ReviewBot: Augmenting Prediction of Hotel Reviews with a Recommendation System

Rishabh Tariyal

A dissertation submitted in partial fulfilment of the requirements of Dublin Institute of Technology for the degree of M.Sc. in Computing (Advanced Software Develop)

2019

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Advanced Software Develop), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: _____

Date: **04 Jan 2019**

ABSTRACT

In this thesis, we explore notions augmenting Recommendation System in the form of case based reasoning, collaborative filtering and sentiment analysis to predict the hotel reviews. Successful prediction techniques would be advantageous for those who write reviews but are not proficient in English language. Additionally, the approaches in this thesis may also be applied to other areas for prediction as well.

We present a novel approach to augment prediction of recommendation system with the addition of extra variables in the existing environment. We explore several classification techniques to predict performance on the hotel text review generator. We also provide insights for future work to build on this thesis to potentially predict reviews better and coherent.

Key words: *Recommendation System, Text Classification, Sentiment Analysis, Latent Semantic Analysis, Case Based Reasoning, Hotel Review Recommendation, CRISP-DM, Natural Language Processing.*

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Damian Gordon whose support throughout this dissertation helped me accomplish something which means a great deal to me.

I would also like to thank Luca Longo and all my classmates for all their assistance and contributions to the research.

And finally, I would like to thank my family and friends who supported me throughout the last few months.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1. Project Background

Curiosity is a part of human nature and travel can help people learn more about the world, but before we start our journey from our home to unknown places we wanted to be sure to book a safe place to stay. The Internet plays an important role in gathering that information in the form of hotel reviews. According to a report by EyeForTravel entitled "Travel Distribution & Marketing Barometer", a majority of 60% of the travel industry still ranks online search as being the number one way to attract more tourists to a particular site. Travellers who plan their trip after reading reviews on the Internet accounts for 55% of visits.

Typically, out of top 10 international tourists, 5 are non-native English speakers (data worldbank, 2017). Online portals like TripAdvisor provide users with the ability to read or write reviews in the English language. This creates a significant linguistic barrier for a non-native English speaker to convey their thoughts properly. At times words could be misleading and are unable to coherently capture reviewers' actual thoughts. Adding in the ability to give numerical scores to subcategories (such as room, location, cleanliness) from a scale 0 to 5 seem to be an effective way to help in this matter. Number systems are universal and remove the fuzziness of English grammar. A smart case-based recommender system can be developed for a user that will produce a template review that correlates with the given ratings.

In this research a context-based search recommender system will be designed, developed, and evaluated. The research will aim to show that context-based systems can be used for the construction of a "cold start" recommender system. It will further show that contextual information can be mined from review texts, and analysed for common traits per a context group. Much research has already been performed in the area of recommender systems and information retrieval. However, most recommender systems base their retrieval decisions solely on previous review text, whereas information about review ratings is often ignored. Table 1 shows a sample review which contains text and some numerical values representing amenities scores for a hotel.

Hotel Review	Location	Hospitality	Price	Room	Food
The room was very spacious and stylishly furnished. Transport is close by and Mitte is easily walkable. There are parks around that are good to run around. This area of around is a little out of the action; but the room is affordable; cleaned every day and I enjoyed my time here.	3	4	5	5	4

Table 1: Sample review along with sub-rating scores.

Case Based Reasoning (CBR) is an automated reasoning and decision making process whereby a new problem is solved through the experience we have accumulated in solving previous cases. Richter and Weber (2008) stated that term "case" is basically an experience of a solved problem, the term "based" implies that the reasoning is based on cases and the term "reasoning" means that the approach is intended to draw conclusions using cases. According to Aamodt and Plaza (1994) CBR is structured as a four-step process, sometimes referred to as the 4R's:

- retrieval,
- reuse,
- revision and
- retention

Retrieval is the process of finding a case that is similar to current situation or state. Reuse is when we retrieve a case and propose it as a valid action to apply in the current state. Revision is when we evaluate through a series of metrics or simulation how well the proposed action will perform. Retention is applied after a successful execution by storing the result of this experience in memory.

1.2. Project Description

A hotel review recommender system implementing sentiment analysis (incorporating Natural Language processing techniques) on the stored cases to retrieve similar solutions was developed by Tatemura (2000) who stated that sentiment analysis identifies and categorizes opinions expressed in the text of reviews. In order to

determine whether the user's attitude towards a particular amenity of a hotel is positive, negative, or neutral sentiment analysis proves to be a promising approach.

A fact that is often ignored is that most of the time users are not professional journalists and thus can miscommunicate their actual opinion for the amenity. In general, sentiment and subjectivity are quite context-sensitive, and, at a coarser granularity, quite domain dependent (Pang and Lee, 2008, p. 19). Even the exact same expression can indicate different sentiment in different domains and different contexts. For instance, a reviewer can say "*The rooms were nice" and gave 5 star ratings to rooms while other reviewer can say "Rooms were the nicest*" and gives 5 star ratings. The problem with first review is that, they gave best possible rating to rooms but didn't use superlative degree in their text while expressing his feelings in words.

Moreover, sentiment analysis typically generates a Boolean value for a given amenity, which is hard to map with an actual scale of rating which is usually a discrete set of values ranging from zero to five.

1.3. Project Aims and Objectives

The retrieval process for finding a text review that is similar to current rating (case) can be improved by taking an amenities individual scores into consideration. When hotel amenity scores that include location, hospitality and rooms are combined with sentiment scores or hotel's review, similarity of retrieval process in a CBR based hotel review recommender system improves significantly.

The objective of this research is to show that the common solution for recommending hotel reviews given sub-ratings (location, hospitality, price, food, and room) does not produce relevant reviews and use of sub-rating scores along with sentiment analysis can effectively improve recommendations. Two CBR based hotel review recommender systems will be built. The first system will use Natural Language Processing (NLP) in the retrieval phase to extract sub-rating scores from text and the second system will use the sub-ratings scores submitted by users along with sub-ratings scores extracted using NLP. Each system will recommend different review text from the sub-rating scores. An online survey will be conducted in which people must choose either of the generated text, which they find most correlated for a given set of sub-ratings. The system, whose text will be selected the most number of times will imply a better

retrieval process. Quantitative research methodologies will be used to justify hypothesis. Each time the user chooses the text, it will be stored in persistent data storage. The ratio between the numbers of times the text was chosen to the total numbers of times the text was generated will be compared between the two systems.

1.4. Orthogonal Issues

This project covers only five aspects of hotel's amenities i.e. location, hospitality, price, food, and rooms. All other amenities such as recreation, outlook are ignored. The user input to the system is limited to whole numbers from 0-5 and no other input is accepted.

The focus of the project is to generate semantically correct sentences rather than grammatically correct sentences. The project is built using Python as a programming language. The project neither explores machine learning capabilities of other programming languages nor do any comparison between models built by using other programming languages.

There are multiple machine learning techniques available but only the best suited is applied in this dissertation. Since there is no hard line to distinguish which technique should be implied under a given circumstance, it becomes hard to choose one. A valid reason is provided in "Technology Choices" section under Experimental Design chapter while choosing a technique based on surrounding factors of the dataset and project need and experiments are conducted to choose the technique.

The size of raw dataset is 68.7 megabytes which is present in a spreadsheet that contains 106,266 rows. The dataset and the output generated by the proposed algorithm is in English language only. The models are trained using this dataset only which was divided into 80% training dataset and 20% test dataset.

The weights of each aspects of a text review are considered equal ranging from 0 to 5 as whole numbers. Reviews present in the corpus that are missing rating of any of the aspect will be discarded.

1.5. Thesis Roadmap

The rest of this dissertation is presented as follows:

Chapter Two; Recommender Systems and Case-Based Reasoning: This chapter provides a detailed description of a Recommender System, its evolution, and modern techniques available to build a recommender system.

Chapter Three; Sentiment Analysis: It explains the importance of natural language processing and how to extract meta-information such as emotions out of English language using computer understandable algorithms. This chapter also highlights some the widely used techniques for sentiment analysis.

Chapter Four; Experimental Design: The Design methodology, experiment architecture and its stages are explained in this chapter. It also explains a reason why a certain methodology was chosen over others.

Chapter Five; Experiment Deployment: This chapter walks through the experimental setup, how it was created, how the volunteers were recruited and how the response was collected and analysed.

Chapter Six; Evaluation: This chapter details the results obtained from the experiment. The result obtained are both qualitative and quantitative. It also draws a comparison between classification algorithms.

Chapter Seven; Conclusions and Future Work: This chapter summarizes the results and proposes future work.

2. RECOMMENDER SYSTEMS AND CASE-BASED REASONING

2.1 Introduction

This chapter covers available Artificial Intelligence techniques for building a recommender systems. First, this chapter explains Recommender systems (RS) and the range of approaches associated with them. It also draws a comparison between different approaches. Secondly, it covers Case Based-Reasoning (CBR) systems, how it is related to RS and how it can be implemented. This chapter also identifies the best applicable RS technique for this research and point out where exactly we want to tweak the existing approaches.

2.2 Recommendation System

The first Recommender System (RS) was developed by Goldberg *et al*. in 1992, in order to segregate emails that users had in their inbox. It is a filtering algorithm where users collaborate by registering their reaction to the documents after reading them. Over the past twenty years, the demand for recommender systems has increased significantly, as they allow users to deal with large amounts of data, providing them with a selection of personalized recommendations, services, and contents. Thus, various techniques have been developed and studied, both by the scientific community and by private companies. A recommender system is composed of two modules: a *database* and a *filtering technique* (Bobadilla, Ortega, Hernando, and Gutiérrez, 2013). The database is responsible for storing the information about users, items, and the associated ratings. The filtering technique is composed of an algorithm, generally split in two stages; first, the similarities between users or items are identified providing a neighbourhood of each item or user, and second, the system predicts ratings corresponding to the items unseen by the user, and only the best ones are used as recommendations (Zhou, Xu, Li, Josang, and Cox, 2011). The three most common approaches of recommender systems are as follows:

- Collaborative filtering
- User/Memory-based
- Item/Model based

2.2.1 Collaborative filtering(CF)

This algorithm performs a comparison of users' ratings, resulting in the identification of the most similar ones (Bobadilla, Serradilla, and Bernal, 2010). Most of the research has been done using this type of filtering provides good results. The two main focuses of research are how to define the similarity metric and how to predict a rating to an item not rated by a user. CF process can be divided into three steps:

1. Collecting user ratings data matrix.
2. Selecting similar neighbors by measuring the rating similarity.
3. Generating prediction.

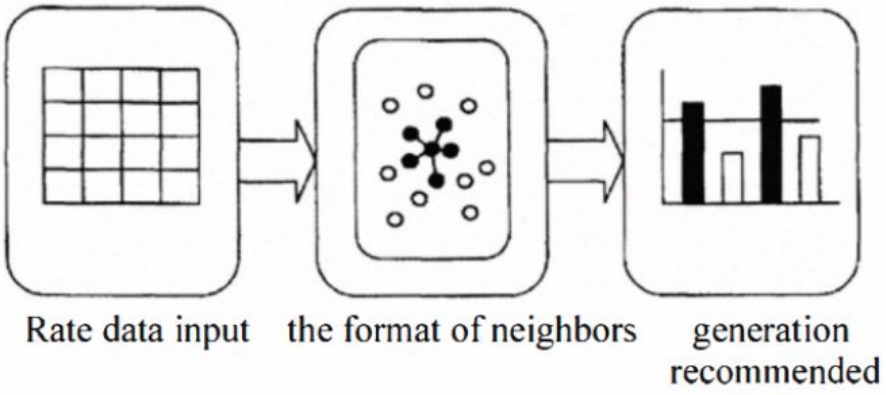


Figure 1: Collaborative filtering process (Sarwar, Karypis, Konstan.,, 2001).

Generally, input data in recommendation system based on the CF technology consists of user, item, and user opinions on observed items as a matrix m × n. Symbol m symbolizes the total number of users and n symbolizes the total number of items $R_{m,n}$ is the score of item In rated by user U_m. The CF approaches use statistical techniques to analyze the similarity between users and to form a set of users called neighbors. A set of similarity measures is a metric of relevance between two vectors (Gong, Y. and Liu, X., 2001).

User-based similarity is to compute the relevance between users as the values of two vectors. In UBCF, after the similarity is calculated, it is used in building neighborhoods of the current target user. Since the similarity measure plays a significant role in improving accuracy in prediction algorithms, it can be effectively used to balance the ratings significance (Gong, Y. and Liu, X., 2001). There are a

many similarity algorithms that have been used in the CF recommendation algorithms such as Cosine vector similarity, Pearson correlation, Euclidean distance similarity, and Tanimoto coefficient (Sarwar, Karypis, Konstan., and Reidl, 2001).

Euclidean distance similarity matrix is used in this dissertation. Euclidean distance method is based on the distance between items. It forms coordinates to put preference values between items and measures Euclidean distance between each point. When distance value between two points, sim(i,j), is large, it means the two points are not similar. When sim(i,j) is small, it means two points are similar. This is Euclidean distance formula is given below.

$$sim(i,j) = \sqrt{\sum(R_{k,i} - R_{k,i})^2}$$

where $\sum$ goes from k=1 to n. $R_{k,i}$ is the ratings of the target Item i given by User k. $R_{k,i}$ is the ratings of the other Item j given by User k. n is the total number of rating users to Item i and Item j.

2.2.2 User/Memory-based

This approach is based on identifying the users' neighbours (i.e. the most similar users) and then predicting a rating, based on the ratings of the neighbours. A memory-based CF approach, or nearest-neighbour (Jin, Chai, and Si, 2004) is said to form an implementation of the "*Word of Mouth*" phenomenon by maintaining a database of all the users known preferences of all items, and for each prediction performing some computation across the entire database. It predicts the user's interest in an item based on ratings of information from similar user profiles. The prediction of a specific item (belonging to a specific user) is done by sorting the row vectors (user profiles) by its dissimilarity to the specific user. In this way, ratings from users that are more similar will contribute more to the rating prediction. Memory-based CF methods have reached a high level of popularity because they are simple and intuitive on a conceptual level while avoiding the complications of a potentially expensive model-building stage. At the same time, they are sufficient to solve many real-world problems. Yet there are some shortcomings (Hofmann, 2004):

- **Sparsity** - In practice, many memory-based CF systems are used to evaluate large sets of items. In these systems, even active users may have consumed well under 1% of the items. Accordingly, a memory-based CF system may be unable to make any item recommendation for a user. As a result, the recommendation accuracy can be poor.
- **Scalability** - The algorithms used by most memory-based CF systems require computations that grow according to the number of users and items. Because of this, a typical memory-based CF system with millions of users and items will suffer from serious scalability problems.
- **Learning** - Since no explicit statistical model is constructed, nothing is learned from the available user profile and no general insight is gained.

The weaknesses of memory-based CF systems, especially the scalability and learning issue have led to the exploration of an alternative model-based CF approach.

2.2.3 Item/Model-based

This approach is based on identifying the items' neighbours (i.e. the most similar items) and then predicting a rating, based on the ratings of the neighbours. The motivation behind model-based CF is that by compiling a model that reflects user preferences, some of the problems related to memory-based CF might be solved. This can be done by first compiling the complete dataset into a descriptive model of users, items, and ratings. This model can be built off-line over several hours or days. Recommendations can then be computed by consulting the model. Instead of using the similarity of users to predict the rating of an item, the model-based approach uses the similarity of items. Prediction is done by averaging the ratings of similar items rated by the user (Sarwar, Karypis, Konstan., and Reidl, 2001). Sorting is done according to dissimilarity, as in memory-based CF. The difference is that the column vectors (items) are sorted around the specific item, and not as in memory based CF, where row vectors are sorted around the specific user. Sorting of the column vectors assures that the ratings from more similar items are weighted more strongly. Early research on this approach evaluated two probabilistic models, Bayesian clustering and Bayesian networks (Breese, Heckerman, and Kadie, 1998). In the Bayesian clustering model, users with similar preferences are clustered together into classes. Given the user's class

membership, the ratings are assumed to be independent. The number of classes and the model parameters are learned from the dataset. In the Bayesian network model, each node in the network corresponds to an item in the dataset. The state of each node corresponds to the possible rating values for each item. Both the structure of the network, which encodes the dependencies between items, and the conditional probabilities, are learned from the dataset.

As mentioned in section previously, memory-based CF approaches suffer from a data sparsity problem. Model-based methods solve this problem to a certain extent, due to their "compact" model. However, the need to tune a significant number of parameters has prevented these methods from widespread practical use. Lately, researchers have introduced dimensionality reduction techniques to address data sparsity (Wang, de Vries, and Reinders, 2006), but as pointed out in Huang, Chen, and Zeng (2004), some useful information may be discarded during the reduction process. Model-based CF has several advantages over memory-based CF. First, the model-based approach may offer added values beyond its predictive capabilities, by highlighting certain correlations in the data. Second, memory requirements for the model are normally less than for storing the whole database. Third, predictions can be calculated quickly once the model is generated, though the time complexity to compile the data into a model may be prohibitive, and adding one new data point may require a full recompilation. The resulting model of model-based CF systems is usually very small, fast, and essentially as accurate as memory-based methods (Breese, Heckerman, and Kadie, C.M, 1998). Model-based methods may prove practical for environments in which user preferences change slowly with respect to the time needed to build the model. Model-based methods, however, are not suitable for environments in which user preference models must be updated rapidly or frequently.

