicit Negative Opinion Entities
<OPINION_POS_IMP>	Implicit Positive Opinion Entities
<OPINION_NEG_IMP>	Implicit Negative Opinion Entities
<BG>	Background Words

Table 2 shows pattern tags set and its corresponding pattern [23]

Table 2. Pattern tags set and its corresponding pattern

Pattern Tag	Corresponding Pattern
<>	Independent Entity
<-BOE>	The Beginning Component of an Entity
<-MOE>	The Middle Component of an Entity
<-EOE>	The End of an Entity

XML tags have a beginning and an ending tag and anything in between the start and the end of a tag is considered as tag's information. For example: <PROD_PARTS> battery </PROD_PARTS>, the start of the tag is <PROD_PARTS> and the end is </PROD_PARTS> and the information in between this tag is useful and is extracted by this application, so in this case the product part will be "battery".

C# was used as a design language over Java since it is a much better design language than Java. C# has a wide range of in-built functions as compared to Java which makes application development much easier. Also, it has a lot of tutorials available which further ease development in this language.

Access database was used over other databases because Access can interact very well with C#. Also, since this project involved very less tables, hence using larger databases was not required in this context. Besides, maintaining Access database is relatively easier than other RDBMS.

This application is not cross-platform because designing a cross platform application brings its own unmanageable issues or defects. Awareness of all the issues related to deploying an application on multiple platforms and effectively dealing with them was outside the scope of

this project. Since, the only platform choice made for this application was for it to be just a windows-based application, all the possible known issues were taken care of.

The software engineering design principles that were used for providing convenient access to the user are –

- Scalability – Both the application user interface and the database were scalable as they can accommodate any new keywords or any new files very well. Even when a synonym for a new keyword does not exist in the database, the application can locate the synonym online and feed it to the database for future reference.
- Reliability – The application is reliable in a way that if the database does not contain information similar to the data and the keywords uploaded to the system through files, the application will look for relevant information online and provide appropriate results to the user based on their query.
- Extensibility – The application conforms to the reusability standards since the implementation is based on tabs functionality to represent features. Hence, additions of other features can be easily implemented in forms of new tabs without disrupting previous feature functionalities.
- Performance – Performance was been taken care of as a key factor in the implementation of this application. Since, the keywords entered into the application via the uploaded files are first compared to the existing data in the database through the use of hash maps, instead of online comparisons, the performance is greatly increased. Further information on performance is covered in performance section of this paper.

2.2. Related Work

To my knowledge, the design of this system cannot be second with any other existing system with even similar functionalities. The size and width of the window selection was made such that it is neither too large nor too small. Larger window sizes or maximize window option would lead to data being scattered while much smaller window size would make data look too congested and compressed. Also, the design choice was made in accordance to the windows based application standards.

Other review comparison tools applications are mostly web-based applications, functionalities and design standards of which are based on web-application's design standards. For instance, Amazon's product review feature functionality works with product reviews and basically just combines and count all the reviews in the ratings of 1 to 5. They apply minimal to none semantic matching approaches to group all the reviews together. For example, if we compare the reviews of canon digital camera on Amazon, the reviews are based on the ratings of 1 to 5. The tool I created uses Semantic Matching with Error Correction techniques which not only helps segregate positive and negative reviews but also takes of the spelling mistakes people make while writing these reviews. Unlike the review classification from the websites like Amazon and eBay, this tool puts all the positive reviews and the negative reviews together in their own separate sections in their respective branches. This eases the review based product selection process for the customers since all the information will lie in the tree-view or grid-view based hierarchy. Even being a windows based application, this tool possesses the capability of showcasing the positive and the negative reviews to its entirety; irrespective of their number; to the customer. This in itself is a feature very different from the other web based applications since

they only highlight a few positive reviews about a product for the customers to view and hide all of the negative reviews.

Another aspect in which this tool is better than other web-based review comparison applications is its ability to separate the reviews on the basis of product features. Product reviews are first separated and put into different product feature categories and further separated into positive and negative reviews within those individual categories. For example, product reviews for canon digital camera will be first separated and put into different categories such as lens, picture quality, image resolution, optical zoom, etc. After the reviews are being separated and placed in these feature categories, there are further processed and separated into positive reviews and negative reviews. This tool ensures that all the positive reviews and negative reviews are available for the customer to view keeping in mind that these are just the review sentences and not the complete reviews. These features make this tool stand out since websites like Amazon and eBay does only highlight very few uncategorized positive reviews.

Again, since Amazon is a web-based application, amount of content and its placement on the web-page is designed according to web-based applications standards. Since, my application is strictly a window based application, it has its own content amount and placement restrictions according to the windows based application standards.

2.3. Existing Work

There have been a lot of ongoing projects for Natural Language Processing (NLP) and one of them is OpinionMiner by Wei Jin, Hung Hay Ho and Rohini K. Srihari where they talk about how to tag the reviews with XML tags discussed in the earlier section of this paper. The motivation to build this application came from this paper where they efficiently tag the reviews with relevant information on relevant tags which are processed through this application.

In their paper on Mining the Peanut Gallery, Dave et al., rated the sentence from the review to determine if it is positive or negative and scored them. They, then, decided the quality of the review on the basis of scores [24]. In this application, a similar ideology was adopted; instead, keywords were used and processed on the basis of its positivity and negativity. Also in their paper, Dave et al., inferred that since a lot of people have a tendency of writing reviews for a particular product either in blogs or on other websites. Hence, they extracted product reviews from several websites and separated them in positive and negative reviews. They took the reviews per product and could not separate them according to specific features or parts of a particular product. Since the tool I developed takes the XML tagged reviews from the research work done by Jin et al., it will accept the data from several sources as well. Besides, all the data which comes from different sources will be further sent to different product categories / product features / product parts which are not being handled by the work done by Dave et al. This is a unique functionality which is being handled by my tool. My tool has the capability of taking all the reviews and putting them in separate product features. After the data is processed by the system, all the keywords are separated in positive and negative reviews, the review sentences are displayed in tree view. Each and every feature / part of a product will display how many reviews were positive and how many were negative. It will also display positive review sentences and negative review sentences. Also, since all of the reviews are being saved in the database and can be emailed, a user can have the freedom to view these results at any given point of time.

Related work has been going on for creating client side application using VS as Integrated Development Environment (IDE), .NET as platform and C# as language. But as far as databases are concerned it depends on the application's functionality and how it will be used. The decision to choose a database actually depends on the relationship of data.

Riloff et al. used bootstrapping algorithms to exploit extraction patterns to learn about subjective nouns to be used to develop a system which can effectively differentiate subjective sentences from objective sentences [26]. They created a subjective classifier which takes subjective nouns with the help of bootstrapping algorithm and then trained a Naïve based classifier using these extracted subjective nouns. Their bootstrapping algorithm learned 1000 subjective nouns and used extraction pattern to learn subjective nouns. While they used extraction pattern, I used HTTP get request to extract the synonyms from thesaurus.com. The extracted synonyms are then further stored in the database for future use for enhancing the performance which none of the references mentioned in this paper are using.

CHAPTER 3. DESIGN

3.1. Overview

There are many ways for designing an application which aid in understanding how an application should work and, most importantly, how it actually works. If a design is good then it helps in understanding the logic, purpose and need for creating any application.

For this project in order to provide a better understanding of the application different types of diagrams have been created in Microsoft Visio which provides a better explanation by giving a visual flow of the application explaining the internal functioning as well as a high level overview of the application.

There can be different types of diagrams created in Visio but for this project a specific type of diagrams were created known as Unified Modeling Language (UML) diagrams. UML can be used to create various kinds of diagrams made but for this project we have restricted our self to only those UML diagrams which are relevant to this project and which will provide a better and clear understanding of the application.

The UML diagram used for this application is a Use Case diagram which first gives a high level design overview and, later UML diagrams provide the core logic and functionality of the application.

In order for the design to be complete, a database diagram was also created for this application to make it is easier to understand the importance of database for this project and how the database helped in achieving performance and optimization. The database diagram was also created in Visio which is also a part of UML diagrams explaining how external entity interacts with the database and how the database responds.

3.2. Use Case Diagrams

A Use Case Diagram (USD) is a diagram specifically designed to understand the flow of system and how different entities interacts with each other to meet desired goals. Use case diagram is different from other UML diagrams as it does not focus on either the timeline of the application or how an activity of an application is carried. It just focuses on behavior of each component and how it interacts with both within the boundary and from inside of the boundary to outside.

The following figure is a base use case diagram that explains the high level overview of the application. User interacts with the interface by uploading the main directory and the processed directory is uploaded automatically. The files are then sent to the server for processing and the processed data is displayed on the interface.

