TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A MAJOR PROJECT REPORT ON

HYBRID BOOK RECOMMENDER SYSTEM

By:

Anil Paudel (072/BEX/404)

Deepak Kandel (072/BEX/417)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

ACKNOWLEDGEMENT

We would like to thank the Department of Electronics and Computer Engineering, IOE for giving us this great opportunity to carry out a major project in the field of recommender system. Among all the members of the department, our special thanks goes to the Head of Department **Dr. Surendra Shrestha**, and the Deputy Head of Department, Mrs. Bibha Sthapit, for providing us the support for this major project. We express our deepest gratitude to **Dr. Bashanta Joshi** for his constant guidance and **Mrs Ranju Kumari Shiwakoti** and for supervising this project with full dedication, support and guidance. We would like to acknowledge all the authors of the research papers and the tutorials found on the web that helped us understand the required concepts and algorithms better.

Sincerely,

Anil Paudel (072/BEX/404)

Deepak Kandel (072/BEX/417)

ABSTRACT

Recommender systems are one of the most successful and widespread application of machine learning. The primary aim of this project is to build a book recommender system that provides personalized book recommendation for each user. The machinelearning algorithms in recommender systems are typically classified into two categories – content based and collaborative filtering. Both methods have their pros and cons. Content-based filtering require rich description of items and well organized user profile before recommendation can be made to users whereas collaborative filtering suffers from the cold start problem. Hybrid method uses the best of both of these methods but they tend to be complex. The goal of this project is to implement a hybrid recommender system in its simplest form without losing much quality in its recommendation. Furthermore, it is important to account both implicit and explicit feedback from the user. In this regard, the sentiment on the reviews of the books are analyzed and they are taken into account along with rating and clicks on particular book to generate a weighted average mean rating that is used in user-item interaction matrix for generating recommendation by method of matrix factorization. Thus, the objective here is to eliminate the need of quality descriptive item data and adequate information about a user in order to make relevant recommendation.

Keywords: Recommender system, Content-based filtering, Collaborative filtering, Hybrid method, Implicit Feedback, Explicit Feedback, Sentiment Analysis, Matrix Factorization.

1. INTRODUCTION

This project entitled '**Hybrid Book Recommender System'** is a platform where users get personalized recommendations of the books that they have not read but which they might be interested in. One of the main motivation for us to work on this project is the fact that direct explicit rating carry less information and are, in a way, a form of information repetition and redundancy. A rating of four may sound fairly good for some users but others may take the same rating harshly.

In this regard, we are willing to analyze the sentiments of reviews made by users on each book and turn that into rating. Moreover, users hardly give rating explicitly whether it is in the form of stars or reviews. So, we have also taken the user clicks on a particular book into account. Therefore, the weighted average mean of user clicks, rating and reviews are taken with priorities in the order of review, rating and user clicks with review carrying the highest priority and this average mean is inserted into the user-item interaction matrix. The sparse interaction matrix is then fed by predicted rating provided by the method of matrix factorization. The loss has been optimized by use of stochastic gradient descent. Finally, n books having the highest predicted rating are recommended to the users.

1.1. Background

The huge growth in the amount of digital information to the Internet has created a challenge of information overload due to which people are not able to get timely access to items of interest on the Internet. Recommender systems are information filtering systems that handles the problem of information overload by filtering vital information fragment out of large amount of dynamically generated information. The use of recommender system ensures that a user gets good and useful recommendation. Moreover, it has become common for enterprises to collect large volumes of transactional data that allows for deeper analysis of how a customer base interacts with the space of product offerings. Recommender systems have evolved to fulfill the natural need of buyers and sellers by automating the generation of recommendations based on data analysis.

The area of recommender system are under active research intersecting several sub disciplines of statistics, machine learning, data mining and information retrievals. Researchers all over the world are continuously trying to improve the performance of such system with the ultimate target being the improvement of user experience over the internet.

1.2. Problem Statement

Most of the recommender system that are in use today makes use of content-based and collaborative filtering which have their pros and cons. The need of quality descriptive data of an item and adequate information about a user are the respective cons of these approaches. Researchers have stepped up to join best bits of both approaches which is now popular as hybrid system but they tend to become fairly complex. On the other hand, a recommender system can't rely on one of the feedback given by users. A study regarding the explicit rating found that the Amazon customers leave feedback on the seller experience only about 15% of the time. Clearly, there is need to address implicit feedback as well. The main motive of this project is to incorporate both rating (star rating) and reviews along with user clicks so that there is combination of both implicit and explicit feedback from user part. Moreover, we also attempt to merge metadata of book: author, genre and description to generate similarities within the books addressing

cold-start problem so that proper recommendation is provided even when the user doesn't have much activity under his profile.

1.3. Scope

The importance of this project is depicted by the sudden explosion of researches on the topic of effective recommender in the last decade. Online shopping are becoming increasingly popular all over the world. They might possibly even dominate the trade system of the world in the near future. Those systems require a good recommender system to be popular and famous among their competitors. Recommender Systems are an integral part of the success of Amazon, bringing more than 30% of revenues, and Netflix, where 75% of what people watch is from some sort of recommendation.

Talking specifically about this project, it focusses on accounting all three formats of user feedback: rating, reviews and user clicks, converting the reviews into rating of its own kind by sentiment analysis, and then, calculating the weighted average mean of ratings and finally using this rating in the user-item interaction matrix to generate recommendations. To address the cold-start problem, the metadata of the books: author, genre and description shall be considered to find the similarity among the books. A website has been developed in e-commerce style where users can create an account, provide feedback on books and get personalized recommendation.

2. OBJECTIVES

The primary goal of this project is to build a recommender system that provides personalized book recommendations to users and to build a website whereby the users can give feedbacks on the books and get recommendations accordingly. The concrete functional objectives of this project are as follows:

- To account all three forms of user feedback: star rating, review and clicks for generating recommendations.
- To develop a recommender system that incorporates book metadata: author, genre and description to find similarities among the books to address cold-start problem.
- To build a website with descent authentication where users can provide feedback on the books and get personalized recommendation.

3. LITERATURE REVIEW

It is vital to account every feedback from the user for delivering quality recommendation. Often, the reviews on particular item are taken as of lower priority than direct rating. Business analysts of online reviews believe that the reviews are directly responsible for their marketing strategies as well as purchasing decisions. Two surveys related to impact of opinion mining on individual, society and organizations were carried out in America on 2000, and on American people in 2008, by Horrigan [4]. The key findings of this surveys are stated below:

- 81% of Internet users make online investigation of the product at least once before purchasing it.
- 20% Internet users perform such investigation on a usual day.
- Consumers report shows that users are ready to pay starting from 20 to 99% additional for a 5-star-rated products than that of a 4-star-rated products.

Isinkaye, et al. have provided different characteristics and potentials of different prediction techniques in recommendation systems in order to serve as a compass for research and practice in the field of recommendation systems [5]. Collaborative Filtering technique filters out the recommendation with the help of user behavior in the form of ratings. This technique generates rating for an unrated item for the user, based on the

commonalities among users and their ratings. Recommendation quality is directly proportional to the size of rating dataset. This technique suffers from cold start problem for a new user and a new item. This is so because user with few ratings is difficult to categorize and item with few ratings cannot easily be recommended. Also users with unusual tastes will not get recommendation as per expectations because it's quite difficult to find their co-relation with other users to extract ratings. Content-based technique uses item features and user preference to provide recommendations. In this technique, item features like keywords are used to describe items, while user preference indicates the items liked by the user. It recommends items that are similar to the items preferred by the user. Since it recommends similar items based on user preference and item feature, crossed genre items preferred by the user cannot be recommended.

Knowledge-based methods mainly utilize the path between concepts in knowledge resources to indicate their semantic similarity [14, 15]. Lin [16] and Resnik [17] measured the semantic similarity by the information content ratio of the least common subsumer of the two concepts. Instead of directly exploiting the graph distance in knowledge resources, several researches [18,19] manufactured concept vectors according to a set of ontology properties for concept semantic similarity. Corpus-based methods assume that words with similar meaning often occur in similar contexts, Latent Semantic Analysis (LSA) [20] represents words as compact vectors via singular value decomposition (SVD) on the corpus matrix, and GloVe [21] reduced computational costs by training directly on the non-zero elements in corpus matrix. Turney[22] measured semantic similarity with the point-wise mutual information. Web can also be regarded as a large corpus, and the Google Distance [23] utilizes the number of words co-occur in web pages for concept semantic similarity.

Mikolov[24] proposed the word embedding approach with neural network to capture the word semantics occurred in a fixed size window. A challenge for determining semantic similarity of short texts is how to go from word-level semantics to short-text-level semantics.

A direct approach is utilizing weighted sum of word-to-word semantic similarity, and [4,5] used the greedy word alignment method to form the sentence semantic similarity.

Banea .et al. [6] measured the semantic similarity between text snippets infused with opinion knowledge. D Ramage et al. [8] constructed a concept graph using words of each text snippet, then measured the similarity between two concept graphs.

Recently, the Sem Eval conference released the Semantic Text Similarity (STS) task especially for the boom of short text semantics, and regression models are adopted by most teams [7,9] to predict the similarity score, and lexical feature and syntactic features are exploited.

The paragraph vector [25], which is similar to word embedding, is also proposed with neural network to measure the semantic similarity between short texts.

The research directly related to document semantic similarity is rare and undeveloped. Traditional similarity researches such as the TF-IDF [29] converted documents to vectors via word counting, and measured the similarity between documents by the similarity of vectors.

Academic articles can be regarded as semi-structured texts, which contain many structured annotations besides plain text. The similarity between academic articles can be measured with the help of annotated information. Martin et al. [26] fused the structured information, such as authors and keywords with traditional text-based measures for academic articles similarity. [10-12] regard articles and their citations as an information network.

Thus LDA can be used in the semantic analysis of long documents. Muhammad Rafi [13] defined a similarity measure based on topic maps in the document clustering task. The documents are transformed into topic maps based coded knowledge, and the similarity between a pair of documents is represented as a correlation between their common patterns. M. Zhang et al. [28] enriched document with hidden topics from external corpora, and measured the document similarity in text classification task with the similarity of topic distributions.

Hybrid technique combines two or more recommendation techniques to predict recommendation.