Collaborative Filtering	**User/Memory-based**	**Item/Model-based**
● Easy to implement with gradual learning curve. ● User profile is not	● Easy to start with but gradually becomes hard to understand. ● Needs a user profile	● Steep learning curve from the start. ● Needs a user profile

required and suits well for cold start problem • Easily scalable but has adverse impact when the dataset increases significantly. • Can easily adapt to changing variables.	to start with. • Works well be large dataset but needs adjustments very time dataset is changed. • Can adapt to changing variables easily.	to start with. • Outperforms other two techniques when while dealing with huge datasets. • Can only work for model and requires complete redesign if variable changes.

Table 2: Comparison of Recommender system approaches.

2.3. Case-Based Reasoning

The roots of CBR approach can be traced back to the work of Schank (1982) on the modelling of dynamic memory, where he explored the role of the memory of previous situations in problem solving and learning, eventually forming the basis of the earliest CBR systems. This occurred around the same time of Gentner's work on developing a theoretical framework for analogical reasoning. A pioneering work in the field was by Kolodner (1983), where she developed the first CBR system called CYRUS, working on cases of the traveling and meetings of the ex-US Secretary of State Cyrus Vance.

In general, the CBR approach can be applied to problem domains that are only partially understood, and can provide solutions when no algorithmic or rule-based methods are available. The main advantages of CBR over rule-based models include the following (Watson and Marir, 1994):

- CBR systems can be built where a model of the problem does not exist;
- Implementation is commonly made easy, as it is a matter of simply identifying relevant case features;
- CBR systems can be rolled out with only a partial case-base, as it will be continually growing due to its cyclic nature;
- CBR systems are highly efficient by avoiding the need to infer answers from first principles each time;
- Retrieved cases can be used to provide satisfactory explanations as to why the given solution is produced; and

- The case-based nature of the learning system makes maintenance easier.

The CBR model has been traditionally presented as a continuous cycle of retrieval, reuse, revision, and retaining of cases, noted as the mnemonic of "the four REs" (Aamodt and Plaza, 1994), see Figure 2 below.

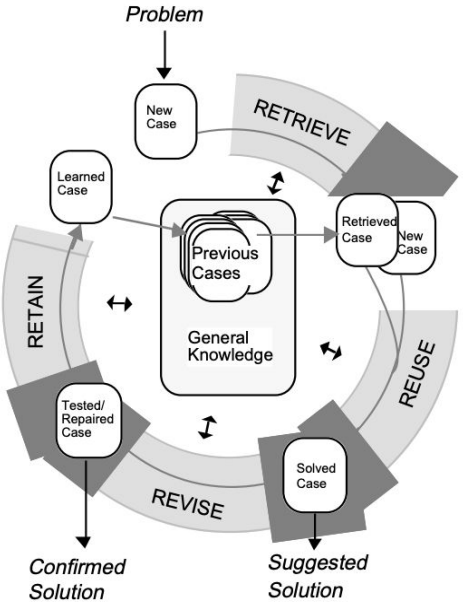


Figure 2: Case Based Reasoning cycle (Aamodt and Plaza, 1994)

2.3.1 Representation

Case representation is a major initial consideration of any new CBR system; fundamentally affecting the implementation of the remaining aspects of the cycle. A case in the CBR comprises of knowledge about a lesson learned in a past situation and the context in which this can be used. A typical CBR case contains information about the following (Watson and Marir, 1994):

- the problem describing the state of the world when the case occurred;
- the solution describing the derived solution to the problem; and
- the outcome describing the state of the world after the case occurred.

One can make different combinations of these three types of information in a case representation scheme: cases comprising problems and solutions can be used to derive solutions to new problems, while cases comprising information about problems and their outcomes can be used to evaluate and make predictions about new problems. Theoretically, all representational formalisms encountered in artificial intelligence

literature can be used as a basis of CBR case representation, including frames, propositional logic, rule-based systems, and networks. Case collection in CBR is an incremental process. That is, due to the cyclic nature of CBR, in which the case-base is repeatedly enlarged with new cases as they are encountered, the system can be deployed initially with a partial case-base. However, there are some factors to consider as to what kind of cases should be included in the initial case-base.

2.3.2 Retrieval

In this process, one or more cases similar to the current case are retrieved using some matching algorithm from the database of previously solved cases. Retrieval is one of the most important research areas in CBR. An issue highly related with case retrieval is the indexing of cases, whereby cases are assigned indices to facilitate their retrieval. There are both manual and automated methods of case indexing. Some common methods of indexing include:

- **Indexing by features and dimensions**, where the domain is analysed for determining the important dimensions and the cases are indexed by their values along these dimensions (e.g. MEDIATOR (Simpson, 1985), indexing along the type and function of disputed objects and the relationship of the parties);
- **Difference-based indexing**, where indices differentiate a case from other cases (e.g. CYRUS (Kolodner, 1983), discovering and selecting indices which differentiate cases best);
- **Similarity-based indexing**, where, after a process of generalization creating a set of indices describing abstract cases covering a common set of features, the unshared features are used as indices to original cases; and
- **Inductive learning based indexing**, where predictive features are identified and used as indices.

Case retrieval in CBR differs from database searches that look for a specific value among a given set of records due to the fact that in general there would be no existing case that would exactly match any given new problem, retrieval in CBR typically involves partial matches. A common method of retrieval widely used in CBR is nearest neighbour calculations, where similarity between cases is calculated using a weighted

sum of their features. A disadvantage of nearest neighbour approaches is that the retrieval time scales linearly with the number of cases in the case-base. Other retrieval methods are based on induction, where features that are most useful for discriminating cases are discovered by a learning algorithm, producing a decision tree structure to parse the case-base.

2.3.3 Adaptation

After the retrieval stage, the solution of the matching case from the case-base should be adapted to address the new case. While issues such as case representation, similarity computations, and retrieval have been amply addressed in CBR literature, adaptation has been considered the most difficult step and remains somewhat under-addressed and controversial. In large scale applications, while it is commonly easy to accumulate enough cases, the formulation of the required adaptation scheme is often difficult. Therefore, it is common to use very simplistic adaptation rules, or, bypass adaptation entirely, and to try to make up for this deficiency with a very comprehensive case-base ensuring the availability of a similar case for every problem instance. Adaptation models in CBR can be classified into several categories (Watson and Marir, 1994; Kolodner and Leake, 1996):

- **Null adaptation** is a direct simple application of the retrieved solution to the current problem without adapting. It is often suitable for classification tasks, or tasks which involve complex reasoning but a simple solution.
- **Substitutional adaptation** is accomplished by substituting values appropriate for the new case in place of the values of the retrieved case. This type of adaptation involves only changes in the values of some attributes and the structure of the new solution remains unchanged from the retrieved case.
- **Transformational adaptation** involves structural changes to the solution, such as the rearrangement of solution elements, or the modification, addition, or deletion of these elements. Transformational adaptation typically employs a fixed set of adaptation operators or rules, which are defined based on domain knowledge (Kolodner, 1993).
- **Generative adaptation** comprises the most complex adaptation techniques encountered in CBR literature, and involves the reworking of a portion or the

whole of the reasoning process that led to the solution of the retrieved case. Generative adaptation has been also referred to as **derivational analogy**.

2.3.4 Retaining

In the classical representation of the CBR cycle, retaining is the final step after an acceptable solution to the new case has been produced by the system. The newly solved case is added to the case-base of the system for making it available for future retrieval, enabling the CBR system to learn from its problem-solving experience. The retaining of new cases enlarges the coverage of problem space represented by the case-base. In addition to the solution to the problem, the steps used in deriving that solution can also be stored as a part of the case. For example, in a CBR system using generative adaptation such as derivational analogy, derivational traces describing the decision-making process for solving the problem can be retained for future use. A related issue is the maintenance of the case-base, for preventing uncontrolled growth of case-bases and addressing issues related to retrieval efficiency. Depending on the design of the CBR system and the complexity of the used case representation, many approaches are possible. For example, if a newly solved case is found to be highly similar to a case already in the case-base, the new case may not be retained at all, or the two cases may be merged.

2.4. Conclusions

Recommendations have become an integral part of modern era. With the increasing number of options, people have become very particular about their preferences. We discussed how RS provides a novel approach to suggest options to users as per their preferences. One way to approach this problem is via CBR. CBR is a problem solving paradigm in Artificial Intelligence where knowledge of previously solved cases is utilized to address a new problem.

The dataset used for this thesis is limited with no prior user input and contains five changing variables (sub-ratings). In which case Collaborative filtering approach fits the best as described in Table 2.1. While retrieving case from CBR system we are going to merge user provided sub-rating score with the sentiment score of the review and examine the impact on predictability of existing recommender system.

3. SENTIMENT ANALYSIS

3.1 Introduction

This chapter focuses on subdomain of Artificial Intelligence and techniques associated with it. This chapter describes how a sentence written in human readable language can be broken down into smaller segments and sentiment of the sentence can be extracted. Shortcomings of the available technique is also discussed in this chapter. The hidden idea is to gain knowledge about sentiment analyzer through which a recommendation system can be build. That recommendation system will then server the purpose to answer the research question.

3.2 Opinion Retrieval

One of the differences between a human and a computer system is that a human has an ability to articulate personal opinions, and a computer system doesn't. The dream of Artificial intelligence is to make machines behave like humans, and a promising field in that task is from computational linguistics that analyses opinions, called either *Opinion Mining* (OM) or *Sentiment Analysis* (SA). Opinion mining deals with the analysis of opinions about products, services, and even people. Liu (2012) says, sentiment analysis and opinion mining primarily focus on opinions that convey or imply positive or negative sentiment. To perform an analysis of opinions, opinions must be extracted.

Sentiments can be recognized as emotions, or as judgments, opinions or ideas prompted or coloured by emotions or susceptibility or feelings (Medhat, Hassan, and Korashy, 2014). There are two types of textual information: *facts* and *opinions*. While the facts are objective expressions about objects, features, entities, events and their characteristics, opinions are ordinarily subjective expressions that identify people's sentiments, views or feelings toward objects, entities, events, and their characteristics (Medhat, Hassan, and Korashy, 2014).

There are millions online users, who write and read content around the world daily. Online daily sentiments become the most significant issue in making a decision. An annual survey conducted by Dimensional Research explores the relationship between the percentages of trust there is in online customer reviews compared to personal

recommendations. These percentages vary in different years, as per Neuhuttler, Woyke, and Ganz (2017):

Year of Survey	Percentage of customers' confidence based on online personal recommendation reviews
2011	74%
2012	60%
2013	57%
2014	94%

Table 3: Summary table of user acceptability on recommended reviews.

3.3 Sentiment Analysis and Natural Language Processing

In the following discussion, we present some of the main definitions for sentiment analysis:

- *Natural language processing (NLP)*: NLP is in the domains of computer science, artificial intelligence, and computational linguistics, and is interested in the interactions between computers and human languages. Many challenges exist in NLP including natural language understanding, that is, enabling computers to deduce meaning from human or natural language input, and others involve natural language generation (Narayanan, Liu, and Choudhary, 2009).
- *Token*: Before any real processing can be done on the input text, it must be segmented into linguistic units such as words, punctuation, and numbers or alphanumeric. These units are categorized as tokens.
- *Sentence*: This refers to an ordered sequence of tokens.
- *Tokenization*: The operation of splitting a sentence into its constitutive tokens.
- *Corpus*: This means a body of text, usually including many sentences, and represents the entire dataset.
- *Part-of-Speech (POS) tag*: A word can be categorized into one or more of a set of lexical or part-of-speech classes such as Nouns, Verbs, Adjectives, and Articles, to name a few. A POS tag is a symbol representing such a lexical category – NN (Noun), VB (Verb), JJ (Adjective), AT (Article).

- *Computational Morphology*: Natural languages consist of a very large number of words that are constructed upon basic building blocks known as *morphemes* (or *stems*), these are the smallest linguistic units possessing meaning. Computational morphology which is interested in the discovery and analysis of the internal structure of words using computers.
- *Subjective Sentence*: It is a sentence in which the writer expresses his or her feelings or sentiments toward entities, events, and their properties. For example: "*I like small rooms*".
- *Objective Sentence*: It is a factual sentence about entities, events, and their properties, for example: "*The package contains free breakfast, laundry and wifi*."
- *Opinion words*: These are words that are commonly used to express positive or negative sentiment. For example: "Nice", "clean" and "spacious" are positive sentiments while "Ugly", "dirty" and "small" are negative sentiments.
- *Sentiment Orientation (SO)*: This is also called *Polarity*, and it indicates whether the expressed opinion by opinion words is positive, negative, or neutral. For example: "*Food was served on time*." has a positive polarity
- *Object / Features*: So far, we have used "amenities" to refer the main subject in hotel reviews. In opinionated documents, amenities and their components or attributes are going to be reviewed and sentiments toward them are expressed in terms of "opinion words"; these components or attributes are called: "object-features" (Nadkarni, Ohno-Machado, and Chapman, 2011). For instance, "*The rooms were kept clean*".

Feature	"*room*"
Object-feature (explicit)	"*clean*"
Opinion word	"*good*"

In this example: the explicit feature is "clean", but sometimes object features must be inferred from the sentence and are called *Implicit Features*. For example: "*The bathroom size is too small*":

Feature	"*bathroom*"
Object-feature (implicit)	"*clean*"
Opinion word	"*small*"

- *Computational linguistics*: This is an interdisciplinary field concerned with the statistical or rule-based modelling of natural language from a computational perspective. Computational linguistics has theoretical and applied components, where theoretical computational linguistics takes up issues in theoretical linguistics and cognitive science, and applied computational linguistics focuses on the practical outcome of modelling human language use.

3.4 Levels of Sentiment Analysis

Based on the levels of granularity of the previous research, sentiment analysis has been mainly investigated at three different levels: document level, sentence level and aspect level (Pang and Lee 2008; Liu 2012).

- *Document level*: Its task is to determine whether an opinionated document that comments on an object expresses an overall positive or negative opinion. For instance, a sentiment analysis system classifies the overall polarity of a customer review about a hotel. This level of sentiment classification assumes that one document expresses opinions on a single object, such as customer reviews of hotels and services, because usually the result of sentiment analysis only have two (positive and negative) or three outputs (positive, negative, and neutral). However, it is common that there might be a few different opinions in one document, thus it is not applicable to documents in which opinions are expressed on multiple aspects. There are several researchers who have carried out document-level sentiment analysis (Pang and Lee, 2008). They mainly focus on how to separate the positive texts from negative texts automatically and they also have presented different methods to improve the accuracy. Due to the elementary output of the sentiment classification, the major limitation of document-level sentiment analysis is the lack of in-depth analysis (Liu, 2012).

- *Sentence Level*: The sentence level of sentiment analysis involves determining whether each sentence expressed a neutral, positive, or negative opinion (Aue and Gamon, 2005). There is no major difference between document-level and sentence-level sentiment analysis, because the sentences are part of short documents (Liu, 2012). It usually contains two sub-tasks:
    1. Identifying whether the sentence is a subjective sentence or an objective sentence;
    2. If the sentence is subjective, identifying whether it expresses a negative or positive opinion (Liu, 2012). The subjectivity classification is quite integral, because it filters out those sentences that contain no opinions.
- The sentence level of sentiment classification works on an assumption that one sentence expresses a single opinion from a single opinion holder. However, Wilson *et al*. (2004) suggest that not only a single sentence may contain multiple opinions, but it can also contain both subjective and factual clauses. Thus, it is of great significance to highlight the factual clauses and identify the strength of sentiments. Giving a sentence a "neutral" classification usually indicates an objective sentence, or sentences absent of opinions. It is also worthwhile to note that subjectivity is not equivalent to sentiment, as Liu (2012) points out, because objective sentences can also imply sentiments, for example: "*Room was nice but the bathroom was not clean*". Those objective sentences that also imply opinions belong to one subset of opinionated sentences. As for compound sentences, they might be comparative or have grouped opinions about different aspects of hotel, thus sentence-level classification is not suitable for hotel reviews (Narayanan, Liu, and Choudhary, 2009).
- *Aspect Level:* Classifying opinions at document level or sentence level is useful in many cases, but can be insufficient to provide the necessary details needed for some applications, because they do not identify sentiment targets or assign opinions to these targets (Liu 2012). At the document level, a positive document on an object does not imply that the writer has positive sentiments on all aspects of this topic. Besides the sentence-level classification of sentiment is often an intermediate step since it is more useful to know what features or

entities of the object the opinions are on. Thus, the Aspect Level classification performs finer-grained analysis when needed. Aspect level classification is also called *Entity Level* or *Feature Level* in some studies (Hu and Liu, 2004; Pang and Lee, 2008).

The aspect level of sentiment analysis focuses on opinions itself instead of looking at the constructs of documents, such as paragraphs, sentences, and phrases. It is not sufficient to find out the polarity of the opinions; identifying the opinion targets is also essential (Steinberger *et al*., 2014). The aspect-level sentiment analysis can be subdivided into two sub-tasks: aspect extraction and aspect sentiment classification (Liu, 2012). The task of aspect extraction can also be an information extraction task, which aims to extract the aspects that opinions are on. For instance, in the sentence, "*the rooms of the hotel Empire are amazing but its service is too slow*". "*Service*" and "*Rooms*" are the aspects of the entity represented by the entity "*Hotel Empire*". The initial approach of extracting aspects is finding frequent nouns or noun phrases, which are defined as aspects. Later, the text containing aspects are classified as neutral, positive, or negative. (Blair-Goldensohn *et al*., 2008).