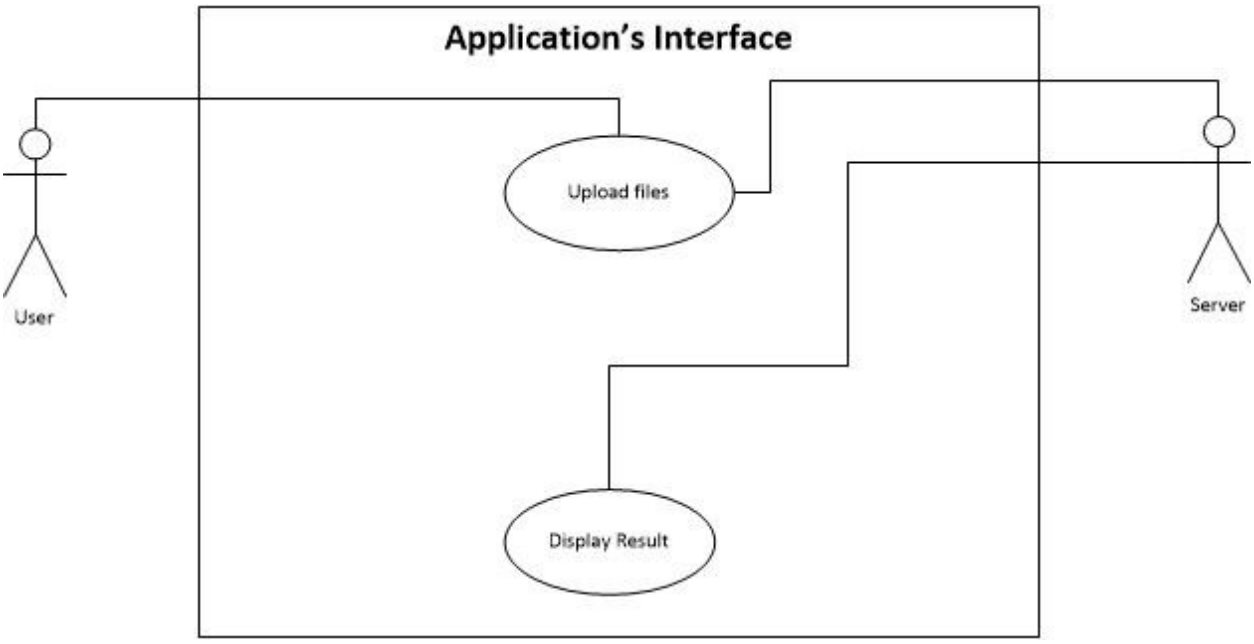

Figure 2. Use case diagram of application's interface

The following figure provides a detailed explanation of how server interacts with the interface for processing the files and generating results. The user interacts with the interface by

uploading the files. The files are then sent to the server for processing. The server has a file parser that extracts the important information from the files uploaded to generate results and also load existing information from the database if any.

If the server does not have a specific keyword found from uploaded files then it looks for that keyword over the internet on www.thesaurus.com and with the help of HTTP GET request retrieves all the related synonyms and stores them into the database. Server with all the collective information then generates results and displays it on the interface.

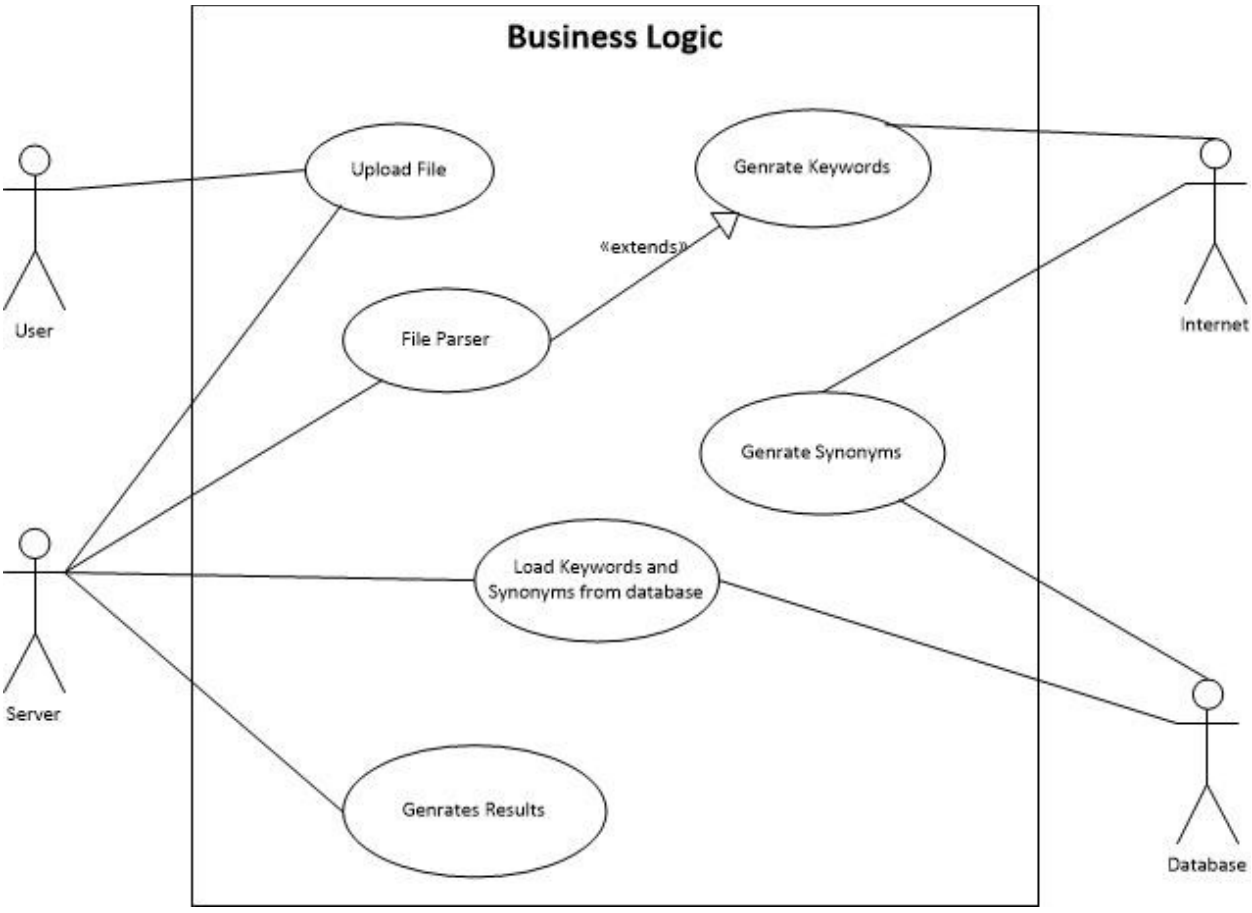

Figure 3. Use case diagram of business logic

3.3. Database

Database is an essential part of the application as it increases performance by not letting the application to go to the Internet and search for the synonym every time. Instead it looks for the synonyms in the database first.

The corrected table in the database stores the keyword and corresponding to the keyword is the corrected word. If a keyword is not found in the synonyms table then the application looks in corrected table and if it's not there then it goes to the Internet to search for synonyms.

Figure 4 shows how with the use of database the processing is optimized.

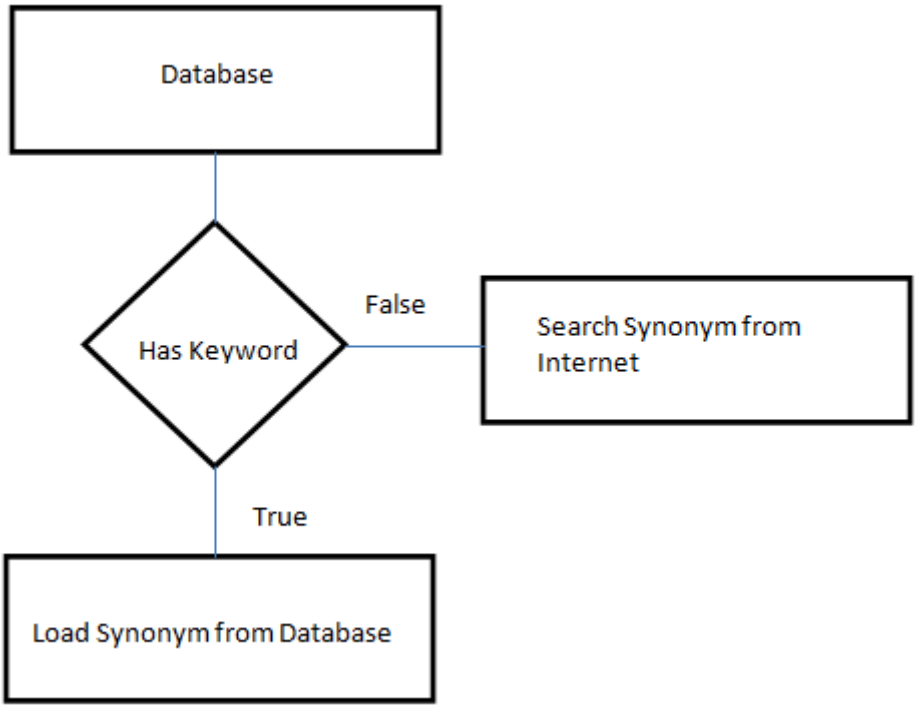

Figure 4. ER diagram

3.4. Semantic Matching

Semantic Matching (SM) is a most commonly used technique to control redundancy. While creating an application for visualization of data, the data needs to be as less redundant as possible.

A clear understanding of the concept of redundancy is a key to understand semantic matching. Data can be redundant in many forms

- Duplicate data - same data represented more than once
- Data in disguise - Something that can have more than one word to describe them. They are known as synonyms. A synonym can be a word or phrase that may mean exactly the same like some other word. For example: Photo is a synonym of Picture and vice-versa is also true.

When creating an application for visualization, it is important to not show redundant information. This is because increased reading is cumbersome and nobody wants to go through the same information over and over again.

For the purpose of this project, since it deals specifically with keywords that may have a lot of synonyms, implementation of SM deemed important as it could reduce redundancy and put all the relevant information together.

SM helps keep information in order. For example if we create two folders viz. photos and pictures respectively. Both of them have same kind of contents i.e. images and same information about images. Since in English they are both synonyms to each other and have the same meaning, this example very well explains SM in terms of its ability to help organize all the relevant information together.