4. DATA SOURCES AND PRE-PROCESSING

Data is the fuel of any machine learning algorithm. To bring the model to life, a meaningful dataset is needed for the algorithms to work on.

This project contains two major parts: recommender system and sentiment analysis. The nature of data to be fed into the network is different for these two. For this reason, data collection and preprocessing has been explained separately.

4.1. Book recommender system

Although, the project plans to develop a system where uses can give rating and add review themselves, for the initial training of the models, proper label dataset is necessary. Unlike for movies, proper dataset is not abundantly for book. After some research, few good data sources were found. This project has merged following two publicly available datasets.

4.1.1. Book Crossing Dataset

This collection consists of 278,858 users providing 1,149780 ratings on 271,379 books. Duplicates entries and missing values were removed from the dataset. The distribution of the data possessed long tail distribution of the number of ratings per book and ratings per user. The books with less than 15 ratings and the user that has provided less than 15 ratings were removed.

4.1.2. UCSD Book Graph Dataset

It is a collection of 2,360655 books and 829,529 authors. This dataset was preferred because the books included in this dataset contains rich variety of genres.

As shown in Figure 4.1, above two datasets were merged based on the ISBN number. The main component of first dataset was user item interaction data and main components of later one was the genres details and descriptions which will further be used in content based filtering. After merging the dataset, interaction matrix was prepared from interaction dataset by using encoding tools (from Scikit-learn library). Thus, obtained interaction matrix is fed into the network.

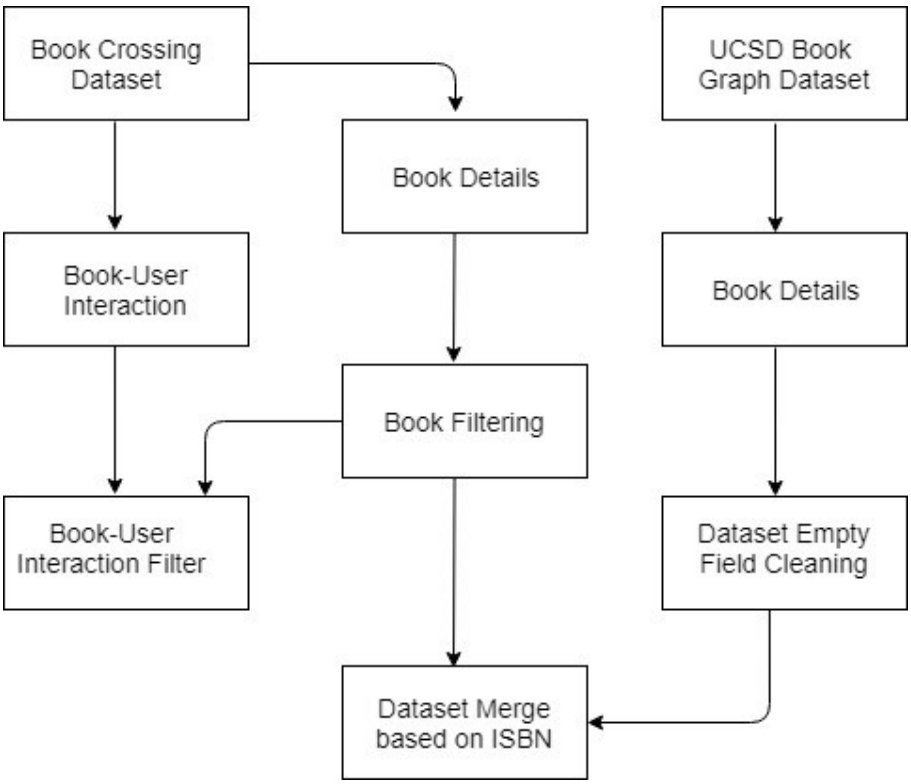

Figure 4.1: Merging of Book Crossing and UCSD Book Graph Dataset

Following section visualizes the basic structure of the dataset obtained after merging.

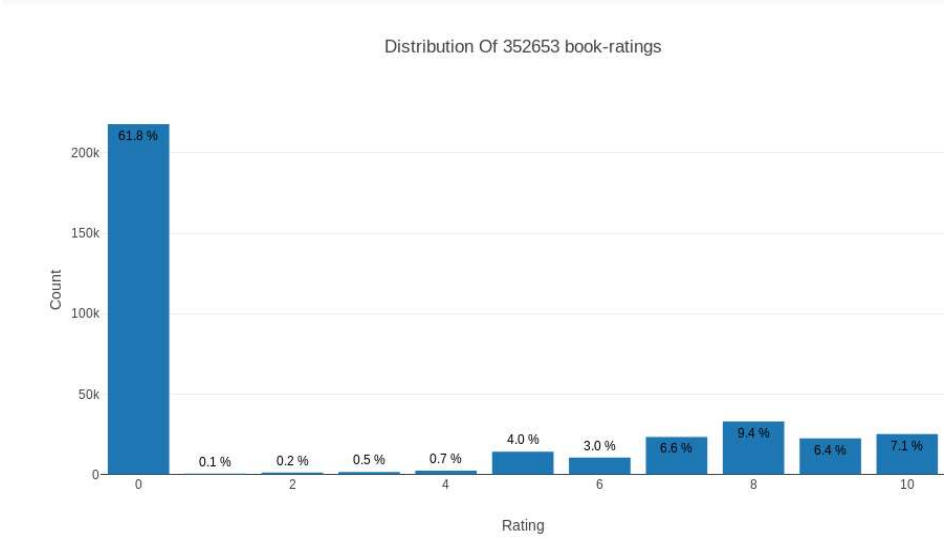

Figure 4.2: Distribution of book with rating

Figure 4.2 shows the number of books at a specific rating value. The dataset originally, contains rating values from zero to ten with zero rating signifying that the book has not been rated. It concludes that the dataset contains large number of book which have not been rated.

Figure 4.3 represents the distribution of rating per user. It can be seen that the dataset contains abundant users who have not rated the book is high. As the rating value goes increasing, the count goes on decreasing.

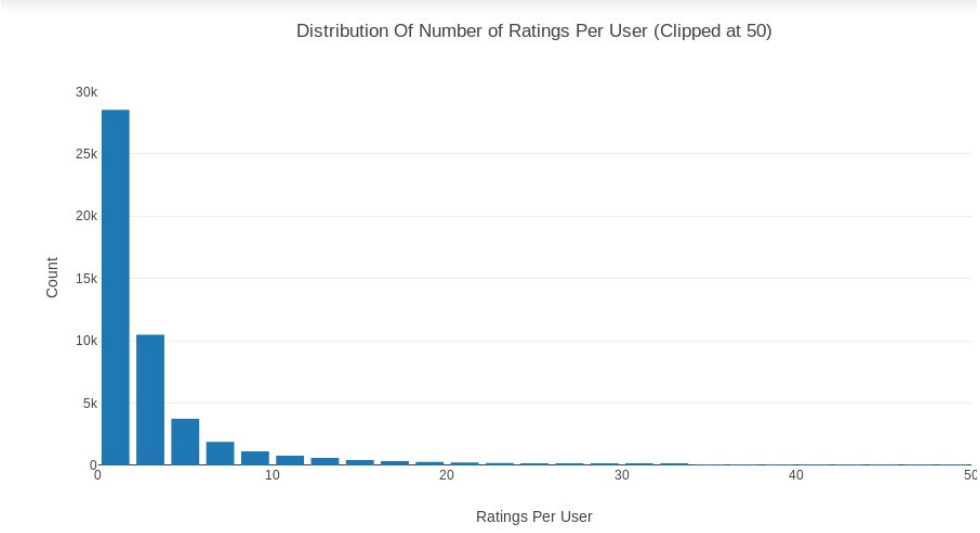

Figure 4.3: Ratings of book per user

4.2. Sentiment Analysis

For making sentiment analyzer, the deep learning framework PyTorch was used which provides access to Large Review Dataset for educational purpose. Along with the review dataset, GloVe, open-sourced by Stanford, was used for obtaining vector representations for words. Word vectors are achieved by mapping words into meaningful space where the distance between words is related to semantic similarity. Training is performed on aggregated global word-to-word co-occurrence statistics from a corpus, and the resulting representations showcase linear substructures of the word vector space.

Following steps were carried out before the data was fed to the network.

4.2.1. Data Split

The above dataset provided by PyTorch comes pre-split into train and test sets. The splitting of dataset is necessary because testing the performance of a model on the same dataset that is used to train the model doesn't give the true indication of how good the model is performing.

4.2.2. Data preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an format that can be understood. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing. This is the basic after the collection of the data. The user may not get the desired datasets for his need, so the user needs to modify the datasets according the need and demand of the project. During this course, the user may need to handle the missing values, outliers or any other misrepresentation of the datasets.

The text review contains a large number of words including regular expression, punctuation, commas, full stops and other forms of the words like past or future form,

which are not needed and they are redundant to our project. For this NLTK, a python library for Natural Language Processing has been used.

4.2.3. Data cleaning

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Since the data provided by PyTorch were already built, there were no any faults to be cleaned.

4.2.4. Data reduction

Data reduction is the transformation of numerical or alphabetical digital information derived empirically or experimentally into a corrected, ordered, and simplified form. The basic concept is the reduction of multitudinous amounts of data down to the meaningful parts. On the other hand, this step is especially used for testing the operation and efficiency of machine learning algorithms. Sentiment analyzer and recommender model were first tested in reduced datasets to make the computation process faster.

4.2.5. Normalization

Data normalization is the process of converting the data to any kind of canonical forms. In terms of the machine learning or signal processing, data normalization refers to transforming the values to al limited range or scale. For example, if a review data provides sentiment labels as rating scales from 1 to 10, it is required to normalize that to convert the labels to the required value that is 1 to 5. In this case, a simple division by 2 seems to be sufficient but on other cases, it might take more such operations such as subtracting mean value and then dividing by the standard deviation.

5. RECOMMENDER SYSTEM

Recommender systems are a subclass of information filtering system that seek to predict the rating that a user would give to an item. Recommender systems deal with the problem of information overload by filtering vital information fragment out of large amount of dynamically generated information according to user's preferences, interest or observed behavior about item. Recommender systems have become extremely common in recent years, and are utilized in variety of areas: some popular applications include movies, music, news, books, research articles, search queries, social tags and products in general. There are also recommender system for experts, restaurants, jokes, garments, financial services and twitter pages.