However, the issue can still arise in aspect-level sentiment analysis, as most of the studies are based on the assumption of the pre-specified aspects by keywords (Wang *et al*., 2011; Li *et al*., 2015). Ding *et al*. (2008) proposed a lexicon-based method for aspect analysis in which they assumed that aspects were known before the analysis. Liu (2012) points out that the accuracy at aspect level sentiment is still low because the existing algorithms still cannot deal with complex sentences well. Thus the aspect level sentiment analysis is more cumbersome than both the sentence-level classifications and document-level.

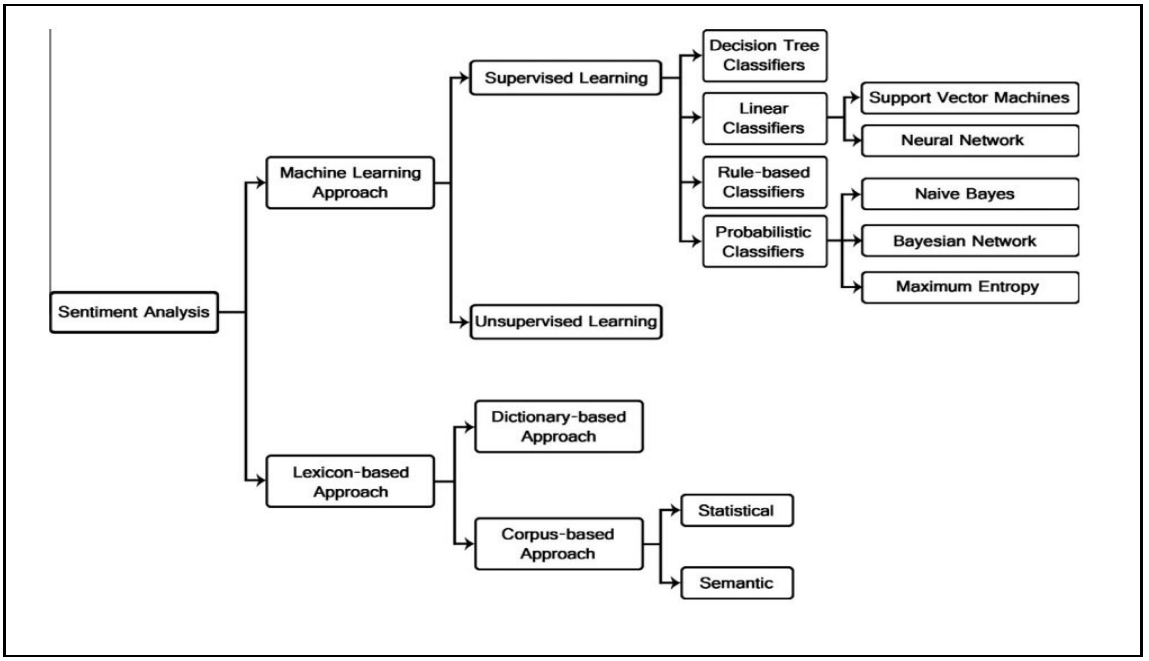


Figure 3 Sentiment classification techniques (Medhat *et al*., 2014)

3.5 Word Embedding

All machine learning algorithms and deep learning architecture ultimately work on numeric data. So it becomes very important to map textual data to numerical values. This process is called *Word Embedding*. It helps in classifying documents more accurately based on the polarity of the sentences, extracting important words and phrases. The classification techniques have evolved over time. In this section we have described three most popular models.

3.5.1 Bag-of-words (BOW) Model

This model (Zhang, Jin, and Zhou, 2010), is widely used in supervised text classification, a document is represented as an unordered collection of words disregarding grammar and even word order. According to this model, during the training phase, a dictionary is built on training data and is then used to characterize the relationship between the positive and negative documents in the testing procedure. For example, if we have the two documents below:

1. "*Rooms were very clean*"
2. "*Rooms were very small*"

The dictionary which is constructed based on BoW will be:

Feature Vector	Vector #
"Rooms"	1
"were"	2
"very"	3
"clean"	4
"small"	5

Hence, the feature vector of each document has "5 dimensionalities" based on the constructed dictionary. As demonstrate in sentiment analysis discussion, word appearance is very informative (in contrast with word frequency in information retrieval). The challenge of natural language is that sometimes one word can express the author's attitude clearly while a sequence of words cannot. The main disadvantages of this model are that it does not keep track of sequence of the words and all the words have same weight. For example, in previous example the feature is "*Rooms*" and opinion words are "*clean*" and "*small*". But this model distributes weight equally to other parts of the sentence thus suppressing the true opinion of the sentence.

3.5.2 Term Frequency–Inverse Document Frequency (TF-IDF) Model

By applying TF-IDF model we can roughly estimate what each document in the set of documents is all about, this model assigns weight to words or phrases based on their occurrence across the documents, and it also helps in identifying important terms in each document by checking its frequency against other documents. This is achieved by breaking the word vector in two metrics (Wu, Luk, Wong, and Kwok, 2008):

- **Term Frequency (TF)**: Term frequency is the ratio of occurrences of a term to total number of terms in a dataset.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- **Inverse Document Frequency (IDF):** is the inverse ratio of the documents that contain that word/phrase vs the total number of terms in a dataset.

IDF(t) = $\log_e$ (Total number of documents / Number of documents with term t in it).

TF-IDF fits perfectly well for this dissertation as it helps in identifying opinion's polarity effectively from a small dataset.

3.5.3 Word2vec

This is Prediction-based vector technique that provides probabilities to the words and proved to be state of the art for tasks like word analogies and word similarities. Word2vec is a combination of two techniques which works by creating a shallow neural networks which map terms to the target variable which is also a term. Both techniques learn weights which act as word vector representations:

- *Continuous bag of words (CBOW):* This predicts the probability of a term in each document. A document may be a single word or a group of words.
- *Skip-gram Model:* This follows the same steps as of CBOW. It just flips CBOW's architecture on its head.

Word2vec models can capture different sentiments of single words and being probabilistic adds more versatility to the output. But this technique requires a huge amount of data for training which is not feasible for this dissertation.

3.6 Key Challenges

Sentiment analysis is one of the toughest problems to be addressed in computer science. Identifying entities in unstructured formats is very hard for computers while it is easy for human beings. Below you can find some intractable situations for computers:

- Dealing with irony or sarcasm, it is difficult to understand that the opposite meaning of a sentence is required. Sometimes ironies can be recognized through special punctuation marks such as *"!!!*" but it is not sufficiently common to be a rule or sign for these types of expressions.
- Pronoun resolution is another complex task. Although there are some techniques and algorithms that can solve in certain cases, it is still demanding task in sentiment analysis. For instance; there are opinion words in a sentence but because the corresponding feature is a pronoun, it is not easy to find which

feature is expressed by those sentiment words, for example *"Rick found the passion of his life"* where *"his"* refers to Rick.

- Defining the *Strength* of an opinion also should be recognized as a complex task in this area. Opinions have different strengths. Some of them are very strong, e.g. "*The location was remote, staff was not very helpful, phone was not working, but it was worth the price and I liked it*" has positive polarity in the context of hotel review although it has lots of negative opinion words. Another difficulty is extracting the implicit keywords or features, to identify the expressed sentiment.

3.7 Conclusions

Sentiment analysis involves research in several fields; Natural Language Processing, Computational Linguistics, and Text Analysis. It refers to the extraction of subjective information from raw data, often in text form. However, also other media types could contain subjective data, like images, sounds and videos but these types are less studied. In accordance, in all media types different kinds of sentiments exist.

The classification of user reviews is a difficult task because review can contain irony, misspellings, emoticons, slang, abbreviations, and it may also contain only a few words. Let us consider the following review example: "*Nice restaurant, lovelyyyyyy meal, warm atmosphere, although the klutzy waiter spill vine on my dress: [*". This review contains an elongated word ("*lovelyyyyyy*"), a misspelled word ("*vine*" instead of "*wine*"), emoticon ("*:[*"), a slang term ("*klutzy*"), it also holds both positive and negative opinions. All these factors complicate the process of classification in this example. Various techniques exist that can be used for the sentiment analysis task. The main approaches are *Machine Learning* (Kim, 2014) and *Lexicon-Based* (Hailong, Wenyan, and Bo, 2014). The machine learning approach uses dataset for training classification which will be further applied for defining the sentiment of a text. The lexicon-based method uses the *Semantic Orientation* (SO) of words or phrases to define whether a text is positive or negative.

The sentiment can refer to opinions or emotions, however these two types are related there is an evident difference. In sentiment analysis based on opinions, a distinction is made between positive and negative opinions. On the other hand, sentiment analysis

based on emotions, is about the distinction between different kinds of emotions, like angry, happy, and sad. All this subtle information will help in building a sentiment analyser using machine learning algorithms. So it is very important to get the basics right to build a state-of-the-art text generator.

4. EXPERIMENT DESIGN

4.1 Introduction

This chapter starts by explaining the design methodology followed in the process of text analysis. Next it describes how the dataset was acquired and how the recent reforms in data protection laws that must be kept in mind while dealing with third party data sources. Later in the chapter, the experimental architecture is explained systematically. The final section explains why a certain technology was chosen out of many available choices that were implemented while creating experiment.

The experiments will provide evidence to explore the research question. The purpose of this chapter is to structure a proposed solution in a well-defined manner so that other researchers could also follow the procedure to generate similar results.

4.2. Development Methodology

The benefits of using development models are significantly greater in software than in other engineering disciplines because of the potential for a seamless link between models and the systems they represent. Machine learning deals with the automation of programs that improve their performance at some task through experience. Machine learning algorithms have significantly improved over time and are proving to be of great practical value in a variety of application domains. They are particularly useful for:

1. imperfectly understood problem domains where little knowledge exists for the humans to develop effective algorithms;
2. domains that deals with huge amount of data containing valuable implicit regularities to be discovered; or
3. domains where programs are subject to change quickly and must adapt to changing conditions.

Machine learning deals with the issue of how to build computer programs that improve their performance at some task through experience. Many times machine learning algorithms require reverse engineering, since we are always aware of the result but it can be hard to achieve in a single attempt. To achieve a state-of-the-art model, we must reiterate the entire model again and again until a satisfactory result is obtained. There

are various ways to implement data mining techniques based on the purpose of the project and the type of dataset. This project implemented the *Cross Industry Standard Process for Data Mining* (CRISP-DM). As CRISP-DM follows the exact process required in sentiment analysis tasks it perfectly fits the requirements of this research. The CRISP-DM methodology is series of hierarchical process models, consisting of four levels of abstraction (Wirth and Hipp, 2000):

1. *Phases*: This is the top-level process of CRISP-DM model, which divides the processes into different smaller and achievable phases.
2. *Generic Tasks*: Since CRISP-DM is a methodology, it must be generic enough to fit in with all the types of data mining task. This step consists of all the generic models and processes to cover all possible data mining situations. These tasks are designed to cover both the process of data mining as well as ability to fit in unforeseen development modelling techniques that will be developed in the future.
3. *Specialized Task*: This phase contains a detailed description of the actions that should be carried out to execute the tasks defined in second phase. For instance, in the second level there is a generic task called data pre-processing. At this level, we will have a task called the data cleansing model which contains activities specific to the dataset pre-processing.
4. *Process Instance*: This phase deals with gathering information and analysing the results obtained from the previous phases.

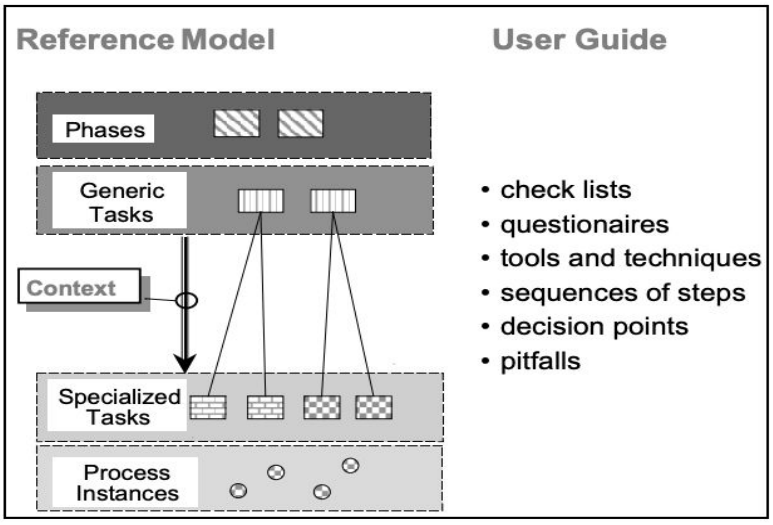


Figure 4: Phases of CRISP-DM methodology (Wirth, R., and Hipp, J., 2000).

Section 4.3, 4.4 and 4.5 contains a detailed implementation of Phase 1, Phase 2, and Phase 3 respectively. The experimental design and the results are covered later in Section 6 and Section 7.

4.3. Dataset Acquisition

The original idea was to scrape text reviews from TripAdvisor.com using their RESTful API. But due to recent reforms in data protection, TripAdvisor has stopped providing that data for academic purposes. Section 4.3.1 covers details of the new law and its amendments in depth. Section 4.3.2 describes the alternative approach through which the dataset was obtained.

4.3.1 The General Data Protection Regulation (GDPR)

The GDPR is a comprehensive approach to data protection by the European Union (EU) authorities. It forms the basis for allocation of responsibilities for ensuring lawful processing of personal data and for defining the applicable data protection legislation (Article 29, Europaeu, 2018). Article 5 (b) says that the controller shall implement mechanisms for ensuring that, by default, only those personal data are processed which are necessary for each specific purpose of the processing and are especially not collected or retained beyond the minimum necessary for those purposes, both in terms of the amount of the data and the time of their storage (Article 29, Europaeu, 2018).

According to Article 89(1) of General Data Protection Regulation (GDPR) data collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes (Europaeu, 2018); further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall not be considered to be incompatible with the initial purposes. Due to this reforms in data protection act introduced by European Union (European Commission, 2018) TripAdvisor has stopped providing data for academic research or data analytics (Request API Access, 2018).

4.3.2 Alternative approach

Each of the authors who described how to enhance a cold start hotel recommendation system published in their paper "Finding a Needle in a Haystacks of Reviews" (Levi, *et al*., 2012) were contacted. One of the author shared the dataset used in their research which was acquired from TripAdvisor before GDPR was implemented.

The dataset contains 106,266 hotel text reviews along with their sub-ratings individual scores ranging from 0 to 5 (whole numbers). The core idea of the system is to add five tags (Location, Hospitality, Food, Price, and Room) with scores from 0 to 5 (whole numbers) to every sentence present in text reviews. The process is divided into separate stages based on the technique we are applying. Initial stages deal with natural language processing techniques and later we applied CBR for text prediction.

Table 4 shows some sample data shared by the author. The column headers were missing from this dataset, which was a crucial part of the data. Without the headers it was hard to map which the amenities with their subsequent scores.

Hotel Milano Italy	4.2	Amazing service, nice location	Oct 23 2007	5	4	3	5	2
Hotel Grand Rome	3.6	Some text	Nov 03 2017	4	5	5	5	4

Table 4: Original dataset sample

But with the help of hotel name present in the dataset, the original review was tracked down from TripAdvisor. In there, review contain amenities scores which was then mapped with the dataset and accurate headers were obtained as shown in Table 5

Hotel Name	**Average Rating**	**Review**	**Date**	**Location**	**Hospitality**	**Room**	**Price**	**Food**
Hotel Milano	4.2	Amazing	Oct 23 2007	5	4	3	5	2

Italy		service, nice location						
Hotel Grand Rome	3.6	Some text...	Nov 03 2017	4	5	5	5	4

Table 5: Sample dataset with column names.

The original dataset provided was in compressed in a zip format. When unzipped, the dataset was stored in multiple Comma Separated Values (CSV) files. Each CSV file belonged to reviews of a hotel. All the CSV files were further merged into one CSV file that contained 106,266 rows and 9 columns. This dataset became the initial input to proposed experiment architecture described in Section 4.4.

Rating	Location	Hospitality	Rooms	Food	Price
1	1670	3725	3263	1495	3561
2	2945	3638	4370	2206	4388
3	8036	8436	10476	5847	9108
4	17635	19374	20928	14884	19374
5	31558	25413	22807	37412	25413

Table 6: Distribution of amenities scores across ratings in dataset.

Table 6 shows distribution of 5 categories obtained from the dataset. Most of the reviews belongs to 4 or 5 star ratings across all sub categories.