A better understanding of how SM has reduced redundancy, improved readability and increased performance can be explained through the way this application works.

When SM was not implemented in the process of creating this application, then it was observed that there were many sentences that referred to about the same thing but were categorized differently. For Example: first sentence goes 'The quality of the picture taken from this camera is superb', while the second sentence says- 'The images when transferred from camera to computer looks even better'. In this example both of the sentences were referring to the same keyword 'picture'. The word 'picture' and 'image' are synonyms to each other but the tool could not recognize it before SM was implemented; it instead considered it as 2 different keywords and hence, did not group the information together.

The example clearly demonstrated the significance and importance of SM. It also explains that with the use of SM, both the sentences could be grouped under one keyword either images or picture instead of having two keywords or each keyword having one sentence under it.

Now let's consider a case where we have hundreds of keywords and thousands of sentences. If the information is not grouped properly, it can create a problem for the user as the sentences they might want to read would be distributed under different keywords. This would result in them taking more time to read all of the relevant information. Since this is visualization of data, it is important that the data is as compact as possible. This is where SM became very helpful.

Another reason behind implementing SM in this project was the presence of large amounts of data. The application would have survived without SM if the data was too small meaning few reviews for each product, but in reality there can be hundreds and thousands of online reviews for each product. Now let's consider a scenario where a product is to be launched

worldwide. In this case some may like it and some won't. Hence there would be a wide variety of reviews written for this product by different people who may have a tendency of using different words to describe the same thing. In this situation the tree of keywords would become too large if there is no SM implemented. Since, then, the information would be scattered all over under many different keywords, this application's purpose towards better understanding of the results won't be solved.

For example: Let us consider a new product that has recently came to the market and it has been few weeks. There are 10,000 total reviews given so far and out of which let's say keyword photo has 20 synonyms. It has 2500 positive reviews and 1000 negative In this situation, if there is no SM implemented then all 3500 sentences would be distributed under 20 keywords and the user would have to add all these relevant keywords by themselves which nobody would like to do. If SM is implemented which it is, in this project, then all of the 3500 sentences would come under one keyword separating positive and negative reviews.

The implementation of SM in this project is explained through an example review file which has following text of tags that shows the XML tagged review information. Each review/file has XML tags, the three important tags from which we get the keywords are <PROD_FEAT>, <PROD_PARTS> and <PROD_FUNCTION>. Any text inside the start and end of these tags are the keywords. When the keywords have been identified, the code written in C# extracts all these keywords and creates lists for storing them. If the database has any keyword then that keyword along with its synonyms are loaded in a hash table. Each keyword which was one by one extracted from the review file is then compared with the keyword and synonym from the hash table. If it already exists in the hash table then we don't look any further and proceed with the other keyword. If a keyword is not found in the hash table then the application looks for

that keyword on www.thesaurus.com and if it has any synonyms it gets those synonyms and stores them in the database along with other synonyms and keywords. We check for each keyword if it's a synonym of some other keyword coming from the review file. By doing this we only create a list of keywords that are not synonym of each other there by controlling redundancy and keeping all the relevant information together. It ensures every keyword is unique and all the sentences are grouped together.

<BG>a</BG><OPINION_POS_EXP>terrific</OPINION_POS_EXP><PROD_PARTS>camera</PROD_PARTS>

<OPINION_POS_EXP-BOE>easy</OPINION_POS_EXP-BOE><OPINION_POS_EXP-EOE>to</OPINION_POS_EXP-EOE><PROD_FEAT>use</PROD_FEAT>

<BG>no</BG><BG>more</BG><BG>wasted</BG><BG>film</BG>

<BG>i</BG><BG>love</BG><BG>everything</BG><BG>about</BG><BG>it</BG>

3.5. Error Correction

People make errors while writing English sentences. Everyone has their own way of writing words. Some use correct English grammar and some use slang to represent their opinion. Here SM alone cannot help reduce redundancy. There should be a mechanism that has a control over most of this discrepancy. There may be situations where the person writing the review accidently writes incorrect word and that happens to be the keyword in our case. In order to deal with these kinds of situations we have implemented Error Correction (EC) in our project.

In order to understand this mechanism properly lets go over the mechanism of SM again. In SM we extract all the keywords from review files and check one by one if it already exists in the database or not. The database also has a table that stores corrected words corresponding to the keywords, so if a keyword is incorrectly spelled its correction will be in the correction table.

Also, if a keyword is correctly spelled then also it would be in correction table with corrected word as the keyword itself.

If a wrongly spelled word is not found in the database then the application goes over the internet to www.thesaurus.com with that incorrect keyword. If the application gets a response that the keyword might be incorrect, then the website gives a 'did you mean' phrase suggestion followed by the keyword. This way we find the corrected word corresponding to wrongly spelled keyword. There may be situations where even the website also can't recognize the incorrectly spelled keyword and only in that situation the keyword is taken as is since the application could not recognize it as an English word.

EC technique has helped this application deal with the situations where a keyword is spelled wrongly and it does not create multiple keywords. For example: a person wrote a review with sentence that included the keyword battery and most of the reviews are talking about this product part and someone also writes in their review about battery by spelling "battery" as "bttery". In this situation if the EC technique is not implemented then the application will consider them as separate keywords and group the information accordingly. This helps in improving readability as well.

CHAPTER 4. IMPLEMENTATION

4.1. Development Overview

Over the years, software was developed either as windows stand-alone applications or browser applications. Windows stand-alone applications not only included Microsoft Office Suite, Notepad++ and NetBeans but were also extended to applications such as Skype, google chat, yahoo messenger, etc. Browser applications included websites, ranging from commercial sites to personal sites.

These browser applications were not just limited to websites; they also included web-based applications such as US Bank web-based application or the Excel Energy web-based application. These web-based applications are user-friendly and allow the user to perform a wide variety of actions and tasks.

For instance, US Bank application allows the user to securely log into a user's bank account, check account balance, pay bills, etc. Some browser applications also incorporate their own stand-alone windows based applications. Examples of these applications include Gmail and Yahoo mail which include their own messengers as part of their web-based applications; which however, may also be installed as stand-alone messenger applications on a machine.

This paper talks about the design of a Windows Form Application (WFA). As mentioned earlier the development was done in C# language using .NET as the platform and VS as the IDE. Creating an application for visualization is always a challenge as one should be wary of the size of the window. Information may become discrete if the application's window is too large or it can become too cluttered it the window is too small.

When creating a WFA there are certain things that should be kept in mind:

1. Application's window size and the appearance of the interface shouldn't be too fancy. A user may get distracted from the main context. It shouldn't be too dull either that the user has no interest in using it.
2. Reduce scrolling by providing features like expand/collapse. It gives the user the ability to manipulate display according to user's needs.
3. One other important thing is try not to have too many irrelevant redirections of displays. It can confuse a user.
4. Speed is definitely the most important of all.

All the points mentioned above are important when creating a WFA and this project has implemented all of them. While creating this application there were other things that were implemented to enhance user experience. We will walk through them in the later sections of this chapter.

4.2. Windows Form Application

WFA is a Graphical Application Programming Interface (API) through which we can access Microsoft Windows Interface (MWI). The API is a part of the .Net framework developed by Microsoft. Windows forms (WF) are built using Windows API.

An application created using WFA can only be used on Windows operating system (OS) and not on any other OS like Mac or Linux because of platform dependencies. Also that WFA uses MWI which could only be found in Windows OS.

The purpose of creating an application for this project in WFA is to build an interactive and rich interface which takes all the parameters into account such as its length, width, color,

application logo, etc. An application is built in WFA ensures that the processing is more towards client side than the server side.

Most of the time when an application is built using WFA the intent is to perform most of the processing on the client side and send the response to the server so that the server does not get crowded with all the client processing. For this project client, server and database all are on client system.

Usually a WFA on the client side does all the work by filling in the form and all the data is transported to the server using the Internet. This relationship of client and server is called Client-Server Architecture.

In client-server architecture, a client sends request to the server using internet, then the server depending on the application sends a request to the database. The database then sends the response back to the server and then the server finally sends a response to the client with information rendered from the database. For this project all client, server and database resides in the same location.

The most common languages that are used for creating WFA are C# and Visual Basic. For this project the application was built using C# language .NET framework.

The three main components for building this project are:

1. **C Sharp (#) –** C# is an object oriented language developed by Microsoft. It first appeared in 2000 and was used alongside .Net to implement the concepts of object oriented programming also known as OOPS.

Memory can also be managed using c# with the help of 'using' keyword which will free up the memory as soon as the work of that variable or a function is done and is known that it

won't be used anymore. By freeing up the memory during runtime, the application increases performance as more memory can be utilized as the application runs.

There are many inbuilt functions that the programmer does not need to define all the time which helps in saving time of the developer developing the application data structures. Linked lists, stacks and queues are some of the examples.

2. **.Net –** It is a framework developed by Microsoft that was first released on 13 February 2002. This framework provides interoperability across several programming languages. It makes the development easy by providing a user interface that can be manipulated by a user according to the user's requirement.

Applications built through this framework use Common Language Runtime (CLR) which is a virtual machine that protects memory by managing the memory and also by handling exceptions.

Memory management is one of the biggest advantages of using .Net framework as the developer developing the application does not have to worry about freeing up the memory since the automatic garbage collector frees up the memory periodically. It works by having 2 separate threads; one for the application and the other for the garbage collector which does not interfere with the main application thread.