Recommender system typically produce a list of recommendations in one of two ways – through collaborative and content-based filtering or the personality-based approach. Collaborative filtering approaches build a model form a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items that the user may have an interest in. Content-based filtering approaches utilize a series of

discrete characteristics of an item in order to recommend additional items with similar properties. These approaches are often combined to form hybrid recommender system.

5.1.1 Phases of Recommendation Process

The recommendation process generally consists of the following phases, as shown in Figure 5.1.

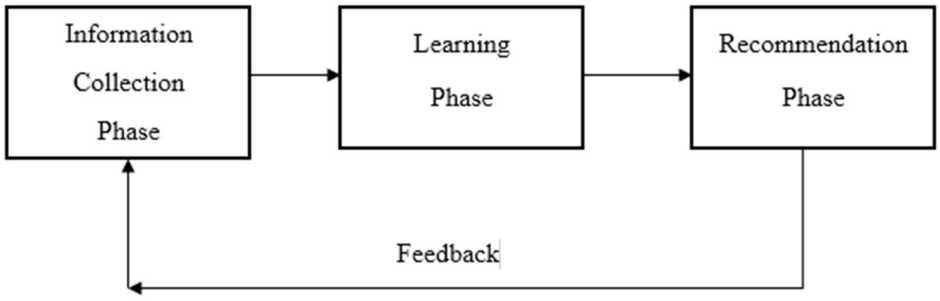

Figure 5.1: Recommendation phases

5.1.2 Information Collection Phase

This is the process of collecting relevant information of users to generate a user profile or model for prediction tasks including user's attribute, behaviors or content of the resources the user accesses. Recommender systems rely on different types of input such as the most convenient high quality explicit feedback, which includes explicit input by users regarding their interest in item or implicit feedback by inferring user preferences indirectly through observing user behavior. Hybrid feedback can also be obtained through the combination of both explicit and implicit feedback.

1. Explicit Feedback: The system normally prompts the user through the system interface to provide rating for items in order to construct and improve the model. The accuracy of recommendation system depends on the quantity of ratings provided by the user. The only shortcoming of this method is that it requires effort from the users and also the users are not always ready to supply enough information.
2. Implicit Feedback: The system automatically infers the user's preferences by monitoring the different actions of users such as the history of purchases,

navigation history, and time spent on some web pages, links followed by the user, content of e-mail and button clicks among others.

3. Hybrid Feedback: The strengths of both implicit and explicit feedback can be combined in a hybrid system in order to minimize their weaknesses and get a best performing system.

5.1.3 Learning Phase

It applies a learning algorithm to filter and exploit the user's features from the feedback gathered in information collection phase.

5.1.4 Recommendation Phase

It recommends or predicts what kind of items the user may prefer. This can be made either directly based on the dataset collected in information collection phase which could be memory based or model based or through the system's observed activities of the user.

Figure 5.2 depicts different types of recommender system in practice.

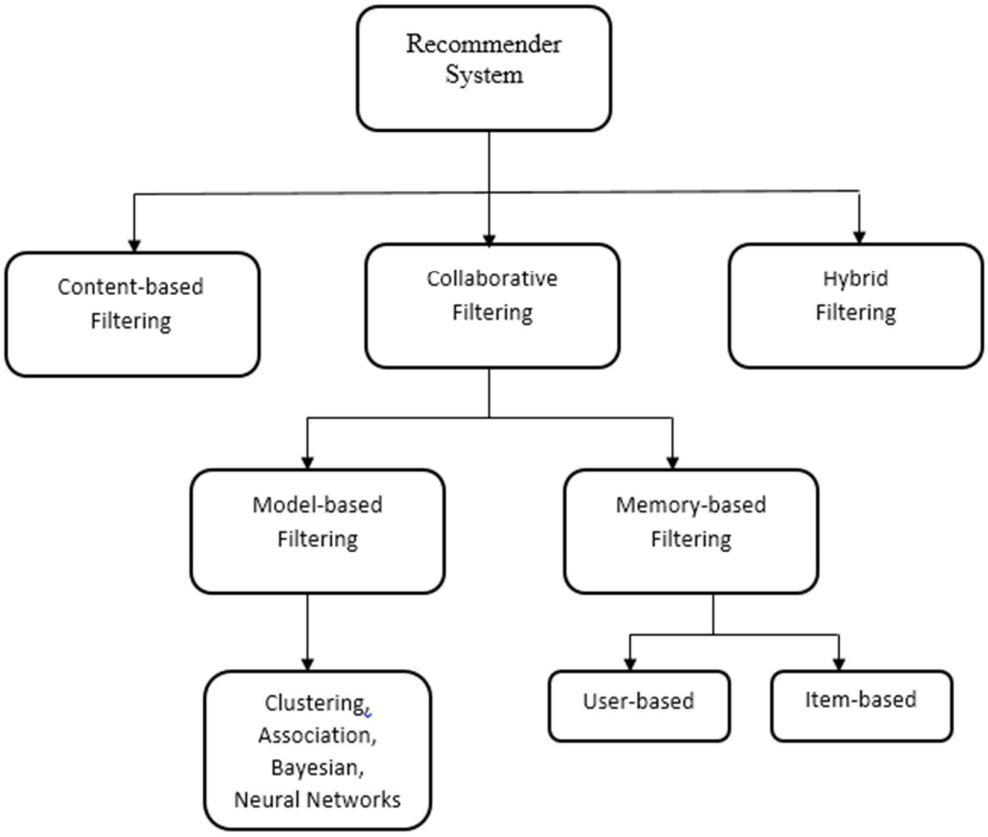

Figure 5.2: Classification of recommender system

5.2 Similarity Measures

Recommender systems recommend us items that are similar to the ones that the user has already liked or that are liked by other users with similar interest. In any case, the notion of similarity is required between users or items.

There are various similarity or distance measures for various types of objects. In this project, the main concern is with the objects that are vectors or real numbers. For such projects, following popular measures are available.

5.2.1 Euclidean Distance

If p and q are two vectors of dimension n, then the Euclidean distance between them is defined as:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

5.2.2 Cosine Similarity

Given two vectors of attributes a and b, the cosine similarity cosφ is represented using dot product and magnitude as:

$$\cos\theta = \frac{\vec{a}.\vec{b}}{|\vec{a}|\,|\vec{b}|}$$

where, $\vec{a}.\vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of two vectors.

There are several other similarity measures available such as Pearson correlation coefficient but not used in this project.

5.3 Content-based Filtering

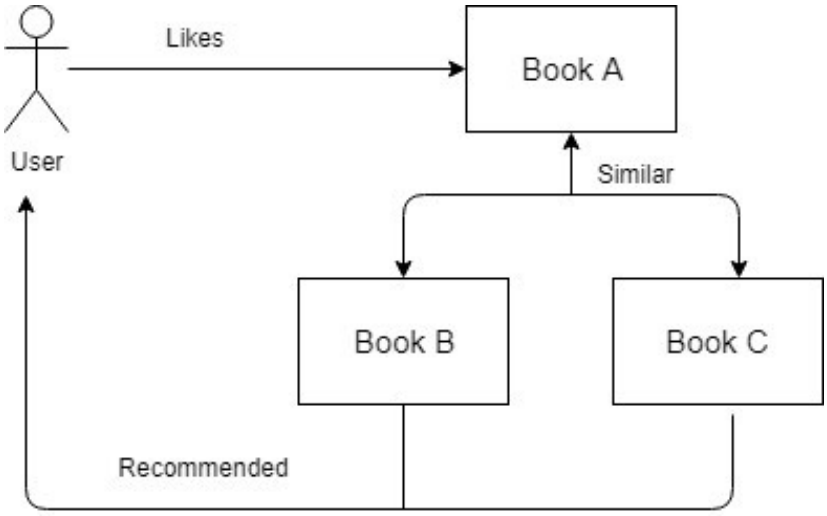

Figure 5.3: Content-based Filtering

Content-based filtering recommends items based on a comparison between the content of items and a user profile. Content-based technique is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items in order to generate predictions.

CBF technique still has the potential to adjust its recommendations within a very short period of time even if the user profile changes. The major disadvantage of this technique is that it needs in-depth knowledge and description of the features of the items in the profile.

For example, consider the features of a book that might be relevant to a recommender system:

1. The genre of book. Some reader may like fiction while other may like science fiction only.

2. The year in which the book was published. Some reader prefer ancient writing whereas some may prefer latest books.

3. The author who wrote the book.

5.3.1 Pros and Cons

CBF techniques overcome the challenges of CF. They have the ability to recommend new items even if there are no ratings provided by users. So even if the database doesn't contain user preferences, recommendation accuracy is not affected.

However, CBF are dependent on item's metadata. They require rich description of items and very well organized user profile before recommendation can be made to users.

5.4 Collaborative Filtering

Collaborative filtering is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. A person who wants to see a movie for example, might ask for recommendations from friends. CF is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many users. The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue x than to have the opinion on x of a person chosen randomly.

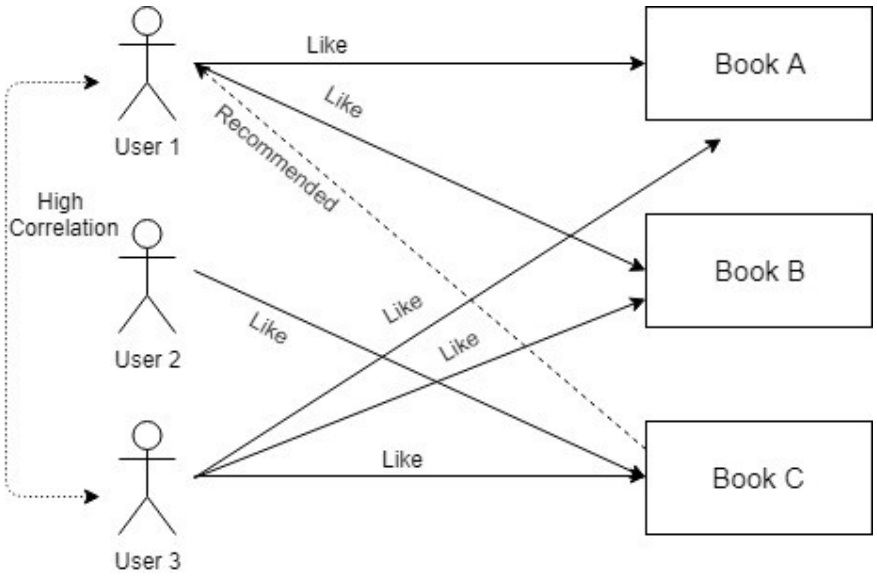

Figure 5.4: Collaborative Filtering

Most common CF systems can be reduced to two steps:

1. Look for user who share the same rating patterns with the active user (the user whom the prediction is for).

2. Use the ratings from those like-minded user found in step 1 to calculate a prediction for the active user.

Collaborative filtering works by building a database (user-item matrix) of preferences for items by users. It then matches the users with the relevant interest and preferences by calculating similarities between their profiles to make recommendations. Such users build a group called neighborhood. A user get recommendations to those items that he/she has not rated before but that were already positively rated by users in his/her neighborhood. The output produced by CF can be either predictions or list of recommendations. Prediction is a numerical value P_{ij}, expressing the predicted score of item j for the user I, while recommendation is a list of top-N items that the user will like the most as shown in Figure 5.4.