4.4 Experiment Architecture

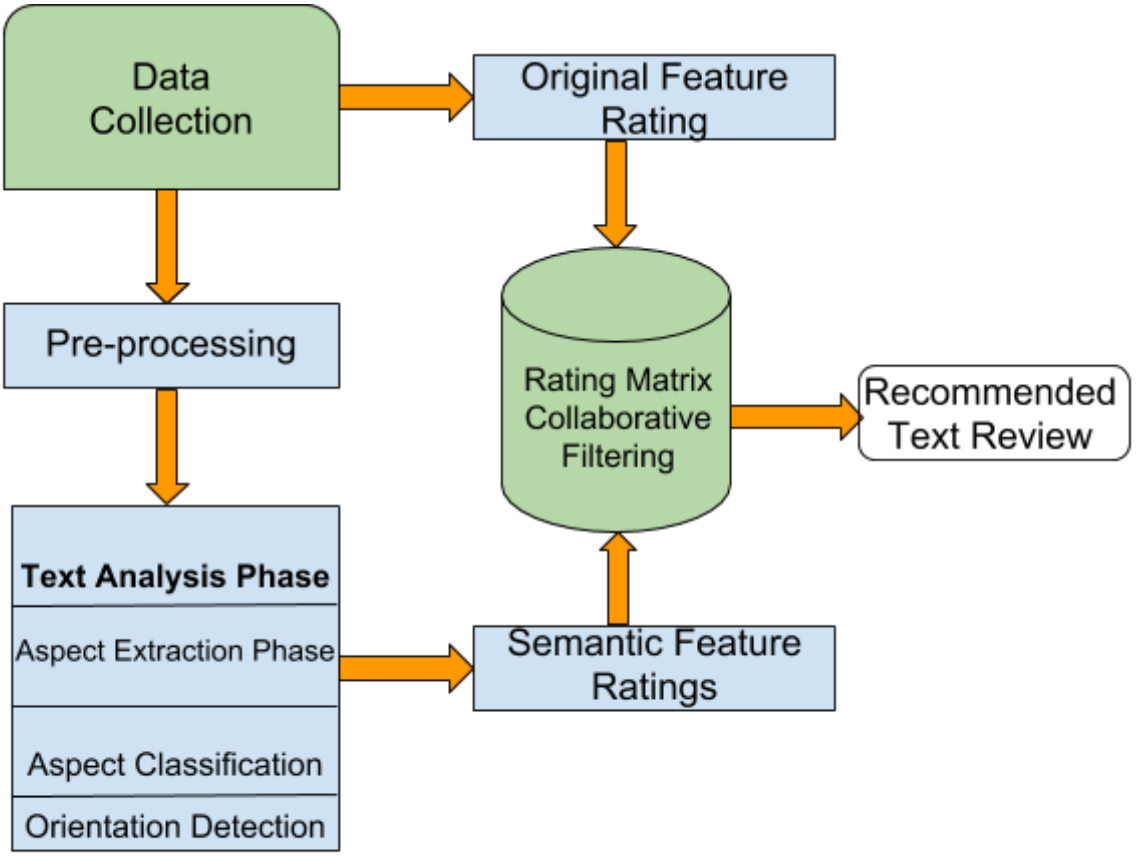


Figure 5 State Diagram of Proposed Design

The first step is to filter dataset by removing text reviews that contains less than 4 words, as explained previously under project description section in first chapter. After the dataset is cleaned, each review text will pass through a 3-stage process. Here is a brief overview of the components and steps of method, depicted in Figure 5. In Stage 1, the text review will be broken down into multiple sentences using the python Language Toolkit (NLTK) as S1, S2...Sn. Then for each sentence S1, sentiment analysis will generate scores for each sub-ratings based on presence of an Aspect Tag. An *Aspect Tag* is a set of synonyms used for sub-ratings in a review. This can be achieved by a *SpinGlass Community Detection Algorithm* as shown in Table 7.

Aspect Tag	Features Per Aspect
Location	*location, area, city, street, metro, station, train, distance, bus, airport.*
Hospitality	*staff, service, hotel staff, reception, front desk, luggage lobby, reception, staff, person, wifi.*
Room	*bathroom, floor, shower, size, window, door, view, building, tv, water, elevator, balcony.*
Price	*continental, dine, drink, fruit, breakfast, morning, food, restaurant, bar, coffee, buffet.*
Food	*price, value, star, money, rate, deal, quality, cheap, expensive, cost, pool, worth, penny, reasonable.*

Table 7: SpinGlass Community Detection Algorithm results (Levi & Mokryn, 2012, p.5)

This will serve as final output for present solution. For the proposed solution an additional step will be carried out in which the average of the sentiment scores and the sub-rating score given by the user will be tagged for each sentence. The final output will contain sentences labelled with sub-ratings scores. In this way two different outputs will be generated for each sentence. For each query, two cases will be retrieved using the *Euclidean Distance* formula, i.e. one from the proposed solution and one from the existing solution. An online survey will be conducted in which user will be presented with randomly generated scores from 0 to 5 against each sub-rating. Apart from the sub-rating scores the user will see two text reviews, out of which the user will be allowed to select only one of the texts based on what user thinks most closely resembles the sub-rating scores.

4.5 Experiment Stages

Stage 1 to Stage 5 are focused on the NLP techniques for sentiment extraction and words tagging. In Stage 6 we will engage a CBR technique to retrieve similar cases

from the past to match against the given input to produce output with similar sentiments.

4.5.1 Data Cleansing and Pre-processing

Data Cleansing: In this step we skimmed over the text reviews and filter out reviews that contains very few words, as it not be possible to classify words in which sub-category they belong to, e.g. if a text review says "*amazing hotel*", which amenity of the hotel amazed the user is hard to retrieve and hence were removed from the dataset. This reduced the dataset to 96,768 reviews. Also, some reviews did not contain rating scores, as this is an integral part of the process I eliminated reviews that did not contain sub-rating scores. This further reduced the data set to 88,128 text reviews.

The data next had to be pre-processed as it helps in reducing vocabulary clutter so that the features produced in the end are more effective. The data was passed step-by-step sequentially in order to clean it as follows:

- *Lower Case:* The first pre-processing step is remodel the text reviews into lower-case letters. This avoids having multiple copies of the identical words, for example, while calculating the word count, "*Clean*" and "clean" would be counted as different words otherwise.
- *Removing Punctuation*: The next step is to get rid of punctuation, because it doesn't add any additional information while treating text knowledge, thus removing all instances of it will facilitate a scale back of the dimensions of the training set.
- *Removal of Stop words*: Stop words (or commonly occurring words) should be removed from the text data. They are mostly articles prepositions which are used to construct structured sentence and have no sentiment associated with them. For this purpose, we can either create a list of stop words ourselves or we can use predefined libraries. For this dissertation the Python NLTK kit was used for removing stop words.
- *Rare Words Removal:* Now we'll remove seldom occurring words from the text. Because they're so rare, the association between them and other words is dominated by noise.

- *Spelling Correction*: While writing reviews users can unintentionally misspell words which can lead to noisy and redundant words. So it is quite essential to fix such misspells and thus reducing multiple copies of words. For example, "*Precarious*" and "*Precarious*" will be treated as different words even if the intent of the user was same. The Python Textblob library will be used to perform spelling correction on the dataset.
- *Tokenization*: In this step we will segregate remaining words from the sentence. These words will be passed further for Stemming/Lemmatization.
- *Lemmatization*: In this step we convert words into their root word, also called a "lemma". This step further helps in reducing the dataset. For example, "*Cleanliness*" and "*cleaning*" belongs to the lemma "*clean*" and they share same sentiment.

4.5.2 Morphological Analysis

Individual words are analysed into their components. Non-words tokens such as punctuations are separated from the words. This phase uses:

- *Part-of-Speech (POS) tagging*: We will assign a tag to each word in a text and classify a word to a morphological category such as noun, verb, and adjective. We will use Hidden Markov Models for developing POS tags.
- *Lemmatization*: As explained above this process converts all the inflected words present in the text into a root form called a lemma, e.g. "*diversity*", "*divergence*", and "*diverging*" are converted into the lemma "*diverge*".

4.5.3 Syntactic Analysis

A linear sequence of words is transformed into structures that show how words related to each other. Some word sequence may be rejected if they violate the language rules. This phase is further dissolved into two steps:

- *Feature Selection*: The purpose of this phase is identify noun, noun-phrases, adverbs, and adjectives which usually emphasise hotel amenities, e.g. a word occurring near a subjective expression can act as a feature. A clustering-based feature extraction technique will be implemented as they have high accuracy and require few parameters to tune.

- *Feature Cleansing*: Since the text could be very vague, many unnecessary features will be produced. The research is deliberately limited to three amenities, hence other features are redundant and will be removed. Redundant features will be eliminated by using a *Compactness Pruning Method*.

4.5.4 Latent Semantic Analysis

Now we have the dataset ready, we must classify text based on the terms it contains. Since the probability of terms with similar context is high, the "features" that are interrelated can be extracted out. The meaning of an individual sentence may depend on the sentences that precede it and may influence the meaning of the sentence that follow it. The stages in this process are:

- *Discourse Integration*: After receiving feature tags from State 3, we will identify sentences (before/after) that are talking about the sentences.
- *Pragmatic Analysis*: The structure representing what was said will be reinterpreted to determine what was meant. This will result in a numerical score for an individual amenity from 0 to 5, e.g. in the sentence "*The room was pretty cool*", cool describes that the room was very good. But if follow standard semantics of English language it means that the room was very cold and this leads to misinterpretation.

Using TF-IDF we can calculate the weight of each word which is then fed into a classification algorithm for distinguishing features. There are many classification algorithms, each of them have their own advantages and depends on the dataset that we are dealing with (Kotsiantis, Zaharakis, and Pintelas, 2007). Some common algorithms include:

- *Decision Trees*: These are trees that classify instances by sorting them based on a feature value calculated using TF-IDF. Each node in a decision tree represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values. Decision trees are usually univariate since they use splits based on a single feature at each internal

node. Since this research is working with more than one feature, Decision Trees are not suitable.

- *Perceptron-Based Techniques*: These can be briefly described as: If x_1 through x_n are input features values and w_1 through w_n are connected weights/prediction vector, then the Perceptron computes the sum of weighted inputs: $\sum x_i w_i$ and the output goes through an adjustable threshold: if the sum is above a threshold, the output is 1; else it is 0. This type of classification technique requires a lot of data and computation power which is beyond the scope of this thesis.

- *Statistical Learning Algorithms*: Conversely to the Perceptron-Based Technique, statistical approaches are characterized by having an explicit underlying probability model, which provides a probability that an instance belongs in each class, rather than simply a classification. K-nearest-neighbour (kNN) is also known as *Instance-Based Learning Algorithms*, is a widely-used implementation of this technique. kNN is based on the principle that the instance within a dataset will generally exist in close proximity to other instances that have similar properties. Since in this project, each text review is independent of others and each feature is independent of each other, kNN classification would not be suitable.

- *Linear Classifiers*: This classification technique is based on an assumption of independence among predictors. *Naive Bayes Classification* is a well-known implementation of a Linear Classifier. It assumes the presence of a given feature in a dataset is unrelated to the presence of any other feature and all of these properties individually contributes to the overall probability. This classification algorithm gave best result for the dataset that was used.

4.5.5 Preserving the Scores Matrix

This step will combine a user's actual amenity rating with the sentiment score generated in Step 4 using a Latent-Factor model. This models predicts a rating $r_{t,a}$ for a text review t and amenity a according to:

$$r(t,a) = o*B_u + B_s$$

where o is an offset parameter, B_u is the user's actual rating and B_s is the sentiment score obtained in Step 4. The product of o and B_u will lie within the range of -1 to +1. This is the Biased Factor (BF) we are trying to introduce in this research to increase the similarity of the CBR recommender system. The BF will dynamically change during the experimental phase and depends on the user acceptance.

4.5.6 Collaborative Filtering

In this phase we will identify similar text reviews based on other users' previous ratings obtained from Stage 5, e.g. if users Bob, Mary and Marley gave a 5-star rating to hotel for its location then when a user X writes reviews for any hotel X with 5 star ratings for location could possibly use similar words for describing hotel's location because the system identifies the reviews of Bob, Mary and Marley as being similar based on the ratings.

Table 8 shows a rating Matrix built for the recommender system. Characters A-Z symbolises the review text for a give category and score. When a new user comes and selects their desired ratings and category the system will pick the text that fits the coordinates in the matrix and present the result.

Aspect	**1**	**2**	**3**	**4**	**5**
Food	A	F	K	P	U
Hospitality	B	G	L	Q	V
Location	C	H	M	R	W
Price	D	I	N	S	X
Room	E	J	O	T	Y

Table 8: Text Matrix for Prediction System.

4.6 Technology Choices

4.6.1 Programming Language

Lot of activities in machine learning which are focused on domain are being carried out constantly. Since the domain is so vast each of available technology has excelled in a domain. Thus, leaving us with no simple answer to the "which language?" question. It depends on what we're trying to achieve, what techniques are involved in it and how the dataset is retrieved and structured. According to a survey conducted by towardsdatascience.com[36], Python is leading the pack, with 57% of data scientists and developers using it. 33% of the remaining are planning to migrate to use python based solutions in the future. Python is followed by R with 31% which is followed by Java, C/C++ and Javascript.

Machine learning scientists engaged on sentiment analysis ranks Python (44%), Java (15%), R (11%) and JavaScript (2%). Java is first choice of scientists on network security/cyber-attacks and fraud detection, contrary to which python is least preferred in these two domains.

Areas like linguistic communication process and sentiment analysis, developers go for Python that offers a better and quicker way to build highly performing algorithms, because of the in depth assortment of specialized libraries that are available in Python community (towardsdatascience.com, 2017).

C/C++ tops the chart when it comes to Artificial Intelligence (AI) in games and robot locomotion. As these languages works close to machine language they provide better level of control, high performance and efficiency required for glitch free software. While R is designed for statistical analysis and visualisations of big data, making it more helpful for business evaluation using charts and curves.

Since the project revolves around sentiment analysis and natural language processing Python becomes an obvious choice. It provides libraries like scikit for data processing, nltk for NLP, matlab for visualization and numpy for data manipulation. Availability of resource and the used case towards computational linguistics is the crux of option for Python as a language of choice.

4.6.2 Technique Choices

Clustering

In this technique recommendation problem is considered as an unsupervised machine learning task. We try to identify user groups and recommend each user in this group the same items. This technique works well when we have enough data, so that we can use clustering as the first step for shrinking the selection of relevant neighbors.

Deep learning approach for recommendations

Today they are applied in a wide range of applications and are gradually replacing traditional Machine Learning methods. It's a very challenging task to make recommendations for such a service because of the big scale, dynamic corpus, and a variety of unobservable external factors.

Case Based Reasoning (CBR)

It's a very elegant recommendation algorithm because usually, when it comes to matrix decomposition, we don't give much thought to what items are going to stay in the columns and rows of the resulting matrices. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns.

All the aforementioned techniques implement following two strategies:

Content-Based(CB): The system learns to recommend items that are like the ones that the user liked in the past. The similarity of text is calculated based on the features associated with the compared sub-categories. The content filtering approach creates a profile for each user or product to characterize its nature.

Collaborative Filtering (CF): In this technique, similarity in taste of two users is calculated based on the similarity in the rating history of the users. It is also referred as "people-to-people correlation." A major appeal of collaborative filtering is that it is domain free, yet it can address data aspects that are often elusive and difficult to profile using content filtering. Collaborative filtering suffers from what is called the cold start problem, due to its inability to address the system's new products and users.

The two primary areas of collaborative filtering are the neighbourhood methods and latent factor models.

Neighbourhood methods are centred on computing the relationships between sub-categories or, alternatively, between users. The item oriented approach evaluates a user's preference for an item based on ratings of "neighbouring" items by the same user. A product's neighbours are other products that tend to get similar ratings when rated by the same user.

Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users.

In this research we lack user information beforehand clustering technique doesn't seems to be plausible choice. The dataset used is static and limited to 50k reviews which makes it relatively small for Deep learning approach. Since our dataset can be easily represented as a matrix of users and sub-category ratings, CBR seems to be the perfect choice.

While recommending text reviews to the user we won't be having any user profile which makes CF as our choice of CBR strategy. The goal is to identify most similar rating of subcategories which other correlated users have given, which makes neighbourhood method a perfect fit.

4.7 Conclusions

In this chapter, the CRISP-DM methodology was explained. First all the steps involved in this methodology were explained and later it was justified why this methodology was chosen for this research. New amendments in GDPR were also explained that created problems while acquiring the dataset, followed by how the alternative option was chosen in order to acquire the dataset. After having the dataset, inspired by CRISP-DM, the experiment architecture was established. The experiment will be divided into six different stages that involve extracting sentiments from the dataset to store and present the results to the end user. Finally, in the last section it was justified why a certain technology was chosen in the aforementioned six stages of experiment stages. This chapter is a vital part of the thesis as the output obtained from it will be passed on for development and evaluation process.

5. EXPERIMENT DEPLOYMENT

5.1. Introduction

This chapter starts by describing how the CRISP-DM methodology was implemented for the experiment. Subsequent subsections of CRISP-DM cycle explains how the noise was handled from the data, how the dataset was filtered, how features and sentiments were extracted. It also looks at preserving the data and predicting the outcomes.

Later in the chapter, the design of an online survey is described. It also describes the criteria of volunteers to be chosen and how they were chosen. The approaches to gathering the results from the experiments are also explained. The results obtained from the experiment will help in answering the research question by providing us with qualitative and quantitative data.

The experimental setup to test the proposed hypothesis has to measure whether people are willing to select a new descriptor over the existing descriptor. User preferences with the proposed system results are measured with an online evaluation methodology. The experimental design (Knijnenburg, Willemsen, Gantner, Soncu and Newell, 2012) does not measure absolute user opinion but only relative user preference with one set of solutions over another. The experimental design is designed to measure the acceptance rate of system's output rather than accuracy.