3. **Visual Studio (VS) –** It is an Integrated Development Environment (IDE) developed by Microsoft in C# programming language. This application was built in Visual Studio 2012. There are, however, newer versions for VS available in the market, the latest version among them is Visual Studio 2013.

Developers use this IDE to create many kinds of applications such as web-based, WFA etc. OS required to run VS is Windows OS. Apart from developing applications in C#, VS

allows the developer to work with many different languages such as Hyper Text Markup Language (HTML), C, C++, VB.Net, F# and many more.

Code written in VS is managed so that everything is grouped and has a specific location. For example, if we create a WF using VS then the code can be viewed under Solution Explorer tab on the right and by double clicking a file can be viewed. Every form file will have two additional files-

a. **CS file –** This file is responsible for all the code that runs in the backend and which manipulates the form by having triggers and events. This is the file where C# code is written corresponding to the form. For this project most of the development was done in this file.
b. **Designer file –** This file manages the design part of the form i.e. it will have all the buttons, textbox or any other GUI components.

4.3. Application Logic

The user interacts with the interface by uploading the main directory which will contain two file types .txt and .pos for each review. File with .txt extension would contain unprocessed reviews which are simple plain English sentences, whereas, file with .pos extension would contain processed reviews. Processed reviews correspond to the reviews processed with help of machine learning. Since we will not require processed file for this project, we won't be uploading it.

When the user uploads the main directory then the processed directory is automatically uploaded provided that both the directories have same name and the processed directory would have _processed at the end of the directory name to distinguish among the two directories.

The processed directory will also have .txt and .fo file types, both of them combined would represent a single review. File with extension .txt will contain XML processed reviews and file with extension .fo file will contain the feature orientation of the review i.e. what part of review is positive and what part is negative.

The application logic is divided into three major layers:

1. **Interface –** It refers to the GUI created in VS on which the user interacts. The interface was made as simple as possible so that anyone who has the access to the application can easily generate results. The visualizations generated from the results are easy to understand too.

The interface was built using .NET framework in VS and all the attributes like length, width, color etc. were kept in mind while creating the application's interface. Since the application is for visualization of data, all the dimensions of the interface like length, width and height were given importance and therefore lot of testing was done in order for the interface to occupy the appropriate space.

If the interface was too large then the focus of the user will be diverted as information would be scattered all over the screen. And, if the interface was too small then the user would have to do a lot of scrolling in order to get to the results the user desires.

Various tabs have been created in the interface to separate related results generated by processing inputted files.

2. **Business logic** – This is the core logic of the application where all the processing takes place, files are uploaded through the interface which are processed using C# code.

After files are uploaded successfully and important information is extracted and grouped, the C# code for each keyword extracted from the uploaded files checks if it is in the database or not and whether it has a synonym or not. If the keyword does not exist in the database it goes to www.thesaurus.com to retrieve all the synonyms for the keyword and inserts them in the database.

If the keyword is not found at the website then it checks whether the keyword is wrongly spelled. The application would then try to find the correct keyword and if found it would be inserted in the database as corrected keyword. The code makes another attempt of finding synonyms for the corrected keyword to ensure integrity of the application.

If a keyword is a synonym of another keyword already processed then only one keyword is considered to avoid redundancy. This process is called Semantic Matching (SM).

3. **Database –** The database for this project has 2 tables: synonyms and corrected. Synonym table contains three columns: ID, keyword and synonym where ID column is a unique identifier, keyword column stores all the keywords processed from uploaded files and the third column synonym stores the synonyms of each keyword. As there can be multiple synonyms for each keyword, there are multiple rows for each keyword in the table representing multiple synonyms.

Corrected table on the other hand also contains 3 columns: ID, keyword and correctedkeyword where ID column is a unique identifier, keyword column stores all the keywords processed from uploaded files and the third column correctedkeyword stores the keyword that was corrected. If a keyword does not have any spelling errors then both keyword and correctedkeyword column will have same value for that keyword.

Figure 5 and Figure 6 below is the design that was created on the basis of which interface was developed. This is just a design to give the look and feel of the interface.

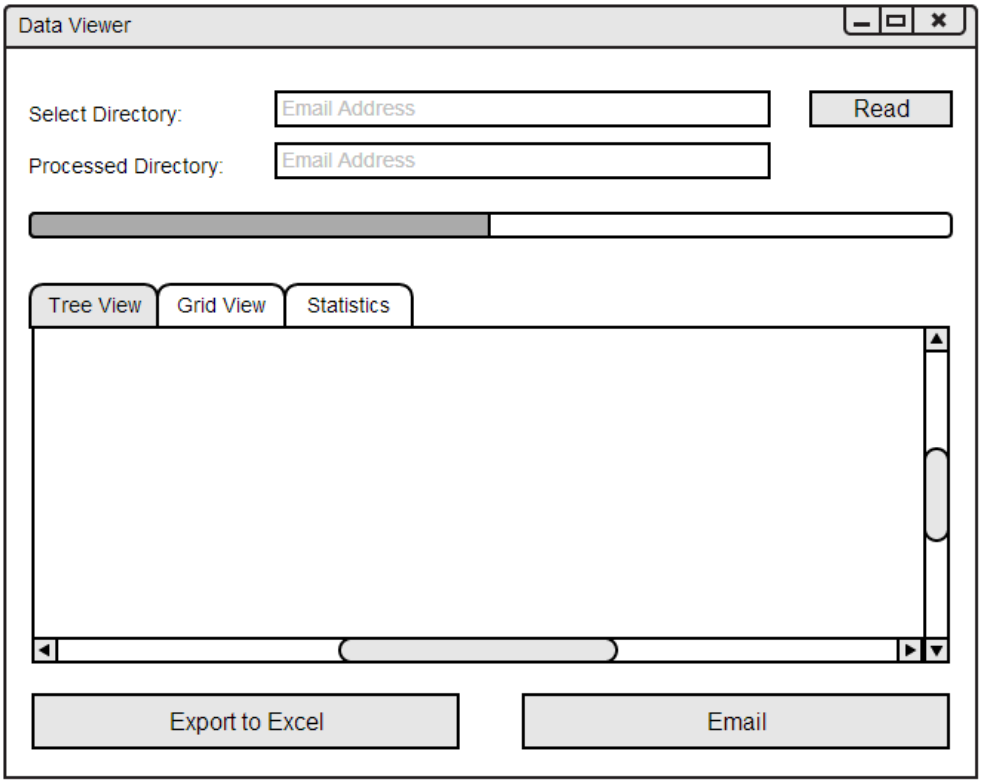

Figure 5. Main interface

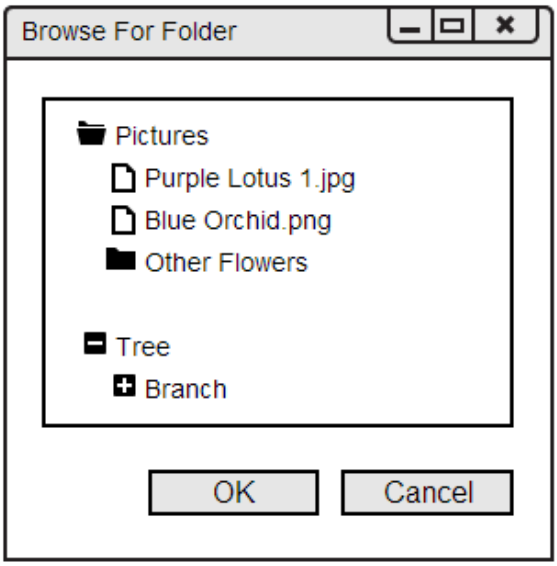

Figure 6. Browse for folder

4.4. Working of the Application

In the previous sections we have discussed how SM and EC has helped us in reducing keywords, there by optimizing the screen space for better readability and understanding of the results. Now we will further discuss how the application works as a whole.

Figure 7 shows how the application looks like when the application's interface is opened for the first time. Upon clicking the text box for select directory, trigger is enabled to open the popup box for browsing and uploading the data files.