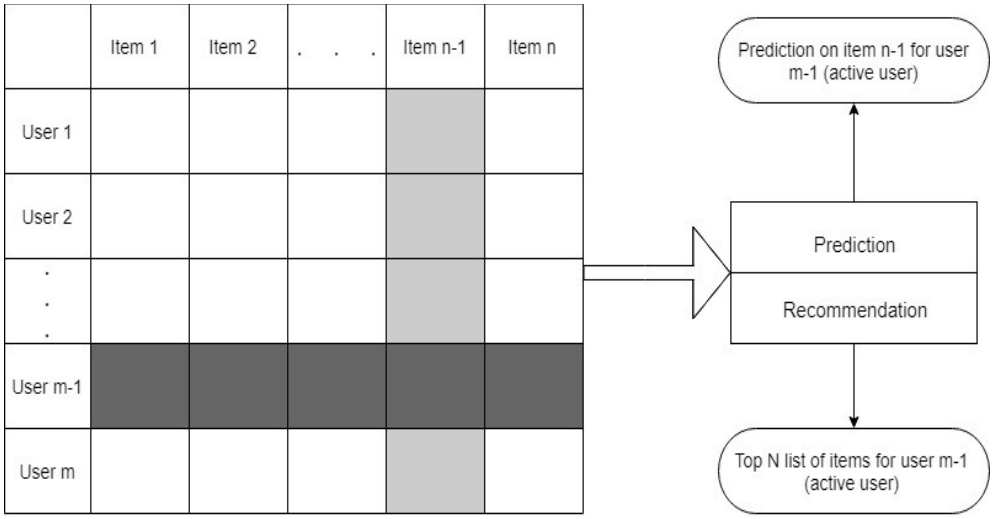

Figure 5.5: Collaborative filtering process

5.5 Hybrid Filtering

Hybrid filtering technique combines different recommendation techniques in order to gain better system optimization to avoid some limitations and problems of individual recommendation algorithms. This approach is usually implemented in order to avoid the cold start problem whereby the model can't recommend items (books) to the user because adequate information about the user has been collected. A hybrid approach combines the two types of information while it is also possible to use the recommendations of the two filtering techniques independently.

In this project, the recommendations from two filtering techniques have been combined separately. The system recommends books similar to a single book or a number of books the user has interacted with based on the similarities among the book metadata as long as the rating count of the user doesn't exceed certain mark (15 books). When the user has interacted with more than 15 books, his/her activity becomes sufficient for the model to give recommendation based on collaborative filtering.

5.6 Evaluation of recommender system

The quality of recommender system can be evaluated in two types of scenarios: offline experiments and online experiments. The evaluation metric may be different for each different use case.

5.6.1 Offline evaluation

The idea of offline evaluation is to use data that is regarded as truthful. Then split the data into two parts and feed one part to the recommender. Use the other part to verify that the recommender predicts ratings on items in the set that were hidden to it; those that are close to the actual ratings or those that produce recommendations that contain items that were highly rated in hidden data,

Offline evaluation covers different statistical accuracy metrics that evaluates accuracy of a filtering technique by comparing the predicted directly with the actual user rating. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are commonly used evaluation metrics. Various studies have found that RMSE doesn't describe average alone and has other implications that are more difficult to tease out and understand. MAE has been used as evaluation metric in this project.

Root Mean Square Error (RMSE) is computed as:

$$RMSE = \sqrt{\frac{\sum_{u,j=1}^{N}\left(p_{u,j} - r_{u,j}\right)^2}{n}}$$

RMSE puts more emphasis on larger absolute error and lower the RMSE is, the better the recommendation accuracy.

MAE is the most popular and commonly used. It is the measure of deviation of recommendation from user's specific value. It is computed as follows:

$$MAE = \frac{\sum_{u,i} p_{ui} - r_{ui}}{n}$$

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation. The MAE is a linear score which means that all the individual differences are weighted equally in the average.

The lower the MAE, the more accurately, the recommendation system predicts user ratings.

5.6.2. Online evaluation

When the offline evaluation results of the recommender algorithm are satisfactory, it can be deployed and tested out on real humans. A way to do something similar is to provide access to a small group of people, which is normally termed family and friends, who you trust to try out the system and return honest feedback. But in order to gain real feedback from real customers, one can count the numbers of clicks on the recommended item.

5.7 Algorithm

5.7.1 Low Rank Matrix Factorization

In this project, Low Rank Matrix Factorization is used to implement recommender algorithm. There are other methods to develop a recommender system but the most widely used approach for sparse dataset is based on matrix factorization. Matrix

factorization techniques are more effective because they discover the latent features underlying the interaction between users and items. As in case of this project, an item refers to a book, item and book will be used interchangeably from here.

Factorization is about splitting things up. For example, one can split a number like 100 into the following prime factorization.

100 = 2 x 2 x 5 x 5

The same idea can be applied to factorizing a matrix. Matrix factorization (MF) deals with finding out two (or more) matrices such that the original matrix can be restored from the smaller matrices. MF can be used to discover latent features underlying the interactions between two different kinds of entities. When the number of latent features is low, the process thus followed to factorize a matrix is called low rank matrix factorization.

In a recommendation system, the matrix (which is to be split into smaller ones) is the user-item interaction matrix. There is a group of users and a set of items (books). Given that users have rated some items in the system, the objective is to predict how the users would rate the items that they have not yet rated, such that the items carrying top N prediction (rating) can be recommended to each user.

A matrix R can be decomposed into following form.

$R = U V$

If R has n rows and m columns (as in n users and m items), you call it an n × m matrix; the size of U will be an n × d matrix and V is a d × m matrix.

The primary objective is to decompose the matrix R into hidden features (read columns for users and rows for items) for items and for users. In the field of recommender systems, U is referred as the user-feature matrix and V as the item-feature matrix.

To perform the factorization, the values in the U and V matrices should be inserted in such a way that UV is close to R as possible.

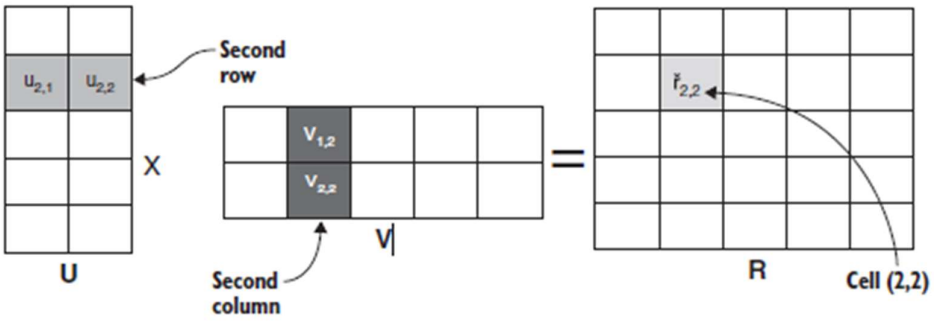

Figure 5.6: Matrix factorization

As in the Figure 5.6, the cell is calculated using the dot product. The dot product is given by:

$$\hat{r}_{2,2} = u_2 \; x \; v_2 = u_{2,1}v_{1,2} + u_{2,2} + v_{2,2}$$

Let,

U be the set of users

D be the set of items (books)

R = |U| x |D| be the matrix that contains all the ratings that the users have assigned to the items.

K be the latent features

The task is to find two matrices P (of size |U| x |K|) and Q (of size |D| x |K|) such that their product approximates |R|.

$$R \approx P \; X \; Q^T = \hat{R}$$

In this way each row of P would represent strength of the associations between a user and the features. Similarly, each row of Q would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item d_j by u_i, we can calculate the dot product of their vectors.

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

The next step is to find a way to obtain P and Q. One way to approach this problem is to first initialize two matrices with some values, calculate how different their product is to M, and then try minimize the difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference.

The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

$$e_{ij}^2 = \left(r_{ij} - \hat{r}_{ij}\right)^2 = \left(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj}\right)^2$$

The squared is considered because the estimated rating can be either higher or lower than the real rating.

To minimize the error, we have to know in which direction we have to modify the values of p_{ik} and q_{kj}. In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately:

$$\acute{p}_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$

$$\acute{q}_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

Here α is a constant termed as learning rate whose value determines the rate of the minimum. The value of α is in range of 0.001 to 0.00001.

The task is not to come up with P and Q such that R can be reproduced exactly. Instead, the aim is to only to try to minimize the errors of the observed user-item pairs. If T be a set of tuples, each of which is in the form of (u_i, d_j, r_{ij}), such that T contains all the observed user-item pairs together with the associated ratings, the aim is only to try to minimize every e_{ij} for (u_i, d_j, r_{ij}) ϵ T (where T is the training data.) As for the rest of the unknowns, it will be able to determine the values once the associations between the users, items and features have been learnt.