5.2. The CRISP-DM Cycle

In CRISP-DM methodology bigger process is divided into smaller achievable phases. Each sub-phase can then be divided into smaller phases and so on. Top phases of experiment are:

1. Dataset Management: It involves, data cleaning, pre-processing.
2. Natural language processing: It involves morphological analysis, semantic and syntactic analysis, and text classification.
3. Similarity Matching: It involves applying collaborative filtering and preserving scores.

Initial dataset was stored in an excel file. It went through series of phases as detailed in the following subsections.

5.2.1 Data Cleaning and pre-processing

Since the dataset contains reviews written by actual users, it contained lots of subtle mistakes that needed to be fixed before the data could be used by machine learning algorithms. Entire dataset was loaded into Pandas dataframe, which is a python library for excel data manipulation. The loaded data frame was processed as below:

- **Filter Results:** The data set contained additional information about the hotel and reviewer such as 'Traveller', 'Nationality', 'Date', 'Service', that are not important for the purpose of the project. Hence, those columns were dropped from the dataframe. df.drop(remove_cols, inplace=True, axis=1), where *"df"* is the dataframe and *"remove_cols"* is the list of redundant columns. Later all the text reviews that contained less than 20 words were also removed. df[df['Review'].map(lambda x: isinstance(x, str) and len(x) > 20)]. There were few cases where sub-rating scores contained negative numbers which doesn't makes sense and hence such reviews were filtered out too. All the words were counted and the words that occurs the most and the least were removed. Since most occuring words are usually nouns, pronouns, preposition also called stop words have no purpose in machine learning algorithms and same with infrequent occurring word as shown in Figure 6.

```
stop.extend(['good', 'great', 'lovely', 'wonderful', 'milan', 'germany', 'pantheon', 'spanish', 'trevi', 'fountain', 'step', 'vatica
             'berlin', 'munich', 'rome', 'excellent', 'take', 'bahn', 'hotel', 'would', 'one', 'definitely', 'recommend', 'perfect',
             'again', 'ok', 'day', 'night', 'within', 'enjoy', 'love', 'beautiful', 'free', 'two', 'really', 'the', 'could', 'like',
             'get', 'also', 'highly', 'highli', 'stay', 'extremely', 'quite', 'everything', 'easy', 'loved', 'got', 'main', 'lot', '
             'although', 'review', 'want', 'emoji', 'use', 'see', 'though', 'etc', 'better', 'enough', 'english', 'vatican',
             'say', 'always', 'went', 'site', 'via', 'choice', 'overall', 'experience', 'block', 'italian', 'open', 'fantastic', 'fe
             'going', 'short', 'told', 'available', 'pleasant', 'bad', 'happy', 'i', 'couple', 'work', 'felt', 'still',
             'nights', 'mum', 'male', 'hey', 'umbrella', 'fifth', 'much', 'even', 'star', 'the', 'time', 'well', 'found', 'around',
             'best'
             ])
frequent_words = ['(', ')', 'u', "'"]
rare_words = ['polizzi', 'eurosto', "'shuttle", 'varioustv', 'femaies', "'q", 'tal/viktualienmarkt',
              'oberon', "'californian'by", 'slef', 'dailyair', 'es', 'oub', 'elina', 'oudour', 'montini', 'awau', 'towelswe', 'lltt'
```

Figure 6: List of rare words

- **Handle Special characters:** After scrutinizing the initial dataset, the text present in the reviews was observed. It was seen that users have used emojis to express their feelings such as sad smiley or happy smiley and a lot of other special characters such as @ for referring something, # for tagging and $ for price. These symbols have no importance for machine learning algorithms and unnecessary consumes time and space. Not all the special characters could be removed as that they can help in guessing the sentiments of the sentence. So, emojis were replaced with words and special characters such as $ was replaced with money as shown in Figure 7.

```
funny_characters_dict = {
    r'(&amp;)|(&)': 'and',  # Changing ampersand to and
    r'(:-?\))|(\(-?:)|(;-?\))|(\(-?;)|(:-?D)|(;-?D)': ' Happy ',  # Changing emoticons to actual meanings
    r'(:-?\()|(\)-?:)|(;-?\()|(\)-?;)': ' SAD_EMOJI ',  # Changing emoticons to actual meanings.
    r'([^\s]+)[0-9]([^\s]+)|[0-9]': ' ',  # Remove all numbers and words with numbers in them
    r'[!@$%^:;,-]': ' ',
    r'[0-9]': ' ',  # Remove numbers,
    r'/': ' or ',  # Remove numbers
    r'(  +)': ' ',
}
for regex, replace in funny_characters_dict.items():
    x = re.sub(regex, replace, x)
```

Figure 7: List of regular expression used to handle special characters.

- **Spell Check:** Review contained lot of spelling mistakes and were sanitized with a spelling correction library "*TextBlob*".

```
sentences = sent_tokenize(text)
for i in range(0, len(sentences)):
    words = word_tokenize(sentences[i])
    new_words = [str(TextBlob(word).correct()) for word in words]
    sentences[i] = ' '.join(new_words)
```

Figure 8: Lemmatizing words using TextBlob.

Text blob has its own limitations as it does not contains all the words and in some case spellings were so off that this library was unable to map it to any possible correct word.

5.2.2 Morphological Analysis

Since the data cleaning step is now complete, it is ready to pass it for NLP related tasks. This will help in identifying patterns and further grouping of words. This phase was completed by achieving follow subtasks:

- **First normal Form:** It was observed that there were lot of words that were used in there second or third forms or superlative degrees. Since this redundancy of words corrupts data while performing latent semantic analysis. It was essential to reduce such words to there first normal words. *NLTK* library was used to perform lemmatization as shown in Figure 9.

```
sentences = sent_tokenize(text)
lemmatizer = WordNetLemmatizer()
for i in range(0, len(sentences)):
    words = word_tokenize(sentences[i])
    new_words = [lemmatizer.lemmatize(word) for word in words]
    sentences[i] = ' '.join(new_words)
return ' '.join(sentences)
```

Figure 9: Lemmatization steps in NLTK library.

- **Part-of-Speech (POS) tagging:** For every sentence all the words are tagged to their corresponding POS. This helps in identifying important and words or phrases for feature selection and classification. Figure 10 shows a sentence with pos tagging.

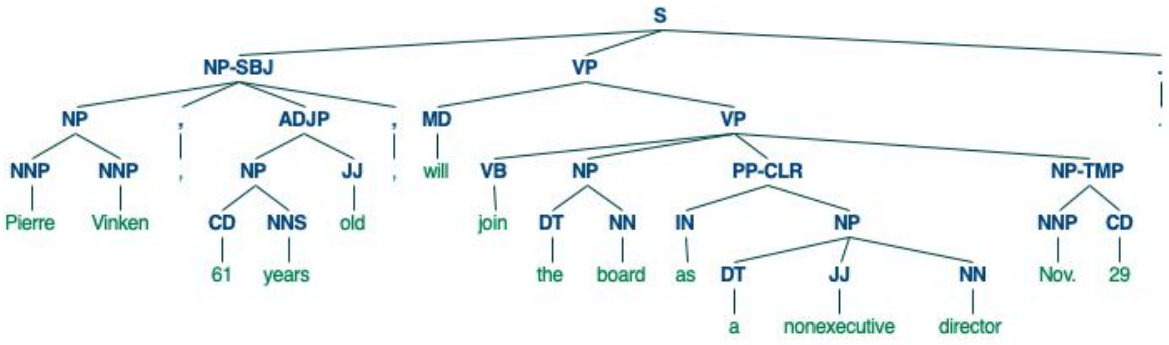


Figure 10: POS tag of a sentence.

5.2.3 Syntactic Analysis

The purpose of this phase is to extract features describing sub-category of the hotel. These extracted features will then be used by classification algorithms for grouping similar sentences. This phase was achieved in two steps:

- **Feature Selection:** Each of the POS tagged words were iterated and there adjacent words were stored in a dictionary. By the end of the iteration each word was associated with a list of adjacent words with their weights. Larger weight implies better coherence with a feature. The list was sorted in descending order and top 15 words were selected for each of the word. Finally a list of concept words was obtained as shown in Figure 11.

```
concept_words = {
    'location': ['peace', 'google', 'pharmacy', 'location', 'area', 'cit
y', 'street', 'metro', 'station', 'train', 'distance', 'bus', 'airport',
 'walk', 'close', 'stop', 'minute', 'centre', 'terminus', 'around', 'min
'],
    'hospitality': ['staff', 'service', 'reception', 'front', 'desk', 'l
uggage', 'lobby', 'reception', 'person', 'friendly', 'helpful', 'nice',
'help', 'friend', 'professional', 'polite', 'welcoming', 'attentive', 'c
omplain'],
    'food': ['continental', 'dine', 'drink', 'fruit', 'breakfast', 'morn
ing', 'food', 'restaurant', 'bar', 'coffee', 'buffet', 'dinner', 'fruit'
, 'terrace', 'variety', 'bread', 'course'],
    'room': ['filthy', 'refurbishment', 'modern', 'bathroom', 'floor', '
shower', 'size', 'window', 'door', 'view', 'building', 'tv', 'water', 'e
levator', 'balcony', 'room', 'lift', 'bath', 'space', 'bed', 'comfortabl
e', 'wifi', 'internet', 'piazza', 'air', 'small', 'spacious', 'noise', '
smallish', 'sleep', 'asleep'],
    'price': ['price', 'value', 'star', 'money', 'rate', 'deal', 'qualit
y', 'cheap', 'expensive', 'cost', 'pool', 'worth', 'penny', 'reasonable'
],
}
```

Figure 11: List of concept words.

- **Feature Cleansing:** Certain words weights equally for for two different concept words. This is possible as reviewers could write about two different features in one sentence, this give equal weight to its adjacent words. Compact pruning method was employed to identify the correct concept word using "*Tree.DecisionTreeClassifier"* library. It trains the model with predefined set of words and later scores the predictability of the words. Concept words used in the thesis achieved 83.37% accuracy.

5.2.4 Latent Semantic Analysis

In this phase all the sentences from text reviews are classified and tagged in their corresponding groups. TF-IDF vectorizer technique was implemented to generate latent semantic analysis. The process has following steps:

1. Fitting the data frame in TF Vectorize to generate term frequency of each word.
2. Inverse transform the fitted model to generate inverse document frequency
3. Multiply the two matrices to generate singular value decomposition.

The final output of this process is groups of text based on the number of value passed in third step. The value usually represents number of variables in the system, which in this case is five(hotel sub-categories). Implementing this steps results five groups of text corpus. These text represents sub-category reviews. Figure 12 shows Python implementation of LSA.

```
df = pd.read_csv('merged_file.csv')
dataset = df['clean']
vectorize = TfidfVectorizer()
sentence_list = []
for text in dataset:
    sentence_list.extend(sent_tokenize(text))
X = vectorize.fit_transform(sentence_list)
terms = vectorize.get_feature_names()
lsa = TruncatedSVD(n_components=5, n_iter=1000)
lsa.fit(X)
aspects = {}
for i, comp in enumerate(lsa.components_):
    comp_terms = zip(terms, comp)
    sorted_terms = sorted(comp_terms, key=lambda x: x[1], reverse=True)
    sorted_terms = sorted_terms[:45]
    aspects["Concept" + str(i)] = sorted_terms
```

Figure 12: Python implementation of LSA

5.2.5 Preserving the Scores Matrix

The final stage of Natural Language processing is to identify sentiment scores of text groups. In this phase all the sentences from every text groups were iterated and corresponding sentiment scores were assigned to them. For calculating sentiment scores *TextBlob Polarity* module was used. This module takes a sentence and predicts it polarity between 0 to 5, where zero represents negative sentiments and 5 means positive sentiments. While iterating through all the sentences certain information was stored in the database table as follows:

- **Aspect**: The sub-category or the group of the text.
- **Sentence**: The sentence in the current iteration.
- **User Rating**: The original rating given by the user for the review to which the sentence belongs to.
- **Sentiment Score**: The sentiment score obtained by Polarity module

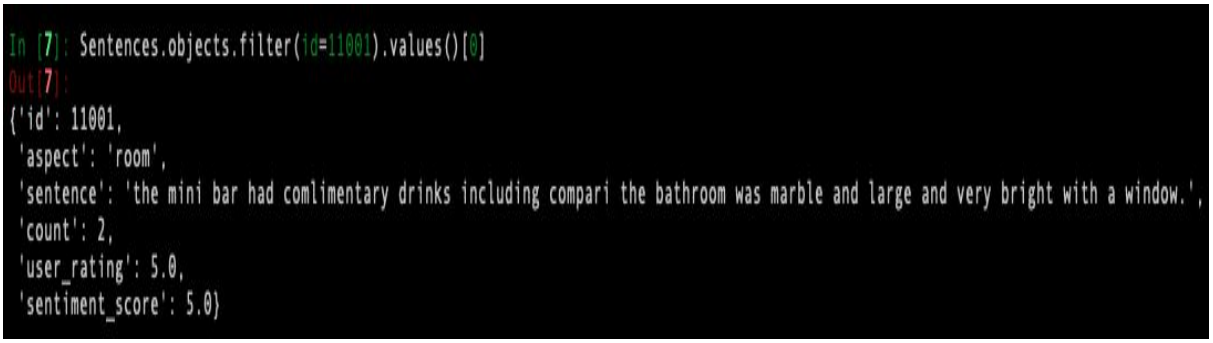


Figure 13: Sample of information stored in the database.

Figure 13 shows are sample sentence that belongs to *Room* sub-category who's user rating as well as sentiment score is 5.

5.3. Collaborative Filtering

After text classification and sentiment analysis, the next step of experiment was to generate text for a given sub-rating score. In order to achieve Collaborative Filtering technique was deployed. Following subsections describes the process in details.

5.3.1 Implementation Process

The correlation between the ratings of a hotel review as the similarity metric was used. To find the correlation between the ratings of the hotel review, a matrix was created where each column is a review name and each row contains the rating assigned by a specific user to that hotel review. As shown in Figure 14, inside red boxes represents empty value which will be predicted by the proposed system.

Ratings	**Room**	**Location**	**Hospitality**	**Food**	**Price**
1	T1	T1	T3	T4	T5
2	T6	??	T8	T9	??
3	T11	T12	??	T14	T15
4	T16	T17	T18	??	T20
5	??	Tn	??		??

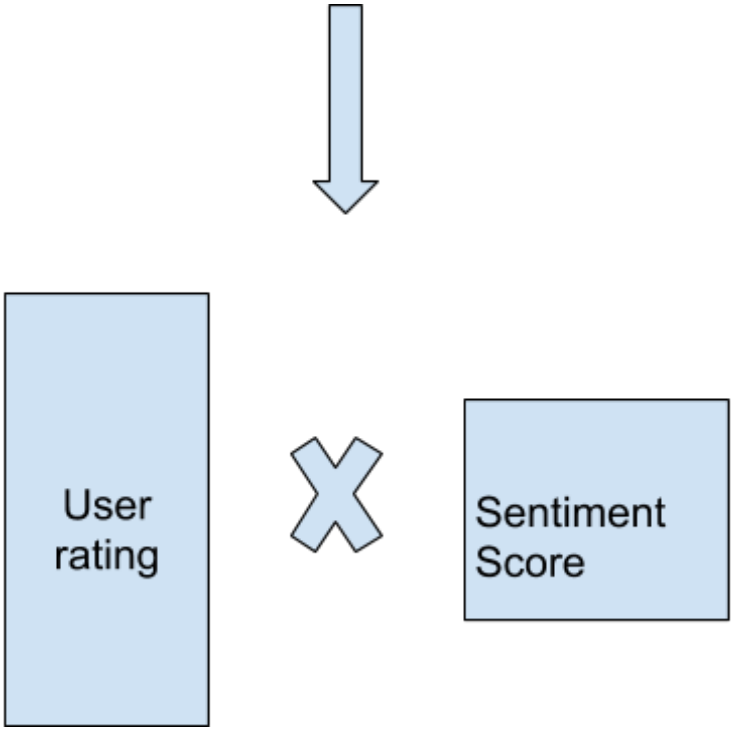


Figure 14: Dot product representation of CF Model.

Whenever a user issues a rating, the system will identify surrounding neighbours using kNN method. *Scikit* library was used as it provides a direct implementation of kNN algorithm. It takes sub-rating scores as the input and returns nearest text review associated to it.

5.4 Challenges Encountered

Firstly, many steps in the experiment cycles uses some predefined python libraries. These libraries are created for generic use, which sometimes fails to prove effective for specific use cases. For instance, for calculating text polarity TextBloB library was used, this library uses movies reviews for training. Words used in movies reviews could mean something different to words used in hotel reviews. It was hard to implement sentiment analyzer from the scratch as it is itself a huge project in itself.

Secondly, many iterations were carried out to achieve satisfactory results. Changing parameters in one step had cascading effects on the rest of the steps. Five variables were hard to manage, changing one variable could change entire outcome.

And lastly, the computer used in this research for the purpose of training and testing models had a low spec and it could take several hours to finish one experimental iteration. Since several iterations were carried out, most of the time was spent waiting for the tasks to complete. Running tasks in parallel for machine learning tasks in not feasible as all the data need to be loaded at once in the memory and hence results can't be shared across sub-processes.

5.5. Design

Reviews were generated in advance for a random set of categories ratings. Their output was saved and presented later in a questionnaire designed for this experiment. The implemented system was available publicly as a web survey.