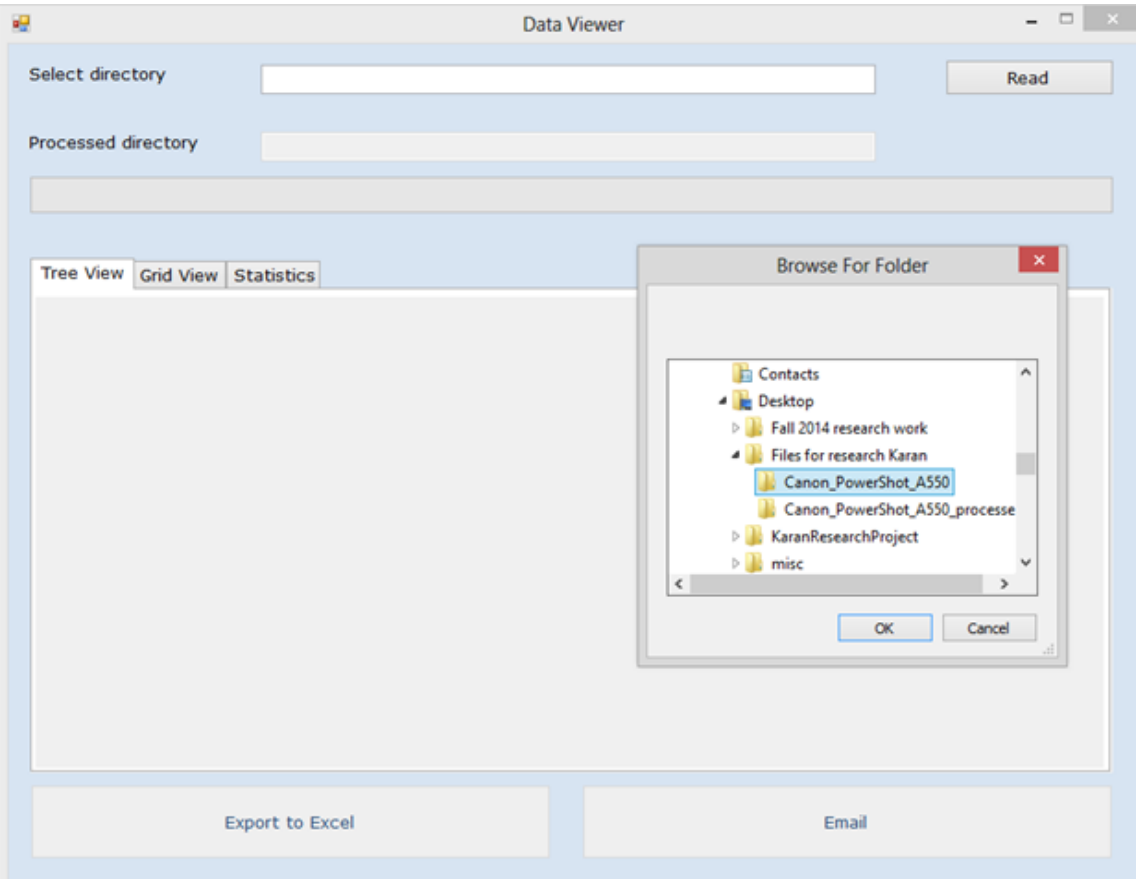

Figure 7. Main interface

The user only uploads the unprocessed directory and the processed directory is automatically uploaded as the processed directory has _processed concatenated at the end of

unprocessed directory. Path for both the directories (processed and unprocessed) can be seen on the text boxes in front of the labels: “Select directory” and “Processed directory”.

After the files are successfully uploaded the user will click on the read button to process all the uploaded files. In this process if the user does not upload anything and clicks on read button or uploads incorrect format files then a validation is also provided so that the user knows what is going wrong. Following screen shot shows the validation:

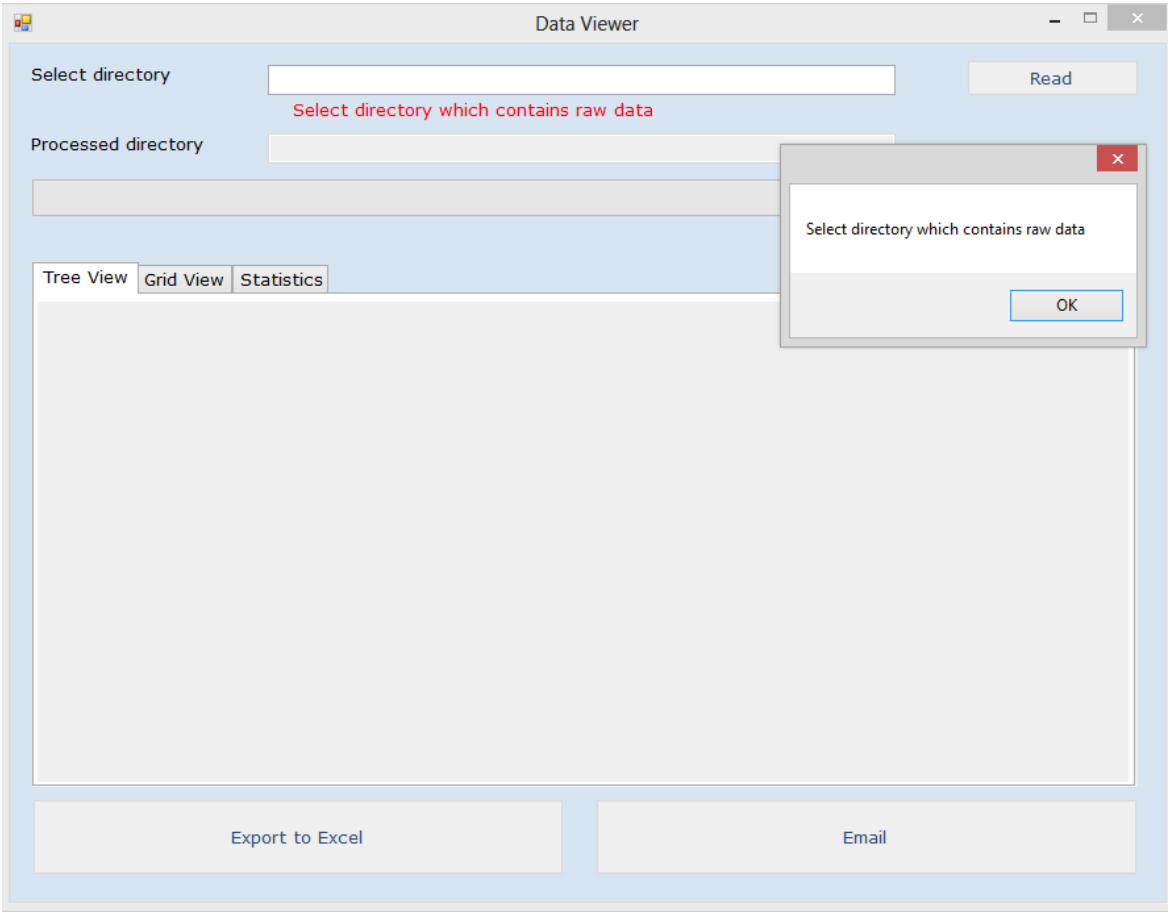

Figure 8. Main interface with validations

When correct files are uploaded and the user clicks on read button the application starts processing all the files uploaded by the user. Time for processing all the files can be seen on the top right corner of the interface where the user gets an estimate of how much time has passed in

processing the files. The user also gets the privilege of viewing how many files have been processed by looking at the long processing bar below the processed directory label which shows how many files have been processed and what's the total number of files that are being processed.

The interface has three tabs: Tree View, Grid View and Statistics. Initially all three tabs are grayed out which means that the results are not generated yet and the user cannot select anything or choose any option while the files are being processed. Once the processing is completed the processing bar turns fully green and shows that all the files have been processed and the results have been generated.

Once all the files are processed and results are generated, all three tabs are populated with results and the user can maneuver over the interface to see their desired section of results. At the very bottom of the interface there are two buttons: "Export to Excel" and "Email", both of which are also grayed out during the processing of the files but are activated as soon as all the files are processed successfully. Export to Excel would export all the data from grid view to excel spread sheet where the user can store the results and can infer to them whenever they want to without running the application again. Email button is for emailing the results from the grid.

Figure 9 shows how the interface looks like when the user has successfully uploaded the files and all the files have been processed successfully with results.

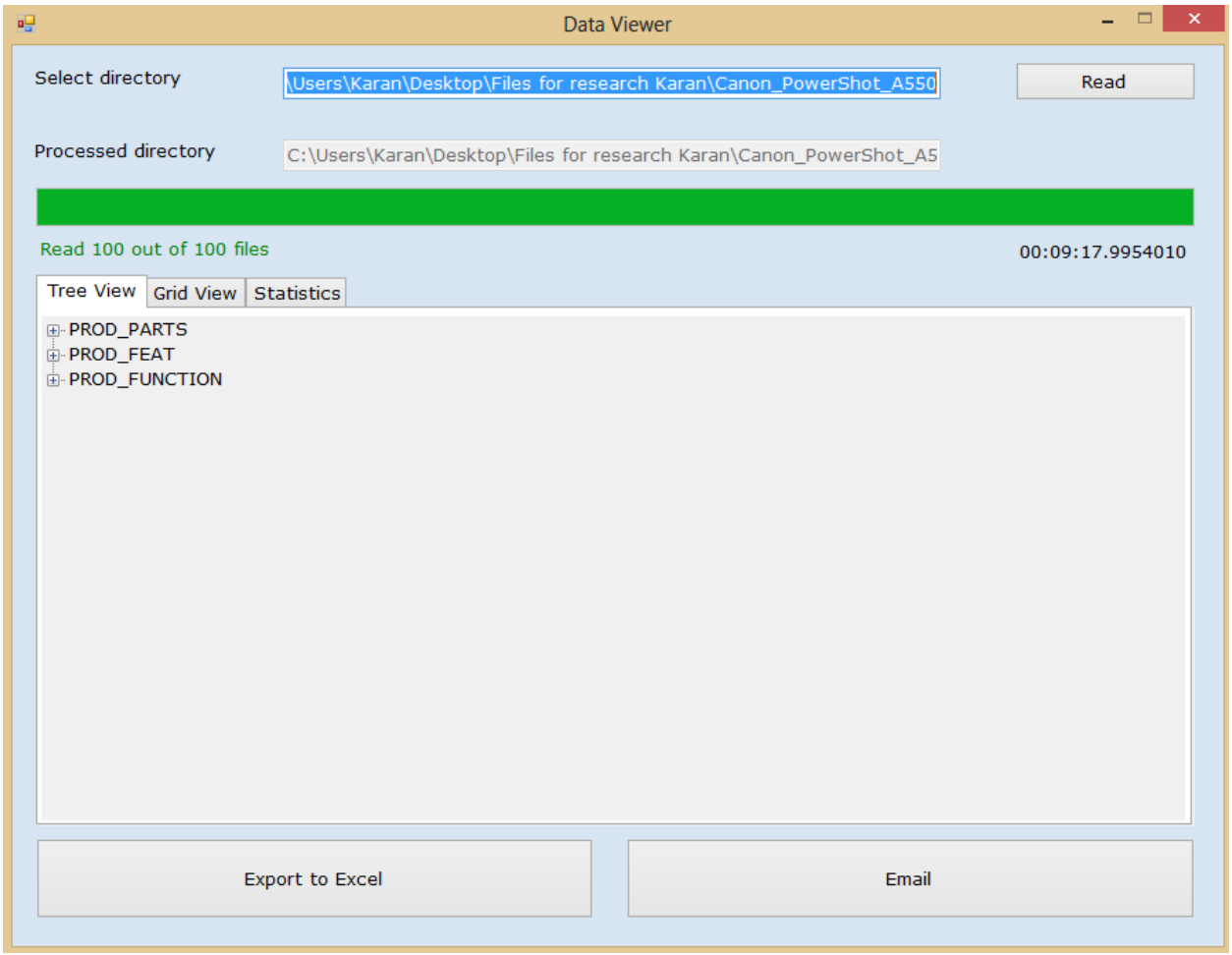

Figure 9. Main interface after generating results

The three major sections where the output is displayed on the interface are:

1. **Tree View:** This is the first view that the user looks at the when all the processing is completed. It has three major segments: PROD_PARTS, PROD_FEAT and PROD_FUNCTION. Each segment can be expanded and collapsed by the user to the view they desire. Each segment can be further expanded to provide relevant information. Figure 10 shows on expanding how the interface looks like and what all results are included in it.

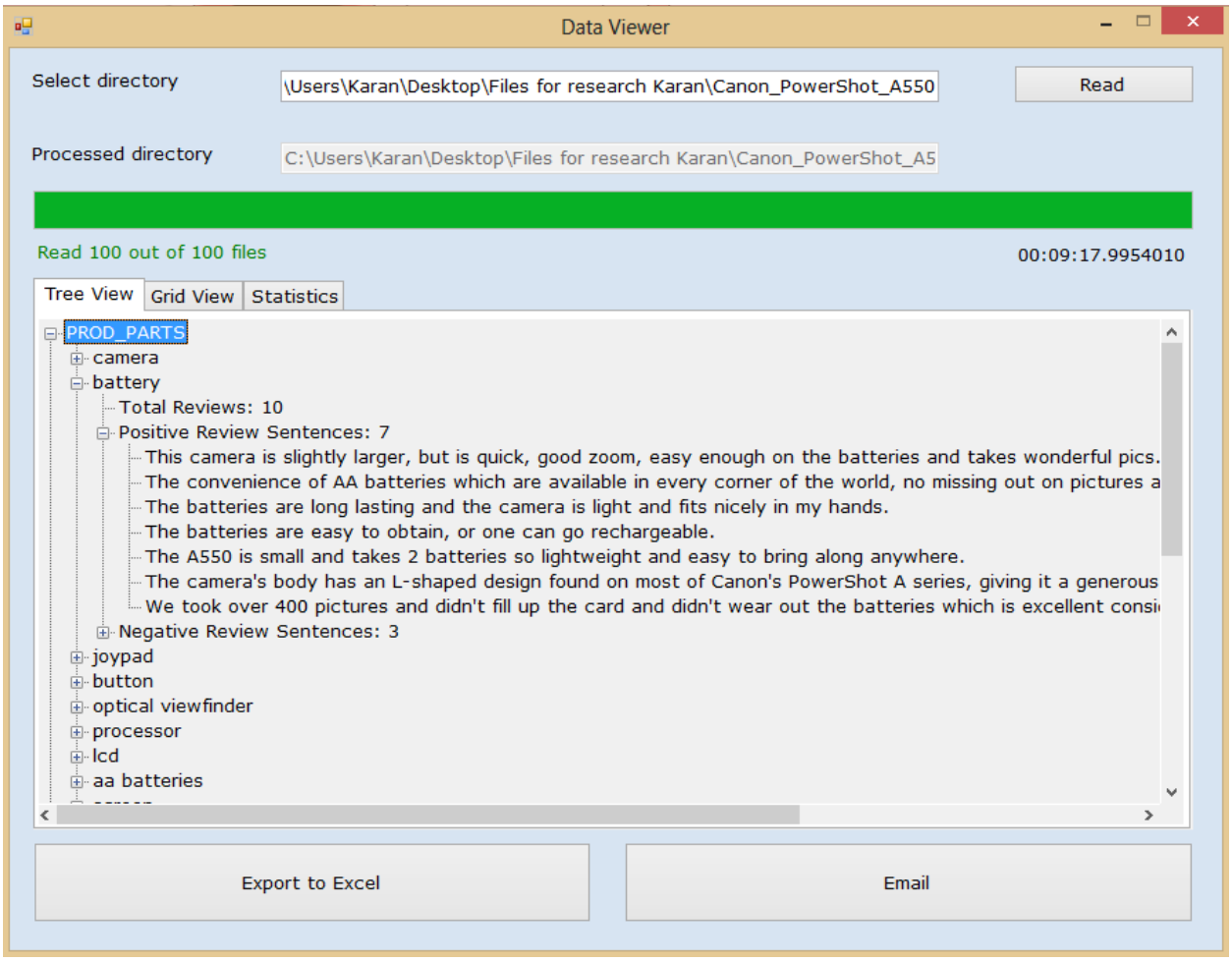

Figure 10. Expand/collapse feature on main interface

2. **Grid View:** The user has the option for viewing all the information in the form of a Grid where they can do analysis in terms of a matrix. Following figure shows the Grid View. The Grid View was designed so that the user has freedom of just looking at how many were positive and negative reviews for a particular feature so that they can do a better analysis on the part of the product which is most liked and/or disliked.

This part of the interface also shows user what type they are looking at so that if they want they can sort the results by clicking on the header.

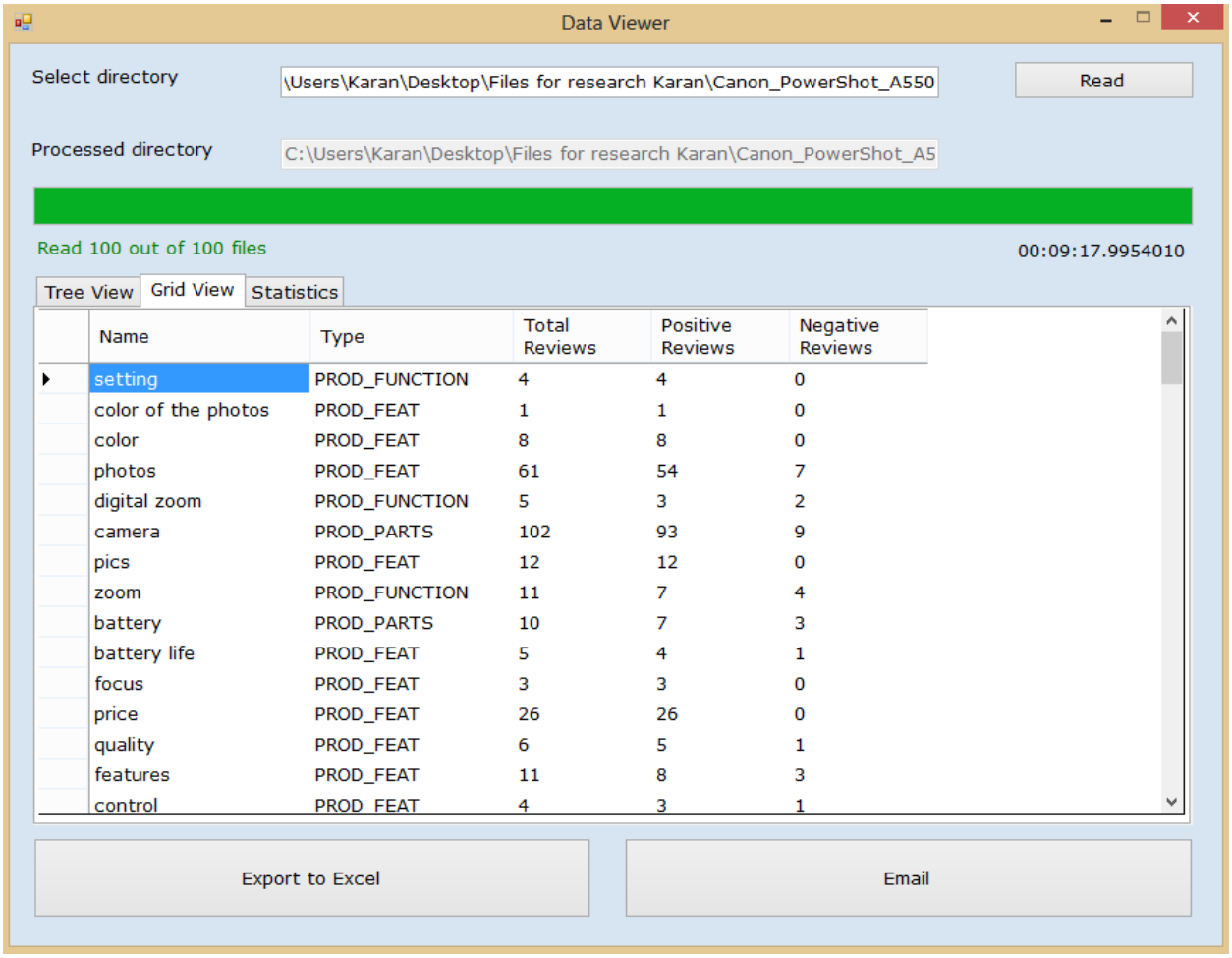

Figure 11. GridView feature on main interface

3. **Statistics:** This section of the interface shows the qualitative analysis of the generated results i.e. total number of keywords that were found, the keywords that were corrected by EC technique, corrected but were synonym, corrections that were found in the database and lastly how many keywords are uncorrectable.