Using above update rules, the operation can be iteratively performed until the error converges to its minimum. The overall error can be calculated using the following equation and determine when the process should be stopped.

$$E = \sum_{(u_i,d_j,r_{ij})\in T} e_{ij} = \sum_{(u_i,d_j,r_{ij})\in T} \left(r_{ij} - \sum_{k=1}^{K} p_{ik} Q_{kj} \right)^2$$

A common extension to this method is to introduce regularization to avoid overfitting. This is done by adding a parameter β and modify the squared error as follows:

$$e_{ij}^2 = \left(r_{ij} - \hat{r}_{ij}\right)^2 = \left(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj} \right)^2 + \frac{\beta}{2} \sum_{k=1}^{K} \left(||P||^2 + ||Q||^2 \right)^2$$

The new parameter β is used to control the magnitudes of the user-feature and item-feature vectors such that P and Q would give a good approximation of R without having to contain large numbers. In practice β is set some values in the order of 0.02. The new update rules for this squared error can be obtained by a procedure similar to the one described above. The new update rules are as follows:

$$\acute{p}_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2e_{ij} q_{kj} - \beta p_{ik})$$

$$\acute{q}_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2e_{ij} p_{ik} - -\beta q_{kj})$$

5.7.2 Paragraph Vector

Doc2vec (aka paragraph2vec, aka sentence embeddings) modifies the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. The goal of doc2vec is to create a numeric representation of a document, regardless of its length.

Doc2vec uses following algorithms:

5.7.2.1 Distributed Memory version of Paragraph Vector (PV-DM)

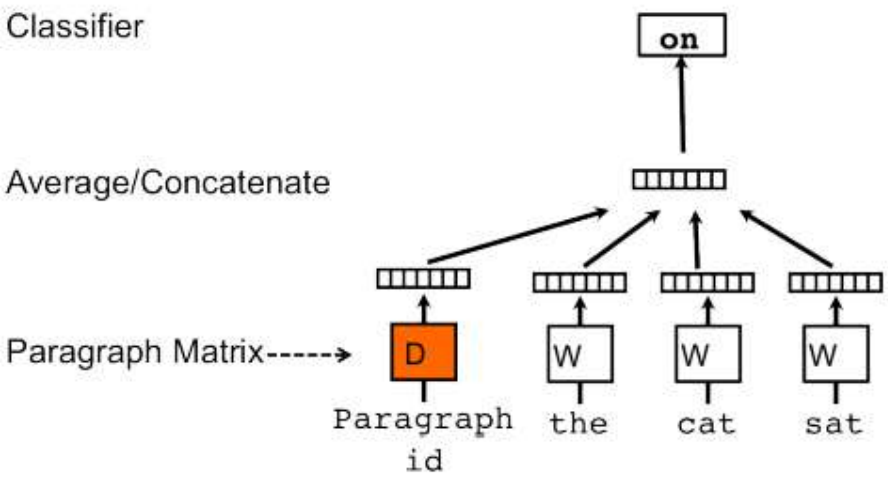

Figure 5.7: PV-DM model

Instead of using just words to predict the next word, another feature vector is added, which is document unique. So, when training the word vectors W, the document vector D is trained as well, and in the end of training, it holds a numeric representation of the document.

The model above is called Distributed Memory version of Paragraph Vector (PV-DM). It acts as a memory that remembers what is missing from the current context — or as the topic of the paragraph. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document.

5.7.2.2 Paragraph Vector Distributed Bag Of Words (PV-DBOW)

As in word2vec, another algorithm, which is similar to skip-gram may be used Distributed Bag of Words version of Paragraph Vector (PV-DBOW)

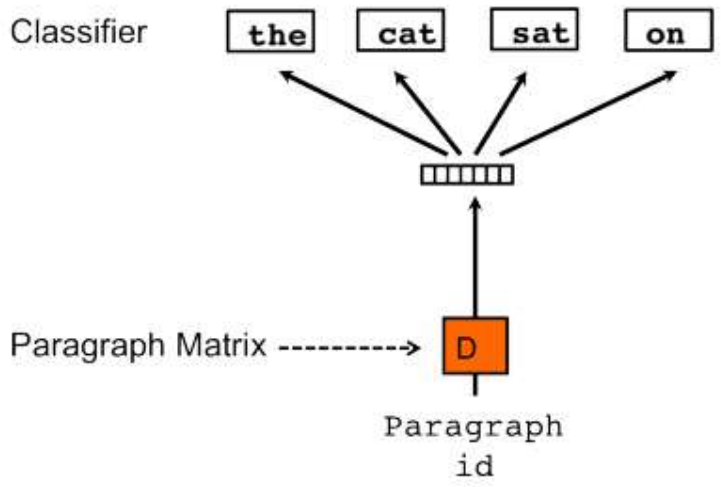

Figure 5.8: PV-DBOW model

Here, this algorithm is actually faster (as opposed to word2vec) and consumes less memory, since there is no need to save the word vectors.

The doc2vec models may be used in the following way: for training, a set of documents is required. A word vector W is generated for each word, and a document vector D is generated for each document. The model also trains weights for a softmax hidden layer. In the inference stage, a new document may be presented, and all weights are fixed to calculate the document vector.

6 Sentiment Analysis

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business understand the social sentiment of their brand, product or service while monitoring online conversations. Sentiment Analysis also known as Opinion Mining is a field within Natural Language Processing (NLP) that builds systems that try to identify and extract opinions within the text.

The sentiment analyzer can be built using different methods. Existing approaches to sentiment analyzer can be grouped into three main categories: knowledge-based techniques, statistical methods and deep learning approaches.

6.1. Knowledge-based approach

Knowledge-based techniques look for unambiguous affect words such as “good”, “bad”, etc. to determine the sentiment polarity of the text. This is probably the simplest way of implementing a sentiment analyzer but this is also the least accurate because reviews contains complex grammatical structures like negations, idioms, and sarcasms that cannot be all listed explicitly to develop a knowledge-based sentiment classifier. Therefore, knowledge-based approach only works for simple sentences.

6.2. Statistical Bag-of-words Methods

Statistical methods use elements of machine learning such as latent semantic analysis, Support Vector Machines (SVM), bag-of-words (BOW), etc. to calculate the probabilistic polarity of a text. This model is similar to the knowledge-based approach in a way that it also looks at words lexically in isolation, i.e. it analyzes the character composition of words and not the true meaning of the word. For example, lexically, the words “swam” is more similar to “swan” than “swimming”.

The only difference between this BOW method and knowledge-based approach is that in this method, the knowledge is learned from the label dataset whereas in the knowledge-based approach, the fact that word “good” carries positive sentiment and the word “bad” carries negative sentiment is explicitly fed to the algorithm. In the BOW method, the sentiment of the word “good” is learned from the statistics of its appearance in the labeled dataset. For example, if there are 100 positive examples and 100 negative examples in the dataset and 60 positive examples contain the word “good” whereas 20 negative examples contain the word “good”, then the model learns that the word “good” must be 75% positive because out of 80 training examples containing the word “good”, 60 are positive and 20 are negative, and therefore positive equals 0.75 (60/80).

If single word is taken into account at a time, it is called unigram BOW model. The order of the words in the sentence is ignored in such model and important information may be lost. If a pair of consecutive words is taken from the dataset and calculated jointly, then bigram BOW model is obtained. Similarly, if three consecutive words are taken, the trigram BOW model is obtained. By using these n-gram BOW models, the accuracy of the model can be boosted further, but it requires more computer memory and processing resources.

6.3. Deep Learning Approach

Sentiment analysis is a subclass of NLP. Specifically, it is a form of text classification problem where one classifies a piece of text into one of the sentiment classes. The modern approach for NLP is deep learning using Artificial Neural Networks (ANN). ANNs are machine learning models inspired from the structure of the biological brain. An ANN is based on collection of connected units called artificial neurons, analogous to axons in a biological brain. Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically 0 and 1. Neurons and synapses may also have weight that varies as learning proceeds, which can increase or decrease the signal strength as it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level is the downstream signal sent.

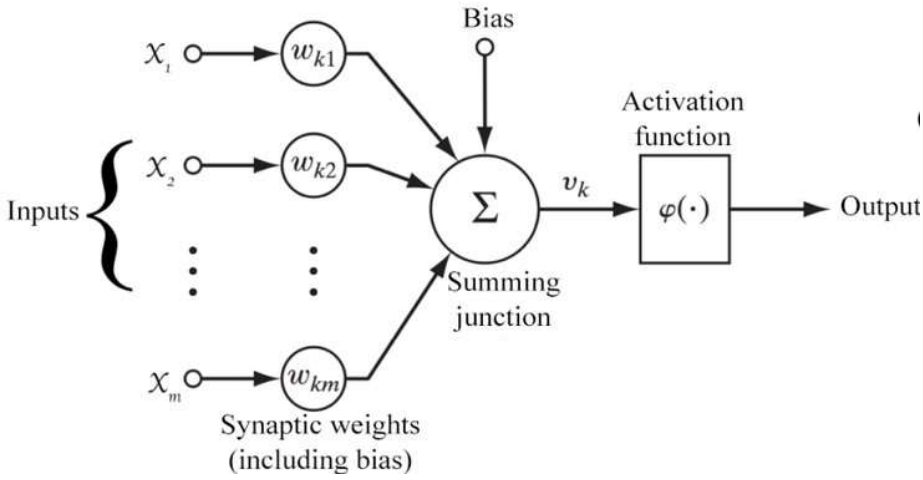

Figure 6.1: An artificial neuron

Let,

$n \in Z^+$ = number of inputs

$x \in R^n$ = vector of input parameters

$w \in R^n$ = vector of weights associated with the corresponding inputs

$b \in R$ = bias term

Then, the output is,

$y = \varphi(\sum_{i=1}^{n} w_i x_i + b)$

It can be written in vector form as,

$y = \varphi(w.x + b)$

Where, φ is a monotonically increasing, continuous, differentiable and bounded non-linear function such as:

$sigmoid(x) = \frac{1}{1+e^{-x}}$

$\tanh(x) = \frac{1-e^{-x}}{1+e^{-x}}$

$rectifier(x) = x^{+} = \max(0, x)$

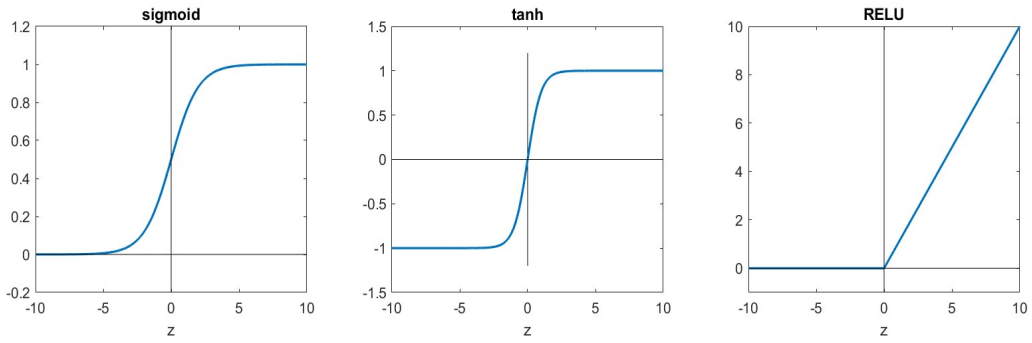

Figure 6.2: Plots of various nonlinear functions used as activation functions in ANN

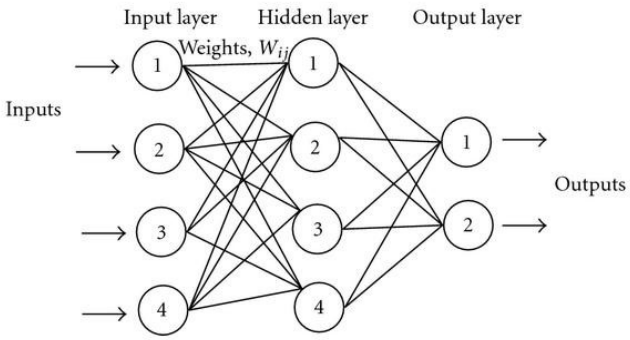

Figure 6.3: A simple ANN with 4 input nodes, 1 hidden layer with 4 nodes and 2 output nodes

Figure 6.3 show a simple neural network. All the arrows shown in the figure have weights associated with them. Those weights are adjusted during training phase, which is done using an algorithm called backpropagation. Backpropagation is an algorithm used to calculate the gradient of the cost function with respect to the weights in an ANN. It is commonly used as a part of algorithms that optimize the performance of the network by adjusting the weights. It is also called backward propagation of errors. The cost function is a measure of the error of the neural network model. Most common cost functions are squared error, cross-entropy error, etc. The neural network weights are adjusted to minimize the cost function using any of the optimization algorithms such as gradient descent, BFGS, etc. For instance, in gradient descent, the weight updating step is:

$$\theta_i = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i} \text{ for } i = 1, 2, 3, ..., n$$

Where,

n = number of parameters to learn

θ = parameter vector to learn

$J(\theta)$ = cost function to minimize

α = learning rate

The term “deep learning” basically refers to ANN models having many hidden layers. They could be simple ANNs with multiple hidden layers, or ANNs with special architectures such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and so on.

In deep learning models, there are lots of parameters to learn (adjust) and usually the training set is not sufficient to learn those parameters with great certainty. Therefore, deep learning models are prone to overfitting. Hence, regularizing the cost function is vital for good performance of the model.

6.3.1. Recurrent Neural Network (RNN)

Unlike feed forward networks, RNNs have a memory which captures information about what has been calculated to make use of sequential information. In order to predict next word in a sentence, it is essential to know the words came before it. RNNs perform same

task for every element of a sequence, with the output being depended on the previous computations.

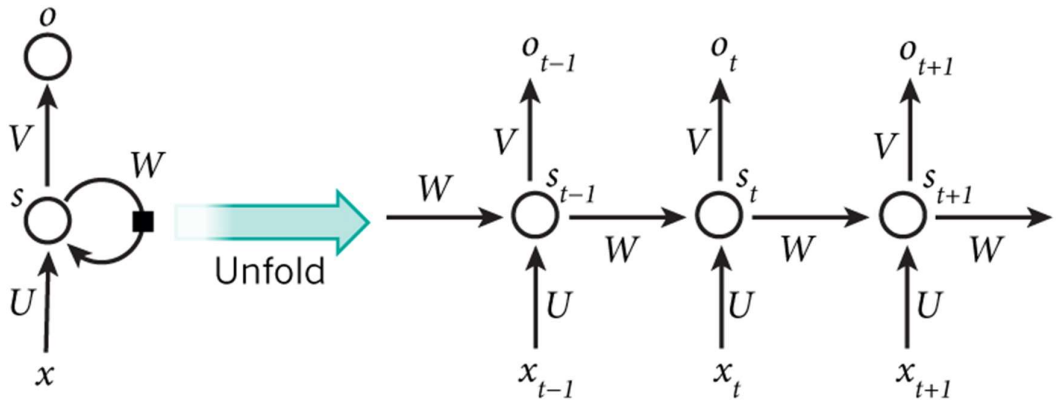

Figure 6.4: A recurrent neural network and the unfolding in time of the computation involved in its forward computation (source: skymind.ai)

Figure 6.4 shows a RNN being unrolled (or unfolded) into a full network. For example, if the sequence is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in a RNN are as follows:

- x_t is the input at time step t. For example, x_1 could be a one-hot vector corresponding to the second word of a sentence.

- s_t is the hidden state at time step t. It's the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$. The function f usually is a nonlinearity such as tanh or ReLU. s_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeroes.

- o_t is the output at step t. For example, if it is required to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. $o_t = \mathrm{softmax}(Vs_t)$.

6.3.2. Long Short Term Memory (LSTM)

Standard RNNs suffer from the vanishing gradient problem. As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train. This is termed as vanishing gradient problem because of which standard RNN doesn't perform well for long sentences. LSTMs overcome this by having an extra recurrent state called a cell 'c' - which can be thought of as the "memory" of the LSTM that uses multiple gates which control the flow of information into and out of the memory. LSTMs can be expressed as a function of x_t, h_t and c_t , instead of just x_t and $y_{t..}$

$$(h_t, c_t) = LSTM\ (x_t, h_t, c_t)$$

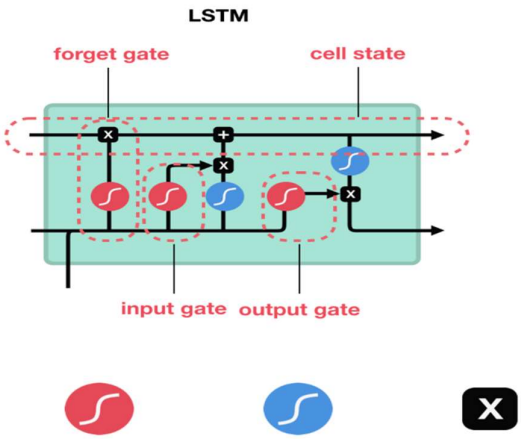

Figure 6.5: A LSTM cell (source: skymind.ai)

The initial cell state (c0) is initialized to a tensor of all zeros. The sentiment prediction is, only made using the final hidden state.

7 System Architecture and Implementation

7.1. System Design

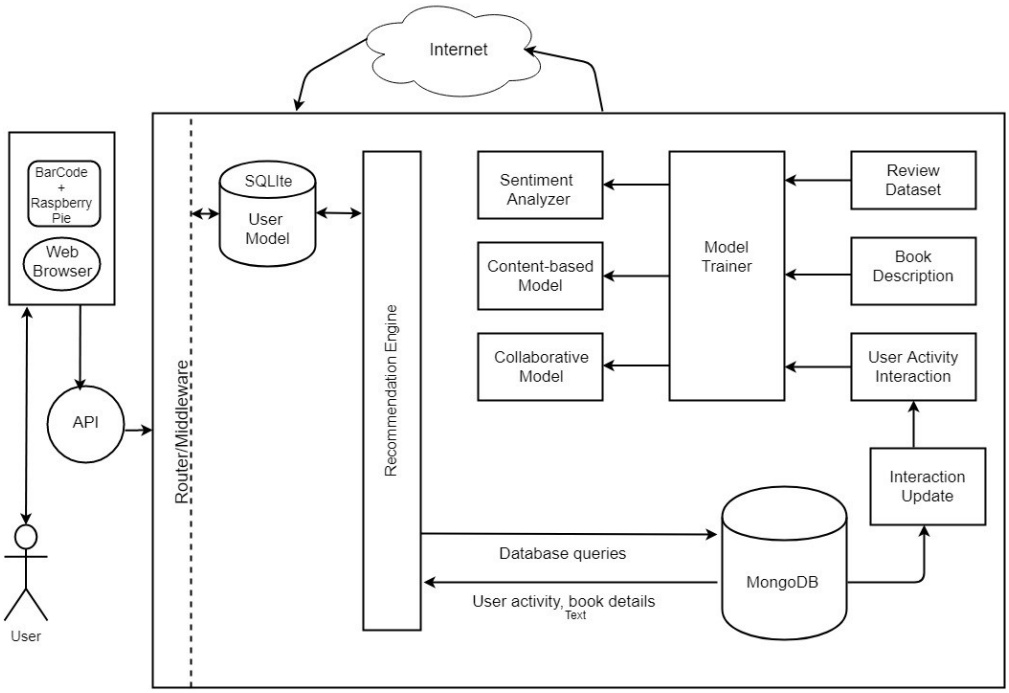

Figure 7.1: Overall system design of hybrid book recommender system

Distinct blocks presented in Figure 7.1 are explained below:

- Recommendation Engine: It is the collective form of multiple functions and actions that interact with and controls recommendation components and database operations. It make call to respective recommendation function that use respective model to generate respective recommendation. It make database queries related to book details, user activity, etc. It implements all the intermediate actions and function of the backend server.

- Sentiment Model: It is the model that is used to get ratings from review. It is trained using book review dataset and generates review based on sentiment polarity of the review given by the user.

- Content Based Model: It is the model that takes book description as input and generates similar books for recommendation as content based filtering. It is trained using book description corpus. It infer new description corpus and ranks similar books based on similarity measure of paragraph value.

- Collaborative model: It is used to generate recommendation based on similar user-item interactions. It is trained using Low Rank Matrix Factorization with gradient descent optimization. Neural network model is constructed using Pytorch library.

- Model trainer: It is the collection of functions that is used to train the dataset and generate the respective models. Content based filtering model is generated using Distributed Memory Model of Paragraph Vectors (PV-DM) on book descriptions dataset. Sentiment model was generated by training review dataset using RNN or CNN neural network model. Collaborative filtering model is trained using Low Rank Matrix Factorization algorithm.