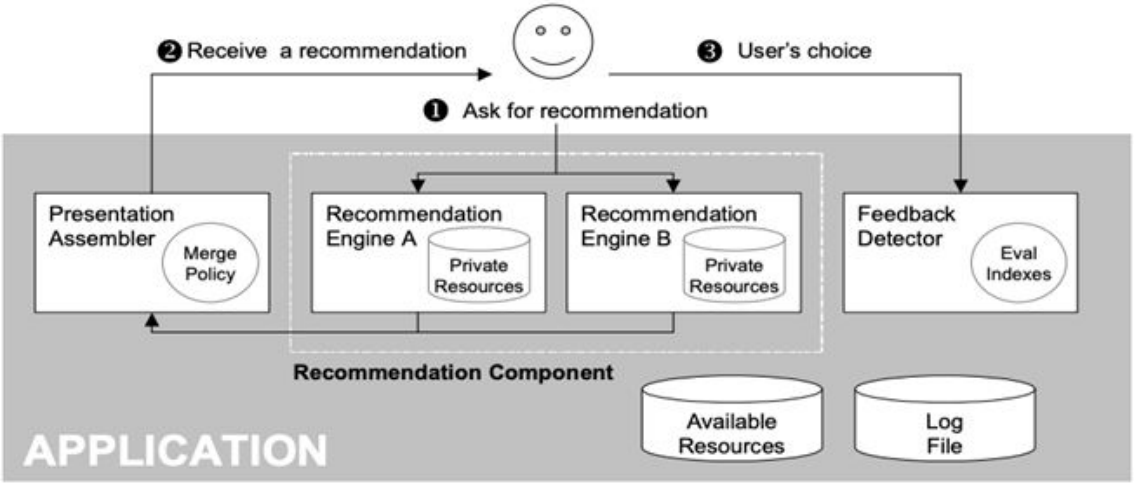


Figure 15 Architecture of experiment (Hayes, C., & Cunningham, P., 2002).

The survey contained four questions, each question consisted of the following:

- Review text called 'A' generated by proposed system
- Review text called 'B' generated by existing system
- A matrix of categories and their corresponding scores based on which review 'A' and 'B' were generated
- Multiple choice answers of which user could choose only one of them:
    - Review A
    - Review B
    - Both of them

- None of them

5.6. Recruiting volunteers

Looking at the results from a different angle is a very crucial part of the experiment evaluation. It brings a range of perspectives and backgrounds to the hypothesis. The experiment was curated in a way that the only requirement to be a volunteer is to be proficient in English. Since the recommendation system generates text in English language, the volunteers were selected based on their proficiency in English language. The survey was circulated to 30 colleagues and students, of which 22 were native English speakers and remaining 8 were non-native English speakers but proficient in English. The chosen volunteers participated in quantitative and qualitative questionnaires.

5.7. Questionnaires and Interviews Deployment

The experiment was divided into two steps,

1. an online survey for quantitative analysis and
2. an interview for gathering qualitative information. This section covers the two steps in details.

5.7.1 Online survey

All the 30 participants took part in online survey web. A survey was hosted on surveymonkey.com, which contained a consent form and a total of four multiple choice questions. Each question contained two text reviews generated from existing and proposed algorithms based on pre-selected categories ratings. The participant could choose either of the option, both of the options or none of them. The average time for completing the survey was between 5 to 6 minutes. Figure 16 shows a sample questionnaire used in the experiment. It contains two reviews, “Review A” and “Review B”, a table that contains category name in one column and category score in another column and list of options out of which a user can select any one.

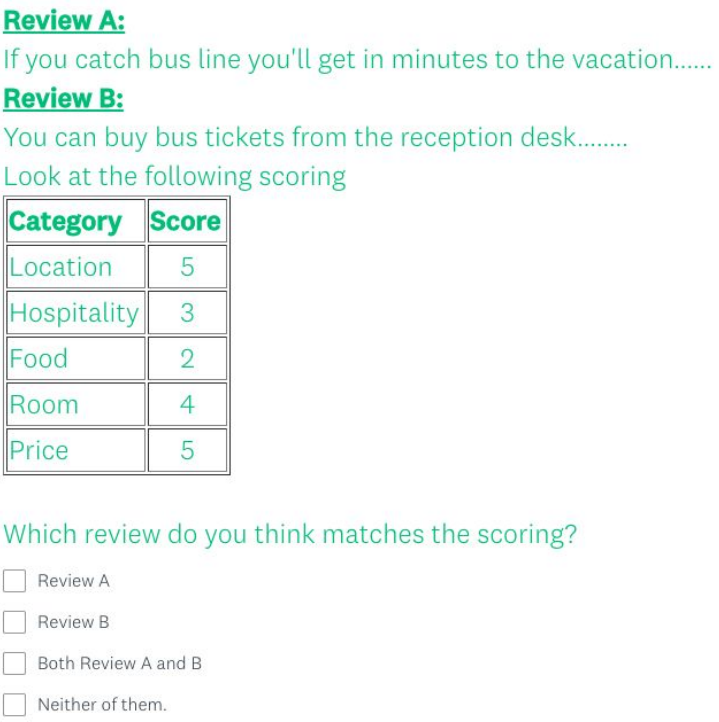


Figure 16 Sample of online survey created in surverymonkey.com

5.7.2 Interviews

In order to gain understanding about the survey, a verbal interview was conducted with 10 volunteers. Historical qualitative approach was followed in this process. Interviewers were first shown an online survey that gave them understanding about the scenario. Later a face to face interview was conducted with each of the participants separately. Based on their experience participants were asked a list of questions as follows:

- *Did you find the suggested text review helpful?*
- *Did you find survey easy to understand?*
- *Do you need an automated assistant that can help writing a review?*
- *Did you any thoughts/suggestions about this survey?*

All the 10 participants were native English speakers, out of which 2 were females and remaining 8 were males. 6 of them were frequent travellers and do read reviews of places and hotels online before travelling. 3 of them like to read movie reviews before watching them and remaining one likes to read books. Which makes everyone proficient in reading and understanding the English language.

5.8. Conclusions

This chapter starts by explaining three top level phases of CRISP-DM cycle used in the experiment creation. Each of the phase was divided into sub-tasks and so on. Data cleaning process was described and what sort of noise was found on the dataset and how it was cleaned was discussed in that section. Then morphological analysis was explained along with its use in future phases. Next, the features from the dataset was extracted which then helped in grouping text into their sub-categories. After grouping sentences, their polarity was calculated and the entire information was stored in the database. And at last phase of CRISP-DM cycle implementation of CF is explained.

The design of the survey was also discussed in this chapter. Two texts were generated using the proposed and existing algorithms and were stored in Recommendation engine A and B respectively. Later the stored texts were shown to the participants in the form of an online survey. The stored results will provide us with quantitative data to make conclusions about the hypothesis. An interview was also conducted to gather qualitative data.

The results obtained from the online survey was stored in a CSV file and the answers obtained from interviews were saved in a document file. This information will help us in concluding the research question precisely.

6. EVALUATION

6.1. Introduction

In this chapter the evaluation of the system is discussed, the first section explains the importance of evaluation and its various types. This helps in building a more efficient and robust evaluation process. The next section discusses how the performance of different classification algorithms is measured. Every classification algorithm has its own pros and cons, sometimes the selection of an algorithm can only be decided with the help of performance analysis. The third section compares the results of classification algorithms. The fourth section discusses patterns present in the user ratings and the sentiment ratings in the dataset. And at last, the results obtained from the experiments are explained in great detail.

6.2. Importance of Evaluation

Evaluation is an important issue for every scientific field and a necessity for software technologies like Case-Based Reasoning. However, evaluation is used differently in different contexts. At first, evaluation can be seen as evaluating one Case-Based Reasoning system, i.e. validating to which degree an application problem has been solved (Grogono and Batareh, 1991). Secondly, evaluation can be viewed as evaluating different Case-Based Reasoning Systems, i.e. comparing systems (Rothenberg and Drenth and Morris, 1992). Thirdly, the notion of evaluation can also be used for evaluating different development methodologies for Case-Based Reasoning systems, i.e. comparing system development methodologies (Hilal and Soltan, 1991).

The main idea is to combine different evaluation criteria:

- Domain and application task oriented criteria (e.g. size theory strength, openness).
- Technically and ergonomically oriented criteria (e.g. case and knowledge representation, similarity assessment, user acceptance).
- Knowledge engineering criteria (e.g. ease of use of methodology, development phase tools).

For the evaluation, the criteria were selected that produce quantitative or qualitative results, where the latter split into mainly domain-dependent ones as well as domain-independent ones.

The quantitative evaluation can also be considered as technically oriented criteria, e.g.:

- Case and knowledge representation
- Similarity assessment
- Handling of noisy and incomplete data
- Performance

The qualitative evaluation can also be seen as ergonomically oriented criteria, e.g.:

- User acceptance
- Adaptability
- Error management

6.3. Quantitative Evaluation

6.3.1 Area Under the Curve (AUC)

AUC is an integration technique used to measure the performance of machine learning classification algorithms. It is particularly useful in the task of predicting a review's positive or negative sentiment when it is a standard binary classification problem, however it is the computed probabilities that are most interesting when computing the AUC value. In some cases, a model with the higher AUC value might not have the highest accuracy, which could be something noteworthy, but the values are usually highly correlated. This evaluation statistic is nice as it supports not just binary class predictions but also probability predictions. Probability predictions always get a higher or the same score as the rounded binary predictions. The AUC curve plots the *False Positive Rate* (FPR) against the *True Positive Rate* (TPR), which are defined as:

$$FPR = FP / (FP + TN)$$

$$TPR = TP / (TP + FN)$$

where TP, FP , TN, FN are the number of *True Positives*, *False Positives*, *True Negatives* and *False Negatives*.

The TPR is plotted against FPR at many different thresholds (for example 0.00, 0.01, 0.02, ..., 1.00) which decides when a prediction is assumed to be true (e.g. at a threshold of 0.90 a prediction is assumed to be true if the computed probability is equal or higher than 90% and any prediction with a lower probability is assumed to be untrue).

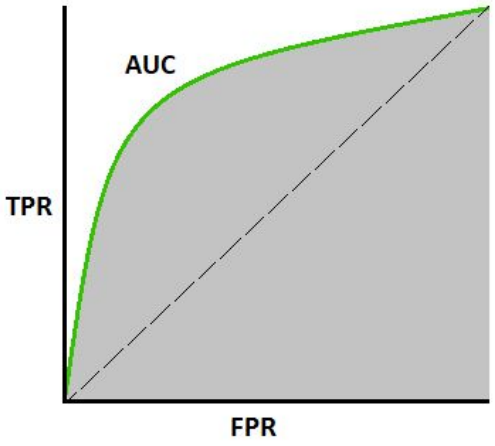


Figure 17: AUC plots False Positive Rate against True Positive Rate

A random prediction will result in an AUC value close to 0.5, which is usually used as a threshold. And if all the predictions are wrong the AUC value is 0 and if they are all correct the value is 1. In the case of binary predictions, the AUC value is the same as the accuracy.

For this research an AUC score of 0.82 was achieved, indicating that predictions using the proposed method were achieving a consistently high score.

6.3.2 Comparison of Approaches

For the classification data, a brute force method was used for evaluating the best classification algorithm. The dataset was split into 80% training and 20% test data. First the model was trained with the training dataset, then three different algorithms was evaluated on the test dataset. The three algorithms that were evaluated were:

- k-Nearest Neighbour
- Decision Tree
- Naive Bayes

The results of the tests show that Naive Bayes outperformed other machine learning techniques as shown in Table 9.

Classification Algorithm	Accuracy
k-Nearest Neighbour	78.23 %
Decision Tree	80.06 %
Naive Bayes	88.91 %

Table 9: Accuracy of various classification algorithms for hotel review dataset.

Since, the five variables in the experiment, i.e., location, hospitality, food, price, and room are independent of each other and may or may not occur in a text review, the Naive Bayes algorithm considers all variables independent of each other which made it the best classifier for the dataset used for the project.

6.3.2 Experimental Outcomes

This research is focused on comparing two approaches to text classification; the proposed method, and the existing method (the "Sentiment Method"), therefore using Naive Bayes for the proposed method, the two techniques were compared and they shared a similar degree distribution of reviews based on their sub-rating scores. A difference was seen on Five Star ratings with 10% gap in favour of our new Proposed Method (PM5) over the existing sentiment method (SM5) and in the case of the One Star ratings with a 5% gap in favour of the sentiment method (SM1) over the proposed method (PM1). A similar gap of 5% is seen in Two Star ratings in favour of the sentiment method (SM2) over the proposed method (PM2) This points in the direction that outcome of both the systems will differ somewhat on extreme star ratings, with the proposed method being more successfully on the higher extreme, and the sentiment method being more successful at the low end.

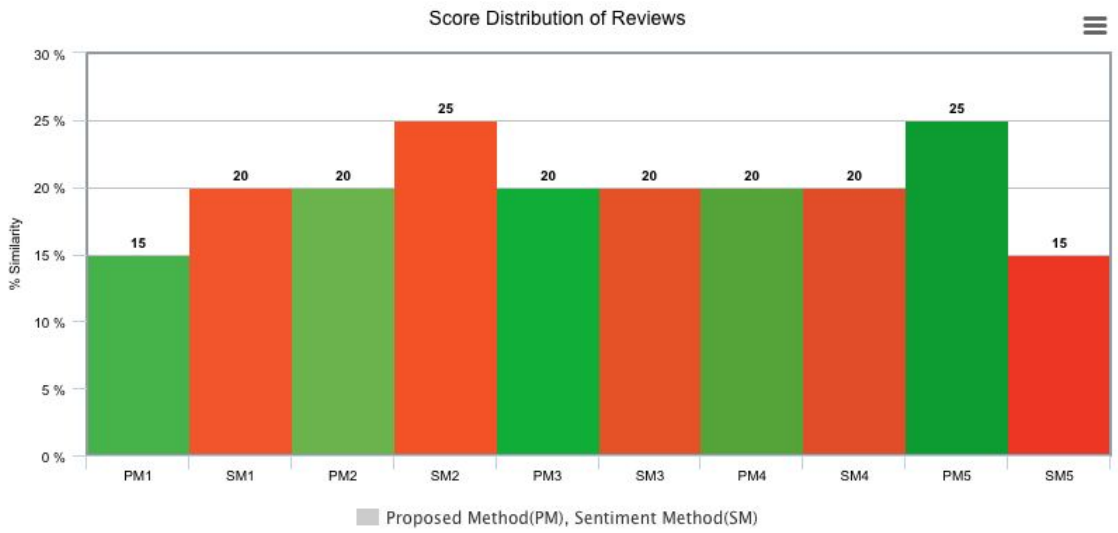


Figure 18: Review distribution based on sub-rating scores.

6.4 Qualitative Evaluation

Two types of methods were deployed to qualitative results, first this section describes the results of an online survey and later the outcomes of interview were discussed.

6.4.1 Survey Evaluation

As discussed in the experimental design chapter, an online survey was created. In the survey, four questions were asked and the participants could choose only one option per question. A total of 120 responses were collected from 30 participants and 4 questions each. Figure 6.4.1.1. shows participant's acceptance for the options, where:

- **Review A** represents reviews generated by the proposed method,
- **Review B** represents reviews generated by the existing method,
- **None of them** means participants didn't like either of the reviews and
- **Either of them** means both of the generated reviews fits the given rating matrix.

Results shows that the most preferred option was Review A with 46.73% followed by Review B with 28.19%. 15.11% participants choose either of the options to be applicable for a given rating matrix while 9.97% participants preferred not to choose any of the review suggested by either of the system.

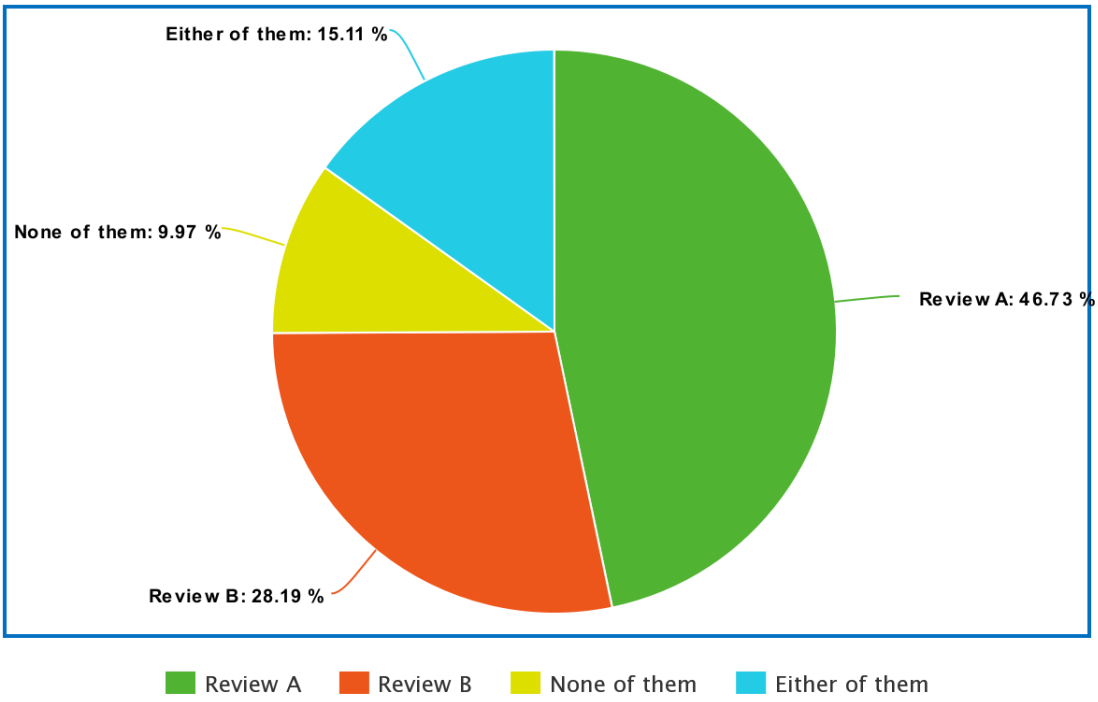


Figure 19: Options acceptance distribution by participants of the survey.