By looking at these statistics the user can infer about the data i.e. how much mistakes people generally make while writing reviews online and also get a broad picture of the quality of the application. Following figure shows the screen shot:

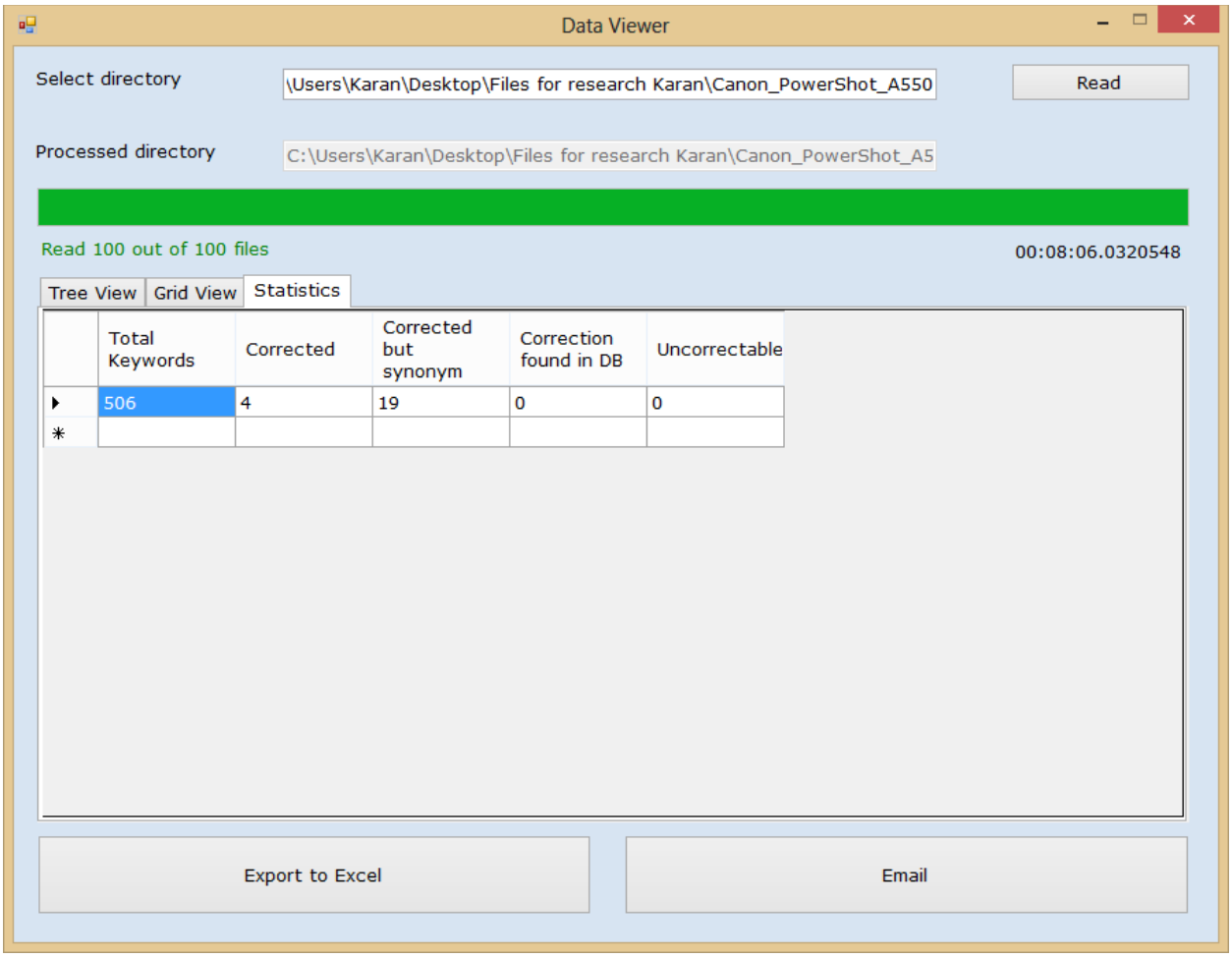

Figure 12. Statistics on main interface

In order for this application to work fast, it required a strong internet connection; stronger the internet connection, faster the processing of all the files. We will see how the speed of an internet connection is responsible for the faster delivery of the results in the Performance section.

CHAPTER 5. PERFORMANCE

Application's performance is the key to developing a good application as no one wants to sit and watch a file get processed or opened. Performance was kept in mind while building this application. As mentioned in the chapters above, we are using SM and EC techniques to improve the quality of the application thereby reducing redundancy, optimizing screen space and making results more readable. But using SM and EC has its toll as it degrades the performance by increasing the processing time because the application has to go back and forth between the Internet and process all the GET requests.

If there was no SM and EC in the application, then the results would have been generated quickly but the quality of the results would have been compromised. In order to avoid that, we included database that would store all the keywords along with the synonyms and their corrections. There are lots of factors that are responsible for improving efficiency but the strongest factor in our application was to include the database.

Database helped us in achieving efficiency as in order for SM and EC to work it needs synonyms and correct words which the application gets from the HTTP GET request. We load the keywords, corrected keywords and synonyms of keywords that were processed earlier, from the database into a hash table and then allow the application to look into the hash table before going over the Internet. If a keyword is found in the hash table then we take that keyword and place it in the list for further processing of the results and if not then it goes to the Internet to search for synonyms or misspelled keywords.

The best case scenario would be if the files that need to be processed have all the keywords already in the database. The processing time is almost reduced up to one fifth of the time it would have taken if there was no data in the database or if there was no database at all.

The worst case scenario would be if the file that needs to be processed has none of its keywords in the database and that none of the keywords have any synonym keywords in reviews. For each keyword the application would have to look up on the internet for synonyms or correcting misspelled keywords.

The time taken to read 100 reviews with an internet speed of 5.51 Mbps and no data in the database i.e. the worst case scenario, is approximately 9 minutes and 18 seconds. Total keywords that found were 506.

Time taken to read the 50 files with an internet speed of 4.29 Mbps with no data in the database i.e. the worst case scenario is approximately 6 minutes and 50 seconds. Total keywords that found were 275.

For the best case scenario, time taken to read 100 reviews with an internet speed of 4.99 Mbps is approximately 2 minutes and 13 seconds. Total keywords that found were 506.

Time taken to read 50 reviews with an internet speed of 4.69 Mbps is approximately 1 minute and 22 seconds. Total keywords that found were 275 and all of them were found in the database.

Table 3 represents the performance in tabular form.

Table 3. Performance of application

Total Keywords	Keywords found on web	Keywords found in database	Processing Time	Internet Speed (Mbps)	Total number of reviews
506	506	0	9 minutes and 18 seconds	5.51	100
275	275	0	6 minutes and 50 seconds	4.29	50
506	0	506	2 minutes and 13 seconds	4.99	100
275	0	275	1 minute and 22 seconds	4.69	50

When we talk about performance we usually ask how good the testing was. If testing was not done then we wouldn't know if there was any room for improvement. While testing the application it came to attention that the performance needs to be improved thereby implementing database. There were three kinds of testing done to test this application:

1. **Unit Testing:** It is done by testing individual units of the source code and for this application all the functions were tested and the response was recorded and analyzed to confirm appropriate results. In total, there are 30 functions defined and all of them were tested individually and as a whole. Since, not all the methods can be tested individually so some of them were tested with other functions and some were tested along with the application.
2. **Black Box Testing:** This is a very common type of testing where we test the application in terms of the way it is supposed to function. In order to achieve this kind of testing, the application was executed many times by increasing and reducing total number of files to be processed, there by generating different sets of results which could be easily analyzed.
3. **Functional Testing:** This type of testing is a kind of black box testing but is input/output specific where we have examined when we have input in the application and after processing what is the output.

Table 4 represents the keywords which were identified by the system and were incorrect which were corrected by it. Phrases, keywords with more than one word, cannot be accounted for as the system cannot identify them as equivalent.

Table 4. Wrongly spelled words and system provided

Keyword	Correction
joy pad	Joypad
view finder	Viewfinder
moive mode	movie mode
aa batteries	aa batteries

Table 5 represents keywords and semantically equivalent words detected by the system.

Table 5. Semantically equivalent words detected

Keyword	Equivalent
Aspect	Aspects
Upload	Uploading
Usability	Use
Video	Videos
lay out	laid out
Control	Controls
Image	Images
Perform	Performs
Setting	Settings
Button	Buttons
Photos	Pictures
Photos	Picture
Style	Styling
Look	Looking
Viewfinder	view finder
Capability	Capabilities
Battery	Batteries
Effect	Results
start up	start-up
Joypad	joy pad
Portrait	Portraits
Highlight	Highlights

Table 6 represents the sample dataset from which all the keywords were extracted. These keywords are distinct in the table.