- Book Review Dataset: It is used for training the sentiment analysis model.

- Book Description Dataset: It is the dataset containing book description as book's metadata collected from UCSD graph dataset.

- User Interaction File: It is file dynamically holds the user generated rating on the books. It is periodically updated using data from user activity collection of MongoDB database. This file is used to update collaborative filtering model.

- Interaction Update: It is used to update interaction matrix dynamically. It runs periodically and obtain data from MongoDB server. It also has other operations such as updating user activity collection, updating timestamp of its last operation etc.
- MongoDB database: It is combination of several collections that stores book metadata, user activity, barcode scanned data, etc.

7.3. Database Schema

MongoDB has been used as Database Management System (DBMS) to store the data in this project. Since this NoSQL database is comprised of nested structures, Entity Relationship Diagram (ERD) doesn't make whole lot of sense as in the case of relation databases. In the following section, database schema has been explained along the detail attribute list. In addition, the fields affected by CRUD operation under different conditions has been displayed in the respective tables.

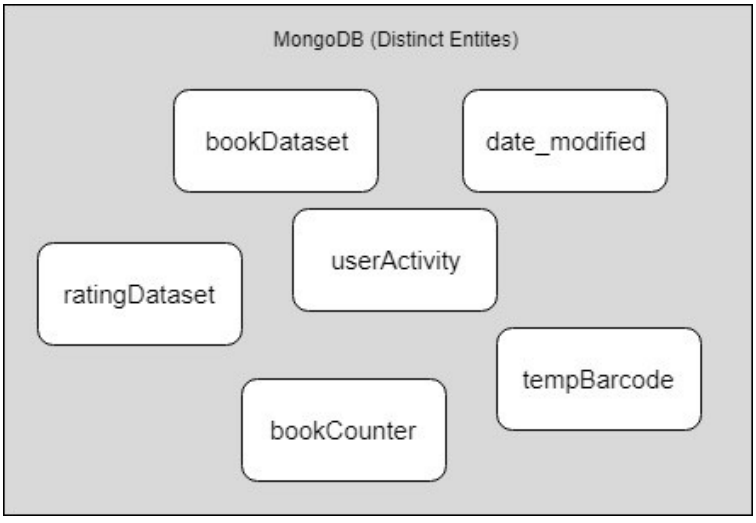

Figure 7.7: Distinct Entities used in MongoDB

1. **bookDataset :** It has all the data about book.
2. **ratingDataset**: It holds the rating the user provided to the given book
3. **bookCounter:** It is for increment of book id when one is inserted in the database
4. **tempBarcode:** It is the collection that holds the barcode of the book that was scanned by the user.
5. **userActivity:** This collection stores the activity of the user which are usually very dynamic
6. **date_modified:** It tracks the time of interaction matrix update

7.4. Model Update Procedure

The model is capable of providing book recommendation once it has been trained but in order to make it valid for new users and books, it should be updated regularly. The model update procedure incorporates update of two major entities in the database: user activity update and interaction matrix.

7.4.1. User Activity Update

The click on particular book, rating and review are termed as user activity here. As the site is kept in use, such user activity increases and these should be updated in the database properly.

Figure 7.8 demonstrates the step by step procedure on how the user activity gets updated.

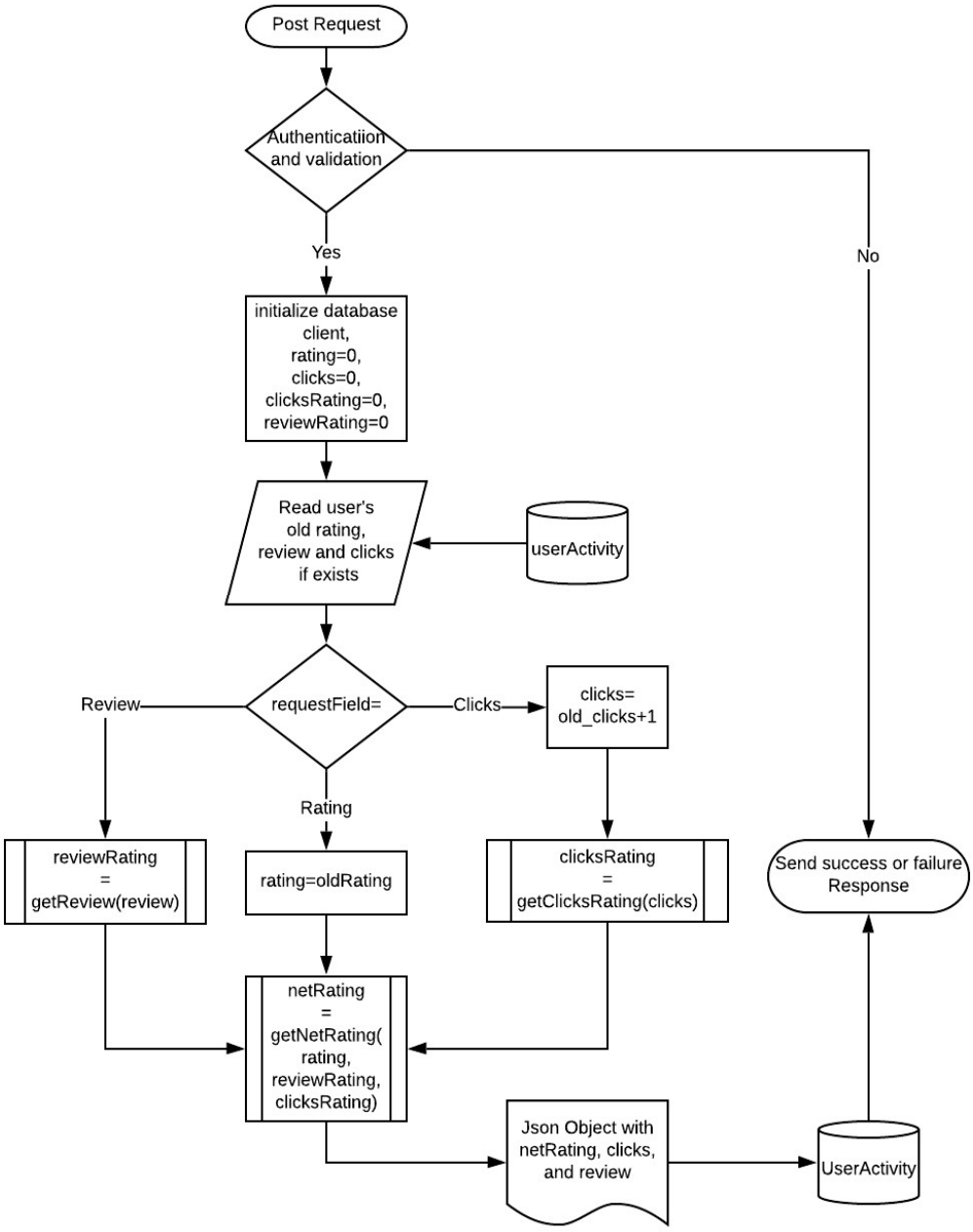

Figure 7.8: Flow chart showing user activity update operation

The post request is parsed which consists of either of 3 items i.e. clicks, rating and review. MongoDB client, database and collection object are initialized. Clicks are obtained as user clicks the books in the app. Rating is sent whenever user gives rating to the book. Review is sent whenever the user writes review to the book.

The request fields are parsed at the backend and its respective if statement is executed based on type of interaction. Different parameters such as ratings, clicks, clicks rating, review rating and net ratings are initialized to zero. If user and book activity already exists, then his previous activity is queried. These initialized parameter are replaced with the data from post request and result from the query. Different actions takes place in the collection like creating the new user object if user does not exist, creation of book activity object if does not exists already, etc.

Clicks of database is incremented as new clicks. Clicks rating is calculated by using formula as $5*(1-0.5^n)$ where n is the value of total no of clicks. This makes rating value increment with clicks from rating range 2.5 to maximum of 5.Review rating is taken by passing a string statement to the function which evaluates the polarity of review by using sentiment analysis in the range from 0 to 5. Net rating is calculated by using the function that takes available values from clicks rating, rating or review rating and taking its weightage average.

Net rating is calculated as:

$$ratings_{net} = \frac{\text{clicksRating} * \text{W}_{\text{cl}} + \text{reviewRating} * \text{W}_{\text{cl}} + \text{Rating} * \text{W}_{\text{ra}}}{\text{W}_{\text{cl}} + \text{W}_{\text{cl}} + \text{W}_{\text{ra}}}$$

Where,

W_{cl} = Weightage of click=0.25

W_{ra} = Weightage of rating=0.75

W_{re} =Weightage of review=1

Weightage of clicks is taken lower as click is comparatively of lower significant than that of rating and review. In case of absence of book rating and review, click can be dominant. Review rating is considered most dominating factor with weightage of 1. Star rating with weightage of 0.75 is more dominating than clicks but less dominating than review. The value zero is passed as argument in case of unavailability of rating. Unavailable value does not impact the rating. Rating is obtained as weightage average of passed arguments only. All the received and calculated data are updated in the "userActivity" collection. The "date_modified" field is also updated with current timestamp value. Then response to the client request is sent about success or failure of the operation.

7.4.2. Interaction matrix update

As the model predicts the rating which were not present in the interaction matrix, it becomes essential to update the matrix regularly to incorporate new users and books. Figure 7.9 shows this operation in a sequential way.