Interestingly, it was observed that 18.54% of the participants prefer the proposed system over the existing system. This is an important finding as it indicates using amenities scores with the sentiment scores predicts better review texts.

As observed in Table 10, both the systems present similar text when the sub-ratings lie between the range 3 to 4. When text reviews with such ratings were shown to participants 80.43% of them chose *Either of Them* as their choice, while 6.94% chose *Review A* and 8.11% chose *Review B*. This signifies that the proposed system shows no improvements when rating lies between a score of 3 or 4. Contrary to that, when reviews were generated from the sub-ratings scores ranging 1 to 3 or 4 to 5, 56.01% chose Review A and 28.29% chose Review B as detailed in Table 10.

Score Range	Review A	Review B	None of them	Either of Them
3-4	6.94%	8.11%	4.52%	80.43%
1-2 or 4-5	56.01%	28.29%	8.31%	7.39%

Table 10: Acceptance distribution based on sub-rating scores.

6.4.2 Interview Evaluation

A face-to-face interview was conducted with 10 participants, as discussed in Experimental Deployment chapter, a total of four questions were asked to these participants. Five participants found the suggested text reviews very helpful while 2 of them said they could use the suggested reviews as an assistance while writing actual reviews as shown in Figure 20. Six of the participants found the review easy to understand, although the estimated time to complete the survey was more than what was suggested by the online tool. Seven of the participants said they really like the suggestions and would like to see a real world application of the text predictor.

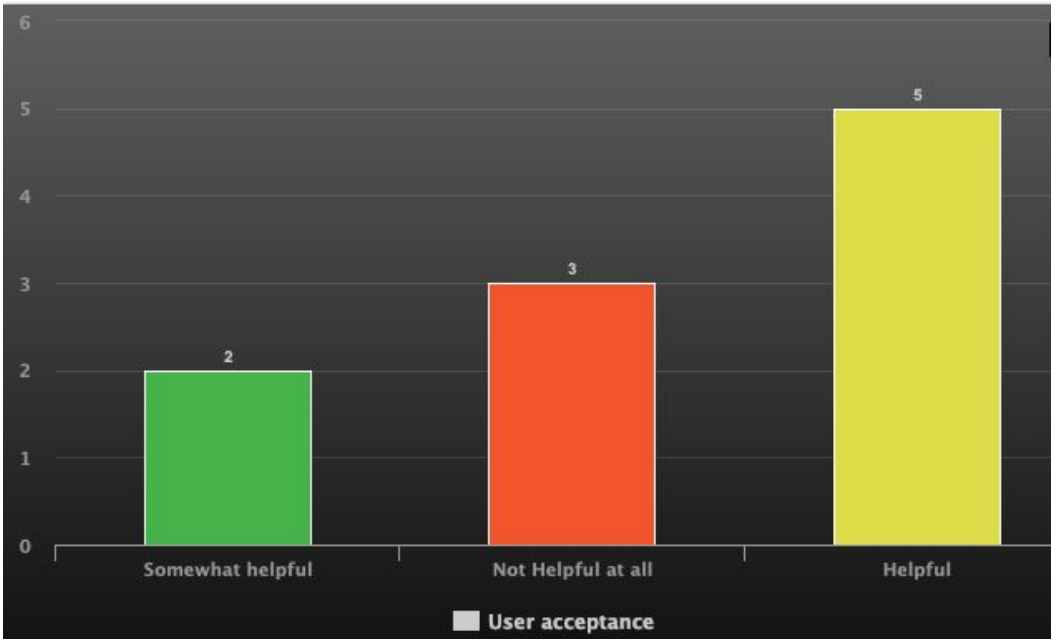


Figure 20: Participants response towards review's helpfulness.

Though the idea of the interview was clear to all the participants they found little difficulty in mapping sub-category score with the text predicted by the system. For instance, if a certain review says "*The food was expensive*", it is hard to understand whether this review is about the food category or the price category of the hotel. Participants suggested that instead of showing a paragraph of review text, it would have been easier if the sentences were grouped as per their sub-category.

Since the predicted text were generated using reviews written by humans and relate to an actual place or a hotel, two of the participants totally disliked the idea of text

prediction as the text generated were very specific to a hotel and its location. For instance, one of the reviews say "*Hotel Amber is the cheapest hotel in Berlin*", it is not feasible to reuse this sentence for describing another hotel or other location. The participants said it would be helpful if the system just suggested phrases, adjectives or verbs based on sub-categories scores.

6.5. Conclusions

The result evaluation is an integral step in answering the research question. From the experiment we gathered both qualitative and quantitative information. The chapter explained each of the gathered results in detail.

This chapter started by explaining the importance of experimental evaluation and different notions of evaluations. The Area under the curve, a mathematical approach to measuring performance of machine learning algorithm was detailed. Results of various classification algorithms were compared, out of which Naive Bayes outperformed all others. Later, a comparison between the users' ratings and sentiment scores was drawn and it was concluded that users can express their opinion more precisely if the rating scores lies in mid-range. Then the experimental evaluation using an online survey was discussed which shows positive results towards the hypothesis. Interview evaluation showed how the survey could be improved and the attitude of real world users towards the proposed system.

7. CONCLUSIONS AND FUTURE WORK

7.1. Introduction

There are many aspects to the research question "*Can hotel amenities scores that includes location, hospitality, price, food and rooms when combined with sentiment score of hotel's text review improves similarity of retrieval process in a CBR based hotel review recommendation system?"* The primary objective of the research was to study the effects of amenities scores on the retrieval process of a CBR-based hotel review recommendation system. In order to accomplish this, using NLP and CBR, a text recommendation system was developed. It was anticipated that when user sub-rating scores were combined with sentiment scores, the similarity of retrieval process improves. This chapter presents the summary of the findings using the quantitative and qualitative analysis as discussed Chapter 6. This chapter ends by proposing future work for the research presented and the technology used while performing the experiments.

7.2. Conclusions

This section will look at the objectives of the chapters discussed earlier in this dissertation. This will then be followed by the key findings of each chapter. This section ends by summarizing the conclusions of all the sub-sections and conclusion for the hypothesis.

7.2.1 Recommender Systems and Case-Based Reasoning

The first objective of this chapter was to develop a basic understanding of Recommender Systems and the factors that affect a Case-Based Reasoning system. The second objective was to point out the potential areas of improvement by using the Collaborative Filtering technique with respect to this research. It was concluded that with the help of a CBR-based system, a text recommendation system can be built.

7.2.2 Sentiment Analysis

Since NLP is a significant domain, the primary objective of this chapter was to narrow down the scope which aligns with the research question. This chapter also explores the past, the present and the future of Natural Language Processing for text analysis.

Another key point was to explain the steps involved in the NLP task. The limitations of current technology were also explained which then helped in building the experiment within the bounds of available technologies.

7.2.3 Experiment Design

Every software design follows a design methodology, this chapter explains a novel approach to building a machine learning program using the CRISP-DM methodology. Also, some problems faced due to reforms in the GDPR rules while gathering dataset are also addressed. The architecture of the proposed solution and its stages were also explained.

7.2.4 Experiment Development

This chapter explained the experimental process from the initial stage to the deployment. First, the phases of the CRISP-DM cycle adjusted for this research were explained with some code samples. Later, the design and deployment of the survey were explained. The participant demographics were also discussed along with a list of questions asked to them. This stage provided qualitative and quantitative data for the evaluation process.

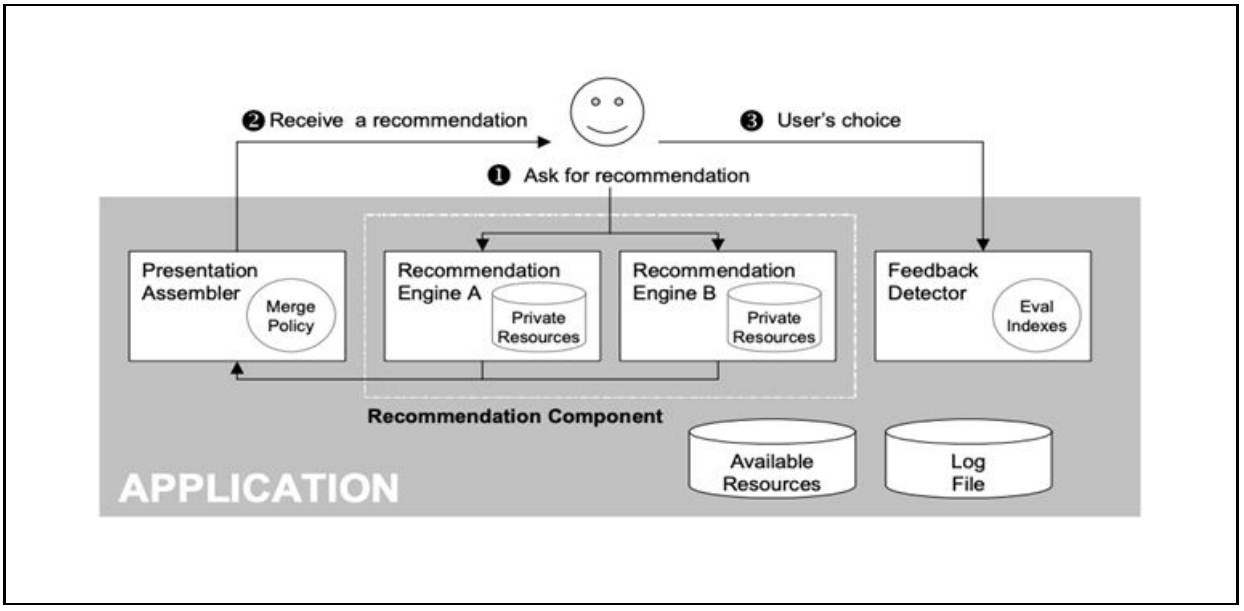


Figure 21: System Overview.

7.2.5 Evaluation Process

The objective of this chapter was to discuss the results obtained from the online survey and the face-to-face questionnaire. The performance of certain classification algorithms were also discussed. The results obtained by the evaluation process provided firm evidence in relation to the research question.

7.2.6 Overall Conclusions

From the quantitative analysis, most of the participants chose the text generated by the proposed method with a difference of 18.54% between the text generated by the proposed system and the text generated by the existing system. The results obtained from the online survey indicate that the similarity of retrieval process of a CBR-based hotel review recommendation system is increased by combining sub-category scores with the sentiment scores of the hotel's text reviews.

It was also concluded that when the subcategory scores lie in mid-range i.e. 3 or 4, the existing system outperforms the proposed system by 1.83%. The difference is seen significantly in favour of the proposed system when the subcategory scores are chosen from extreme ends i.e. 1 or 2, and 4 or 5.

The results gathered by the qualitative analysis suggest that 60% of the participants are satisfied with the generated text.

7.3. Future Work

This project uses Machine Learning and Natural Language Processing which are two subdomains of the Artificial Intelligence. There are still lots of stones unturned in this area, the project uses existing tools and technology available. Since, the resources (dataset and machine capability) were limited there were many techniques that were not used. It is an important issue to address and research can continue by increasing the available resources, which will open new possibilities of techniques and technology.

7.3.1 Recommendations for Similar Experiments

In context with the work presented in this dissertation there are many possible areas that can be expanded:

- There are many neural network techniques available for text classification and language generation which were not used in the project due to limitations of the dataset. They have many advantages over techniques that were used in terms of performance and predictions.
- The sentences generated by the algorithm were directly selected from the corpus. Those sentences contained some grammatical errors and usually contained names of a hotel or place.
- Standard libraries such as "TextBlob" were used for calculating the sentiment score of a sentence. It would be great to implement a personalised sentiment analyser for the dataset. This could result in more precise ratings.
- The dataset used in this dissertation contains only text in the English language. The outcomes could be different for different languages, as each language has its own grammar and requires different techniques to handle it.
- There are five variables addressed in the dissertation, further variables could be uncovered and used.

7.3.2 Recommendations for Technologies Incorporating this Research

- NLP tends to be based on turning natural language into machine language, but with time as the technology matures – especially the AI component –the computer will get better at "understanding" the query and start to deliver answers rather than just search results.
- Language is a huge barrier when it comes to communication with non-native English speakers. With the help of AI techniques an earbud could be build which can translate any language in real time.
- With the help of AI techniques, it could be possible to automatically analyse documents and other types of data in any business system which are subject to GDPR rules. It will allow users to search quickly and easily, retrieve, flag, classify and report on data mediated to be very sensitive under GDPR.
- NLP models require existing data to produce results, these results could become monotonous after some time. A system could be built that constructs sentences based on grammar and vocabulary.

- Many times the meaning of a word changes based on the context which is not possible to understand with the current techniques. For instance, "*The comedian killed the show*", in this sentence the verb "killed" signifies that the comedian performed the best which means that sentence has a positive sentiment. But, "killed" in categorised as a bad world and current NLP technique will produce negative sentiments for this sentence.

8. BIBLIOGRAPHY

Aamodt Agnar, & Plaza Enric. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(1), 39–59. https://doi.org/10.3233/AIC-1994-7104

Araujo, L. (2004). Symbiosis of Evolutionary Techniques and Statistical Natural Language Processing. IEEE Transactions on Evolutionary Computation, 8(1), (pp. 14–27). doi:10.1109/tevc.2003.818189

Aue, A., & Gamon, M. (2005, September). Customizing sentiment classifiers to new domains: A case study. In Proceedings of recent advances in natural language processing (RANLP) (Vol. 1, No. 3.1, pp. 2-1).

Basile, V., & Bos, J. (2011). Towards generating text from discourse representation structures. 11 Proceedings of the 13th European Workshop on Natural Language Generation (pp. 145-150). https://dl.acm.org/citation.cfm?id=2187705

Blair-Goldensohn, S., Hannan, K., McDonald, R., Neylon, T., Reis, G. A., & Reynar, J. (2008, April). Building a sentiment summarizer for local service reviews. In WWW workshop on NLP in the information explosion era (Vol. 14, pp. 339-348).

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. Knowledge-Based Systems, 46, 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

Bobadilla, J., Serradilla, F., & Bernal, J. (2010). A new collaborative filtering metric that improves the behavior of recommender systems. Knowledge-Based Systems, 23(6), 520–528. https://doi.org/10.1016/j.knosys.2010.03.009

Bock, K. (1996). Language production: Methods and methodologies. Psychonomic Bulletin & Review, 3(4), (pp. 395–421). doi:10.3758/bf03214545

Breese, J.S., Heckerman, D., & Kadie, C.M. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. CoRR, abs/1301.7363.

Bridge, D., & Healy, P. (2012). The GhostWriter-2.0 Case-Based Reasoning system for making content suggestions to the authors of product reviews. Knowledge-Based Systems, 29, (pp. 93–103). doi:10.1016/j.knosys.2011.06.024

Charniak, E. (2001). Immediate-head parsing for language models. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL ’01. Association for Computational Linguistics (pp. 39-42). doi:10.3115/1073012.1073029

Cowie, J., & Lehnert, W. (1996). Information extraction. Communications of the ACM, 39(1), (pp. 80–91). doi:10.1145/234173.234209

Debnath, S., Ganguly, N., & Mitra, P. (2008). Feature weighting in content based recommendation system using social network analysis. In Proceeding of the 17th international conference on World Wide Web - WWW ’08. ACM Press (pp. 259-266). doi:10.1145/1367497.1367646

Ding, X., Liu, B., & Yu, P. S. (2008). A holistic lexicon-based approach to opinion mining. In Proceedings of the international conference on Web search and web data mining - WSDM ’08. ACM Press. https://doi.org/10.1145/1341531.1341561

Douglas E. Appelt, (1992). Logics of conversation. Studies in natural language processing. Cambridge University Press (pp. 99-126). https://isbnsearch.org/isbn/9780521438032

Europaeu. 2018. European Commission - European Commission. [Online]. [17 December 2018]. Available from: https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en

Faridani, S. (2011). Using canonical correlation analysis for generalized sentiment analysis, product recommendation and search. In Proceedings of the fifth ACM conference on Recommender systems - RecSys ’11 (pp. 121-138). ACM Press. https://doi.org/10.1145/2043932.2044005

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61–70. https://doi.org/10.1145/138859.138867

Gong, Y., & Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR ’01 (pp. 441-448). doi:10.1145/383952.383955

Hailong, Z., Wenyan, G., & Bo, J. (2014). Machine Learning and Lexicon Based Methods for Sentiment Classification: A Survey. In 2014 11th Web Information System and Application Conference. IEEE. https://doi.org/10.1109/wisa.2014.55

Hayes, C., & Cunningham, P. (2002). An on-line evaluation framework for recommender systems. Trinity College Dublin, Department of Computer Science.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. ACM Transactions on Information Systems, 22(1), 89–115. https://doi.org/10.1145/963770.963774

Huand, .M & Liu, .B, (2004). Mining and summarizing customer reviews. ACM SIGKDD international conference on Knowledge discovery and data mining , (pp. 168-177). doi:10.1145/1014052.1014073

Huang, Z., Chen, H., & Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems, 22(1), 116–142. https://doi.org/10.1145/963770.963775

J. Kolodner and D. Leake. Case-Based Reasoning: Experiences, Lessons and Future Directions, chapter A Tutorial Introduction to Case-Based Reasoning, pages 31–65. MIT Press, 1996.