Table 6. All words in sample dataset

Manage	red-eye removal	shooting modes	joy pad	Video	manual settings	digital camera
burst mode	lithium batteries	lcd screen	memory card	Noise	moive mode	indoor photos
movement stabilizer	Shutter	focusing system	Operated	macro shots	viewing screen	Iso
optical viewfinder	lcd display	second-curtain flash	zoom setting	Controls	flash photos	Camcorder
Download	Transfer	Worked	Detail	Screen	Details	default standard setting
start-up	Use	Handle	Usability	Looking	Transfering	Price
Fetures	Feature	landscape mode	Lcd	night photos	Performs	Features
video settings	flash setting	digital zoom	quality of the pictures	Value	Auto	quality of pictures
Vivid	Capabilities	Photos	aa battery	Viewfinder	battery cover	image quality
Money	Flash	Obtain	lag time	Exposure	Product	Contrast
Focus	battery compartment	optical zoom	ease of transferring the pictures	Control	Flexibility	Battery
Bring	Settings	battery charger	between-shot times	movie mode	Body	optical viewer
Uploading	Works	owners manual	Working	Products	Pictures	Batteries
View	Performance	video recorder	Program	video shots	automatic white balance	quality of the photos
battery life	Processor	Videos	double-a batteries	Zoom	Results	ease of use
aa batteries	Evening	Software	Images	Learn	shutter lag	laid out
Macro	Sound	Hold	digital and optical zooms	anti-blur feature	Pics	

Table 6. All words in sample dataset (continued)

face-detection metering system	view finder	Size	auto setting	Buttons	shot finder	
Camera	transfer pics	Colors	close-up mode	manual mode	Color	
audio/movie	flash power	menu tree	start up	Snap	set up	
Aspects	Portraits	Effect	Grip	auto red eye correction	Weight	
Highlights	Shot	Shots	movie capabilities	shutter cover	picture quality	
Resolution	Picture	Design	Quality	Service		
aa bateries	auto mode	purple fringe	Styling	Work		

Table 7 represents the keywords and their semantically equivalent words that were missed by the system.

Table 7. Semantically equivalent (missed)

Keyword	Equivalent
Pics	Photo
Color	Colors
Work	Works
Feature	Features
Work	Working
Snap	Photo

Table 8 represents the keywords that the system could not correct

Table 8. Error detection (missed)

Keyword	Correction
fetures	Features

Semantically equivalent detected = 22

Semantically equivalent (not counting phrases) = 28

Errors detected and corrected = 4

Actual Errors = 5

Final table i.e. Table 9 shows precision and recall calculated from the above tables.

Table 9. Precision and recall

	Precision (%)	Recall (%)
Error Correction	100	80
Semantic Matching	100	78.5

Precision for Error Correction is 100% as the system was able to detect all the incorrect words and correct all of them and recall is 80% ((4/5)*100) as on manually reading all the keywords I found that there was 1 error that the system couldn't detect so it couldn't correct. Similarly, precision for Semantic Matching is 100% as the system was able to identify all the semantically equivalent keywords and recall is 78.5% ((22/28)*100) as the system couldn't semantically equate 6 keywords.

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

People will always buy products whether they purchase online or by going to the shop, one thing that they will never stop doing is to give their opinion about the product they bought or sold i.e. to provide a detailed description of what they liked about the product or what they disliked.

Everyone wants to retain lot of information without having to sacrifice time and memory. In other words when we have to process a lot of information we store it and use it again and again, but the more efficiently we monitor data we make more improvement in the area of data storage and data retrieval.

Upon looking at same kind of data, different people can interpret it differently. In order to avoid confusion and to have everyone understand the meaning behind data, machine learning came into existence and with the help of machine learning it became easy to understand data. There was, however, still a need for data visualization.

Taking the advantages and power of machine learning, this application has visualized data in such a way that different kinds of users on reading the results will have a common conclusion about the data. The application will also allow user to understand the data in depth, comparing the results and hence reaching a fair conclusion to data and on the product it generated results for.

Hence this application will make everyone's life easier by visualizing data in such a way that anyone looking at the results can easily figure out the opinions given by hundreds of people in a short amount of time.

6.2. Future Work

I haven't seen any application where an improvement cannot be made so it goes the same with this application too. Currently this application is only supported by windows OS so an option to make this application platform independent could be a possible area of future work.

Also, currently this application can only be used by those who have the specific data files needed by this application, so creating this application as a web-based application could also be a good idea that can be looked into. As this project is data specific, creating a web-based application that could store results and update them dynamically can be looked forward to.

The results generated from this application can be used as live feed for many applications. Also, various mobile apps can also be considered so that the user can look at the results on their phone or even generate their own results.

The statistics section can also be improved by adding timestamps and storing all the results generated in the course of execution of the project where the user can look at the dates and compare results for different time, different date and different products.

Another area of improvement could be to use a stronger database. Currently the application uses Access database but due to increase in data, a stronger relational database could be used such as Oracle, SQL Server or even MySQL.

By implementing these features in this application it would become more powerful in terms of processing as well as understanding.

CHAPTER 7. REFERENCES

[1] Windows Forms: Definition – What does Windows Forms mean,
http://www.techopedia.com/definition/24300/windows-forms-net
(Last accessed on 20 August 2014)

[2] Windows Forms: Windows Forms Overview,
http://msdn.microsoft.com/en-us/library/8bxxy49h(v=vs.110).aspx
(Last accessed on 20 August 2014)

[3] How to: Create a C# Windows Forms Application,
http://msdn.microsoft.com/en-us/library/360kwx3z(v=vs.90).aspx
(Last accessed on 20 August 2014)

[4] Semantic matching and Error Correction,
http://thesaurus.com/
(Last accessed on 20 August 2014)

[5] UML Use Case Diagrams: Tips,
http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html
(Last accessed on 20 August 2014)

[6] XML Introduction – What is XML?,
http://www.w3schools.com/xml/xml_whatis.asp
(Last accessed on 20 August 2014)

[7] UML 2 Diagrams Tutorial,
http://ima.udg.edu/~sellares/EINF-ES2/uml2_diagrams.pdf
(Last accessed on 20 August 2014)

[8] Introduction to UML 2 Use Case Diagrams,
http://www.agilemodeling.com/artifacts/useCaseDiagram.htm
(Last accessed on 20 August 2014)

[9] How to: Make Requests to HTTP-Based Services,
http://msdn.microsoft.com/en-us/library/cc197953(v=vs.95).aspx
(Last accessed on 20 August 2014)

[10] Access Database: Database basics,
http://office.microsoft.com/en-us/access-help/database-basics-HA010064450.aspx
(Last accessed on 20 August 2014)

[11] Database: Building an Access database From the Ground Up,
http://databases.about.com/cs/tutorials/a/widgetmenu.htm
(Last accessed on 20 August 2014)

[12] Walkthrough: Connecting to Data in Access Database (Windows Forms),
http://msdn.microsoft.com/en-us/library/ms171893.aspx
(Last accessed on 20 August 2014)

[13] Semantic Matching: Semantic Matching,
http://semanticmatching.org/semantic-matching.html
(Last accessed on 20 August 2014)

[14] Semantic Matching: Semantic Matching
http://eprints.biblio.unitn.it/381/1/013.pdf
(Last accessed on 20 August 2014)

[14] Semantic Matching: Semantic Matching in Search
http://www.hangli-hl.com/uploads/3/1/6/8/3168008/ml_for_match-step2.pdf
(Last accessed on 20 August 2014)

[15] Error Correction: Error Correction and detection,
http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Error_detection_and_correction.html
(Last accessed on 20 August 2014)

[16] Error Correction: Error Correction and detection,
http://www.tutorialspoint.com/data_communication_computer_network/error_detection_and_correction.htm
(Last accessed on 20 August 2014)

[16] Error Correction: ECC (Error Correction Code or Error Checking and Correcting),
http://searchnetworking.techtarget.com/definition/ECC
(Last accessed on 20 August 2014)

[17] Testing: Unit Testing,
http://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx
(Last accessed on 20 August 2014)

[18] Testing: Black Box Testing,
http://softwaretestingfundamentals.com/black-box-testing/
(Last accessed on 20 August 2014)

[19] Testing: Functional Testing,
http://www.techopedia.com/definition/19509/functional-testing
(Last accessed on 20 August 2014)

[20] Visual Studio: Visual Studio 2012,
http://msdn.microsoft.com/en-us/library/vstudio/dd831853(v=vs.110).aspx
(Last accessed on 20 August 2014)

[21] XML: Extensible Markup Language,
http://www.w3.org/XML/
(Last accessed on 20 August 2014)

[22] Amazon: Customer Reviews,
http://www.amazon.com/Canon-Digital-Camera-18-135mm-Lens/product-reviews/B00DMS0LCO/ref=cm_cr_dp_qt_see_all_top?ie=UTF8&showViewpoints=1&sortBy=byRankDescending
(Last accessed on 6 September 2014)

[23] Jin, Wei, Ho, Hung Hay, Srihari, Rohini K., OpinionMiner: A Novel Machine Learning System for Web Opinion Mining and Extraction, *KDD'09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009; 1195-1204

[24] Dave, Kushal, Lawrence, Steve, Pennock, David M., Mining the Peanut gallery: opinion extraction and semantic classification of product reviews, *Proceeding WWW '03 Proceedings of the 12th international conference on World Wide Web*; 519-528

[25] Pang, Bo, Lee, Lillian, Opinion Mining and Sentiment Analysis, *Foundations and Trends in Information Retrieval*, 2008; 2(1-2):1-35

[26] Riloff, Ellen, Wiebe, Janyce, Wilson, Theresa, Learning subjective nouns using extraction pattern bootstrapping, *CONLL'03 Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*:4:25-32

[27] Hu, Minqing, Liu, Bing, Mining and summarizing customer reviews, *KDD'04 Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining;* 168-177