Jakob, N., Weber, S. H., Müller, M. C., & Gurevych, I. (2009). Beyond the stars. In Proceeding of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion - TSA '09. ACM Press (pp. 81-92). doi:10.1145/1651461.1651473

Jay J. Jiang & David W. Conrath (1997). Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In the Proceedings of ROCLING X, Taiwan, 1997 (pp. 32-39). https://arxiv.org/abs/cmp-lg/9709008v1

Jin, R., Chai, J. Y., & Si, L. (2004). An automatic weighting scheme for collaborative filtering. In Proceedings of the 27th annual international conference on Research and development in information retrieval - SIGIR '04. ACM Press. https://doi.org/10.1145/1008992.1009051

Kevin, L., Goldensohn, L.B, & McDonald, R., (2009). Sentiment summarization: evaluating and learning user preferences. In EACL '09: Proc. of the 12th Conference

of the European Chapter of the Association for Computational Linguistics. ACL, Morristown, NJ, USA, (pp. 514–522). https://dl.acm.org/citation.cfm?id=1609124

Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2007). A large-scale classification of English verbs. Language Resources and Evaluation, 42(1), (pp .21–40). doi:10.1007/s10579-007-9048-2

Knijnenburg, B. P., Willemsen, M. C., Gantner, Z., Soncu, H., & Newell, C. (2012). Explaining the user experience of recommender systems. User Modeling and User-Adapted Interaction, 22(4–5), 441–504. https://doi.org/10.1007/s11257-011-9118-4

Ko, Y., & Seo, J. (2000). Automatic text categorization by unsupervised learning. In Proceedings of the 18th conference on Computational linguistics -. Association for Computational Linguistics (pp. 114-119). doi:10.3115/990820.990886

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160, 3-24.

Lamontagne, L., & Lapalme, G. (2004). Textual Reuse for Email Response. In Lecture Notes in Computer Science (pp. 242–256). Springer Berlin Heidelberg. doi:10.1007/978-3-540-28631-8_19

Lange, T., Roth, V., Braun, M. L., & Buhmann, J. M. (2004). Stability-Based Validation of Clustering Solutions. Neural Computation, 16(6), (pp. 1299–1323). doi:10.1162/089976604773717621

Levi, A., Mokryn, O., Diot, C., & Taft, N. (2012). Finding a needle in a haystack of reviews. In Proceedings of the sixth ACM conference on Recommender systems - RecSys '12. ACM Press (pp. 41-48). doi:10.1145/2365952.2365977

Levine, E., & Domany, E. (2001). Resampling Method for Unsupervised Estimation of Cluster Validity. Neural Computation, 13(11), (pp. 2573–2593). doi:10.1162/089976601753196030

Lindstrom, L., & Jeffries, R. (2004). Extreme Programming and Agile Software Development Methodologies. Information Systems Management, 21(3), 41–52. https://doi.org/10.1201/1078/44432.21.3.20040601/82476.7

Liu, B., Hu, M., & Cheng, J. (2005). Opinion observer. In Proceedings of the 14th international conference on World Wide Web - WWW '05. ACM Press. https://doi.org/10.1145/1060745.1060797

Liu, B. (2012). Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies, 5(1), 1–167. https://doi.org/10.2200/s00416ed1v01y201204hlt016

McDonald, D. D. (1983). Description directed control: Its implications for natural language generation. Computers & Mathematics with Applications, 9(1), (pp. 111–129). doi:10.1016/0898-1221(83)90011-1

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal, 5(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Mooney, R. J., & Bunescu, R. (2005). Mining knowledge from text using information extraction. ACM SIGKDD Explorations Newsletter, 7(1), (pp. 3–10). doi:10.1145/1089815.1089817

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. Journal of the American Medical Informatics Association, 18(5), 544–551. https://doi.org/10.1136/amiajnl-2011-000464

Narayanan, R., Liu, B., & Choudhary, A. (2009, August). Sentiment analysis of conditional sentences. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1 (pp. 180-189). Association for Computational Linguistics.

Nasukawa, T., & Yi, J. (2003). Sentiment analysis. In Proceedings of the international conference on Knowledge capture - K-CAP '03. ACM Press (pp 44-53). doi:10.1145/945645.945658

Navathe, Shamkant B., and Elmasri Ramez, (2000), “Data Warehousing And Data Mining”, in “Fundamentals of Database Systems”, Pearson Education pvt Inc, Singapore, (pp .841-872). https://dl.acm.org/citation.cfm?id=1855347

Neuhuttler, J., Woyke, I. C., & Ganz, W. (2017). Applying Value Proposition Design for Developing Smart Service Business Models in Manufacturing Firms. In Advances in Intelligent Systems and Computing (pp. 103–114). Springer International Publishing. https://doi.org/10.1007/978-3-319-60486-2_10

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval, 2(1–2), (pp. 1–135). doi:10.1561/1500000011

Plaza, E., (2008). Semantics and Experience in the Future Web. In Lecture Notes in Computer Science (pp. 44–58). Springer Berlin Heidelberg. doi:10.1007/978-3-540-85502-6_3

Pustejovsky, J., (1991). The generative lexicon. Computational Linguistics archive Volume 17 Issue 4, December 1991 (pp. 409-441). https://dl.acm.org/citation.cfm?id=176324

Pustejovsky, J., & Boguraev, B. (1993). Lexical knowledge representation and natural language processing. Artificial Intelligence, 63(1–2), (pp. 193–223). doi:10.1016/0004-3702(93)90017-6

Request Api Access https://developer-tripadvisor.com/content-api/request-api-access/

Samuel Brody , Noemie Elhadad, An unsupervised aspect-sentiment model for online reviews, Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, June 02-04, 2010, Los Angeles, California (pp. 804-812). https://dl.acm.org/citation.cfm?id=1858121

Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the tenth international conference on World Wide Web - WWW ’01. ACM Press. https://doi.org/10.1145/371920.372071 2018

Schank, R. C. (1982). Dynamic memory: A theory of reminding and learning in computers and people (Vol. 240). Cambridge: cambridge university press.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval-SIGIR '02. ACM Press (pp. 55-68). doi:10.1145/564376.564421

Smyth, B., & McClave, P. (2001). Similarity vs. Diversity. In Case-Based Reasoning Research and Development (pp. 347–361). Springer Berlin Heidelberg. doi:10.1007/3-540-44593-5_25

Soderland, S. (1999). Machine Learning, 34(1/3), (pp. 233–272). doi:10.1023/a:1007562322031

Steinberger, J., Brychcín, T., & Konkol, M. (2014). Aspect-level sentiment analysis in czech. In Proceedings of the 5th workshop on computational approaches to subjectivity, sentiment and social media analysis (pp. 24-30).

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-Based Methods for Sentiment Analysis. Computational Linguistics, 37(2), (pp. 267–307). doi:10.1162/coli_a_00049

Tatemura, J. (2000). Virtual reviewers for collaborative exploration of movie reviews. In Proceedings of the 5th international conference on Intelligent user interfaces - IUI '00. ACM Press (pp. 272-275). doi:10.1145/325737.325870

The research hub of the app economy, providing fact-based insights and developer tool benchmarks (2017). Retrieved from https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7

Wang, J., de Vries, A. P., & Reinders, M. J. T. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '06. ACM Press. https://doi.org/10.1145/1148170.1148257

Watson, I., & Marir, F. (1994). Case-based reasoning: A review. The Knowledge Engineering Review, 9(4), 327. https://doi.org/10.1017/s0269888900007098

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05. Association for Computational Linguistics. https://doi.org/10.3115/1220575.1220619

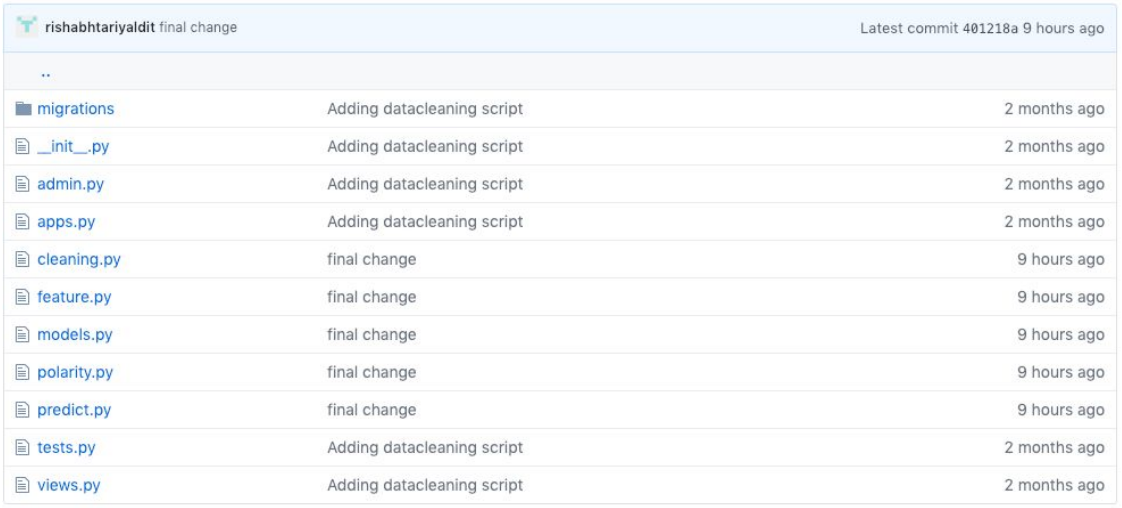
Wirth, R., & Hipp, J. (2000, April). CRISP-DM: Towards a standard process model for data mining. In Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining (pp. 29-39). Citeseer.

Wu, H. C., Luk, R. W. P., Wong, K. F., & Kwok, K. L. (2008). Interpreting TF-IDF term weights as making relevance decisions. ACM Transactions on Information Systems, 26(3), 1–37. https://doi.org/10.1145/1361684.1361686

Zhang, Y., Jin, R., & Zhou, Z.-H. (2010). Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics, 1(1–4), 43–52. https://doi.org/10.1007/s13042-010-0001-0

Zhou, X., Xu, Y., Li, Y., Josang, A., & Cox, C. (2011). The state-of-the-art in personalized recommender systems for social networking. Artificial Intelligence Review, 37(2), 119–132. https://doi.org/10.1007/s10462-011-9222-1

APPENDIX A: CODE STRUCTURE



rishabhtariyaldit final change		Latest commit 401218a 9 hours ago
..		
migrations	Adding datacleaning script	2 months ago
__init__.py	Adding datacleaning script	2 months ago
admin.py	Adding datacleaning script	2 months ago
apps.py	Adding datacleaning script	2 months ago
cleaning.py	final change	9 hours ago
feature.py	final change	9 hours ago
models.py	final change	9 hours ago
polarity.py	final change	9 hours ago
predict.py	final change	9 hours ago
tests.py	Adding datacleaning script	2 months ago
views.py	Adding datacleaning script	2 months ago

The code is hosted on Github.com. It can be cloned using *"https://github.com/rishabhtariyaldit/recommender.git"*. The directory structure is as follows:

- migrations - it contains database scripts that creates tables in a database.
- admin.py - admin web page related controls
- cleaning.py - data preprocessing and cleaning scripts
- feature.py - feature extraction and similarity assessment scripts
- models.py - database class definition
- polarity.py - sentiment scores calculation logic.
- predict.py - collaborative filtering logic
- tests.py - test cases for ReviewBot
- views.py - html view for rendering information.

Model definition

```python
from django.db import models

# Create your models here.


class Sentences(models.Model):
    LOCATION = 'location'
    HOSPITALITY = 'hospitality'
    FOOD = 'food'
    ROOM = 'room'
    PRICE = 'price'
    ASPECT_CHOICES = (
        (LOCATION, LOCATION),
        (HOSPITALITY, HOSPITALITY),
        (ROOM, ROOM),
        (FOOD, FOOD),
        (PRICE, PRICE),
    )
    aspect = models.CharField(choices=ASPECT_CHOICES, max_length=10)
    sentence = models.TextField()
    count = models.IntegerField()
    user_rating = models.FloatField(null=True, blank=True)
    sentiment_score = models.FloatField(null=True, blank=True)
```

Data Cleaning Steps

```python
def clean_data(df):
    print("Removing funny characters.....")
    df['Review'] = df['Review'].apply(replace_funny_chars)
    print("Removing stop words.....")
    df['clean'] = df['Review'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
    print("Applying lemmatization.....")
    df['clean'] = df['clean'].apply(apply_lemmatization)

    print("Applying stemming.....")
    df['clean'] = df['clean'].apply(apply_stemming)

    print("Removing stop words.....")
    df['clean'] = df['clean'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))

    print("Removing frequent and infrequent words that are not important.")
    df['clean'] = df['clean'].apply(lambda x: " ".join(x for x in x.split() if x not in frequent_words))
    df.to_csv('merged_file.csv', index=False)
    return df
```

Calculating sentiments and saving to database

```
def analyze_sentiment(aspect):
    for sentence_obj in Sentences.objects.filter(aspect=aspect, user_rating__lte=3):
        sentence = sentence_obj.sentence
        polarity = TextBlob(sentence).polarity
        if polarity <= -.1:
            score = 1
        elif polarity <= 0:
            score = 2
        elif polarity <= .1:
            score = 3
        elif polarity <= .3:
            score = 4
        else:
            score = 5
        sentence_obj.sentiment_score = score
        sentence_obj.save()
```

Merging multiple csv to one dataset file

```
def merge_files():
    all_files = glob.glob(file_path + "/*/*.csv")
    list_ = []
    frame = pd.DataFrame()
    headers = ['Traveller', 'Nationality', 'Rating', 'Review', 'Date', 'Cost', 'Room', 'Location', 'Cleanliness', 'Hospitality'
    for file_ in all_files:
        try:
            df = pd.read_csv(file_, index_col=None, header=0, names=headers)

            list_.append(df)
        except:
            print('invalid file %s' % file_)
    frame = pd.concat(list_, sort=False)
    return frame
```

APPENDIX B: ONLINE SURVEY

Online survey was hosted on survermoney.com

Sample survey question

Review A:

"Very straightforward getting bus or tram to where all the sights are but we walked everywhere after that best way to get the real feel of Rome couple of sightseeing tips if you only have days in visiting st peters or vacation do a tour that will concentrate on the main areas.. there was no wireless in the room but it was free with a strong signal in the lobby.. it s walkable from termini but easier to take the metro to the nearby stop.great service (people at the front are extremely helpful sad emoji very clean funky decor.. by far the biggest and nicest hotel room i have stayed in in europe where small and over rated is the norm in my experience.. staff was ok. never tried the restaurant because it was crazy expensive."

Review B:

"The location is just great minutes from the pantheon and piazza Navona.. the hotel staff was excellent and could arrange tour for you during the day.. when i confronted the hotel staff they said there is nothing they can do.. later when we asked if the room had a bath (not a bathroom) she said yes with a sluggish eye roll.. the hall was smelly like a sewer the walls were paper thin you can hear your neighbour s conversation and tv the towels were thin like a sheet the shower was moldy and very tiny the front desk clerks were unfriendly the breakfast was disgusting oz bowls for a poor choice of cold cereals hard rolls w no flavour unsweetened or unflavoured yogurt awful juice that was messy to deal with and very poor coffee from machines that you might find at a gas station or carwash...an insult really of a breakfast..this hotel was very expensive for what we received."

Look at the following scoring:

Category	Score
Location	5
Hospitality	5
Food	2
Room	2
Price	1

Which review do you think matches the scoring?

- ☐ Review A
- ☐ Review B
- ☐ Both Review A and B
- ☐ None of the above

Sample question two

Review A:

"However the location is terrible.. wherever you stay in Berlin though you will at some point need to make use of the excellent public transport system euros will get you a day ticket with unlimited use.tips on food the self service Vapiano across the road serves excellent pizzas pastas and salads for a very reasonable price.. large elaborate breakfasts very helpful and friendly staff.. on arrival the entrance is small dark and dirty.. the bar in the hotel is expensive the wifi is expensive euro for a half hour."

Review B:

"my room had a broken toilet flusher incredibly hard bed finicky tv and incredibly loud drunk neighbour stomping around at am.. staff was great communication was excellent.. i would recommend the cappuccino that the staff is offering during the breakfast...it was one of the best i have tried in Italy.. the rooms was filthy greasy yellow netting hung limply at the window stained sofa and worktop cracked bathroom.. i got a little tired of salami sandwiches after days and wanted my usual oats but all we were served was very good.make note that the restaurant at the foot of the entry to the hotel is rather expensive even with the discount you receive as a hotel guest."

Look at the following scoring:

Category	Score
Location	1
Hospitality	5
Food	5
Room	1
Price	1

Which review do you think matches the scoring?

- ◯ Review A
- ◯ Review B
- ◯ Both Review A and B
- ◯ None of the above