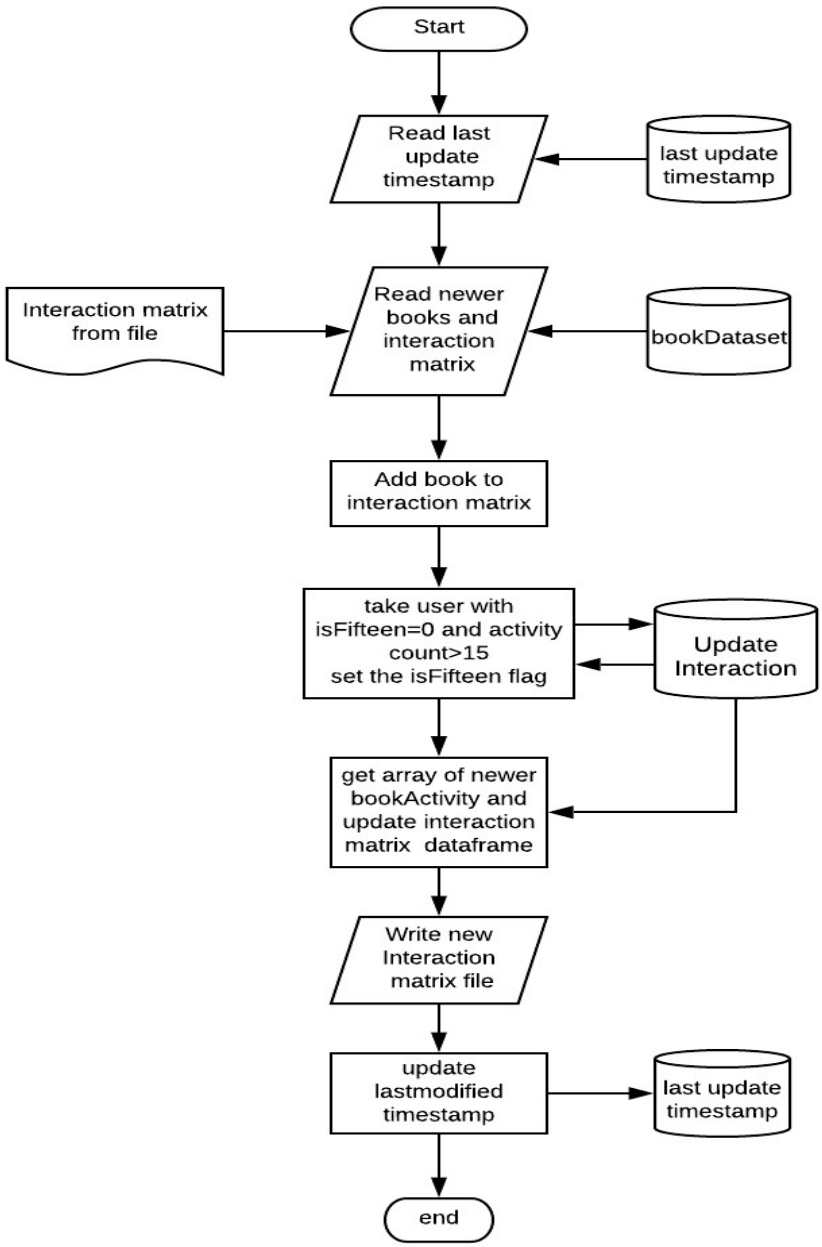

Figure 7.9: Flowchart showing interaction matrix update operations

Interaction matrix update occurs at regular interval as a batch operation. For better result of trained model, the minimum requirement conditions of 15 books interaction for given user is considered. For this operation, previous timestamp is read from “date_mofidied” collection. Newer books are fetched from “bookDataset” collection and is inserted into matrix with all user’s rating initialized to zero.

Interaction matrix update operations for a user is based on his “isFifteen” flag and “date_modified” field of “userActivity” collection.”isFifteen” flag indicates whether user has interacted in atleast 15 books. “date_modified” field indicates the timestamp of previous update made in the activity collection. “date_modified” field of “userActivity” is updated with its timestamp whenever user make interaction.“isFifteen” flag in the “userActivity” collection is set to 1 only if user have interaction with at least 15 books. Database client is initialized with “userActivity” collection and with update operation, “isFifteen” flag is set to 1 for all of these user.

The user activity is taken for matrix update operation only if isFifteen flag of user is set. We use aggregation pipelining of MongoDB with stages like addFields, size, match ,project ,unwind etc. to get all the data in desired format at single line query. It is an array containing object with “userId”, ISBN and net rating. After parsing the data, interaction matrix is updated by setting rating for given user id and ISBN no in loop. Then output is written back to the server which further is used for updating model in batch operation. Record of timestamp of this update operations is made. So the timestamp of batch update is updated with current timestamp and stored in “date_modified” collection

7.4.3 Implementation of similarity based recommendation

Doc2Vec model was instiantiated with a vector size with 50 words and iterating over the training sample of our books description dataset corpus 40 times. We set the minimum word count to 2 in order to discard words with very few occurrences. We then inferred a vector for any piece of text without having to re-train the model by passing a list of words to the model.infer_vector function. This vector can then be compared with other vectors via cosine similarity. The infer_vector() does not take a string, but rather a list of string tokens, which should have already been tokenized the same way as the words property of original training document objects.

The underlying training/inference algorithms are an iterative approximation problem that makes use of internal randomization, repeated inferences of the same text will return slightly different vectors.To assess our new model, we'll first infer new vectors for each

document of the training corpus, compare the inferred vectors with the training corpus, and then returning the rank of the document based on self-similarity.We saved the model and inferred vector array in a pickle file which serialize the variable and saved in disk. So in our app, we load the models and inferred vectors. So to obtain similarity, we take cosine similarity and rank the document which is sent as recommendation.

8 Experiments and Results

Book recommender system and sentiment analysis are two separate models implemented in this project. The following section examines these two models separately.

8.1. Book Recommender Model

During the research phase, different approaches for recommender system were examined. Most popular methods including nearest neighbor and latent factors were studied. Nearest neighbors methods are centered on computing the relationships between items but since, the dataset was sparse and this method becomes costlier and less effective as the dataset grow sparser, the low rank matrix factorization with batch gradient descent was chosen for recommending books out of sparse user-item rating matrix.

The entire dataset was divided into train, validation and test set with 80% of the whole dataset separated for training and remaining 20% for test set. To validate the model, 15% was randomly selected as validation set each time the sample get fed to the network. A number of steps were considered starting from a simpler dot product model to adding concatenation layer followed by dense and to adding the metadata of books for implementing the hybrid recommender model. In each step along the process, Mean Absolute Error (MAE) was taken as a measure of evaluation so as to examine how these model were performing.

As the best practice to tackle a deep learning problem is to start out with the simplest solution and then, introducing non-linearity to achieve higher performance, the dot product model whereby the smaller matrices obtaining after decomposing the user-item matrix were subjected to dot product to estimate the missing rating was implemented. Figure 8.1 shows the model summary in the abstract form.

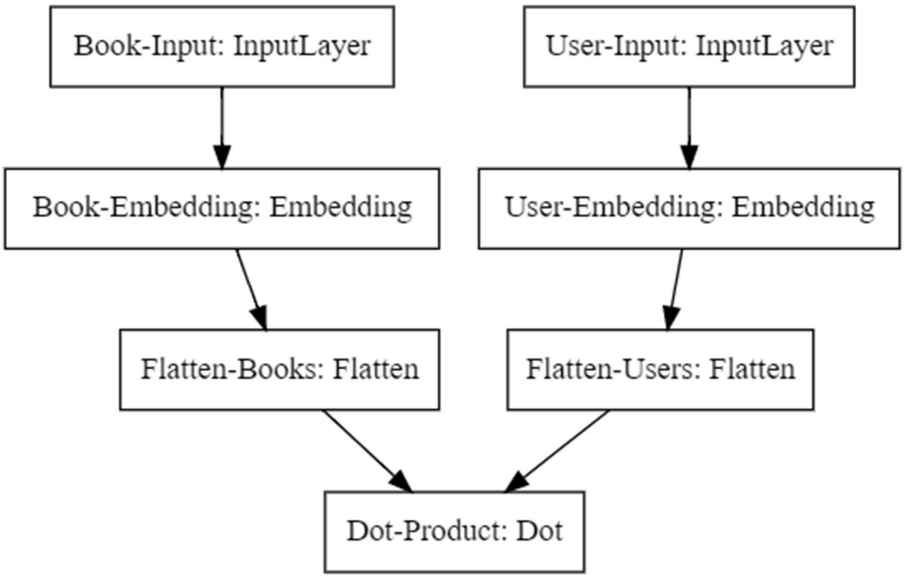

Figure 8.1: Block level summary of dot product model

Separate input layers for books and users were followed by an embedding layer with input dimension size of 53424 for users and 96832 for books and with output dimension size of 5. Outcomes of these embedding layers were flattened and were subjected to dot product. The result obtained from this simpler model is shown in Figure 8.2.

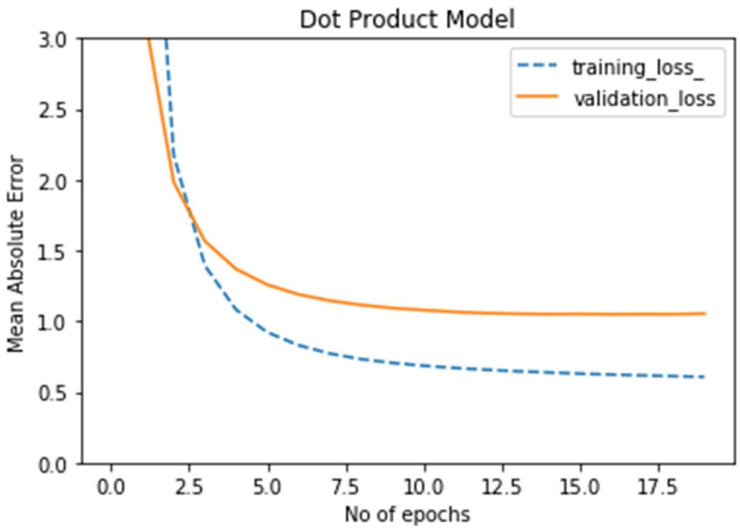

Figure 8.2: MAE vs No of epochs

This model with no dense layers showed poor performance. Under the course of 30 epochs, the training error (MAE) was brought down to 0.56 and still continued to

decrease whereas the validation error hardly reach the MAE of 1. The best validation error obtained from the model was found to be 1.24 at epoch 16.

Addition of dense layers introduced non-linearity in the model structure. Figure 8.3 show the model summary in the abstract form.

Instead of using dot layer to merge the embedding output from users and books, a concatenation layer was used providing freedom to the network itself to find the best metric to merge and a couple of dense layers were added increasing the chances for model to learn complex relationship from the user-item rating matrix. Dropout layer was added to prevent overfitting at early stages of training the model. The corresponding result is shown in Figure 8.4. Adding dense layers showed increased performance with MAE of 0.72 in validation set at epoch 24.

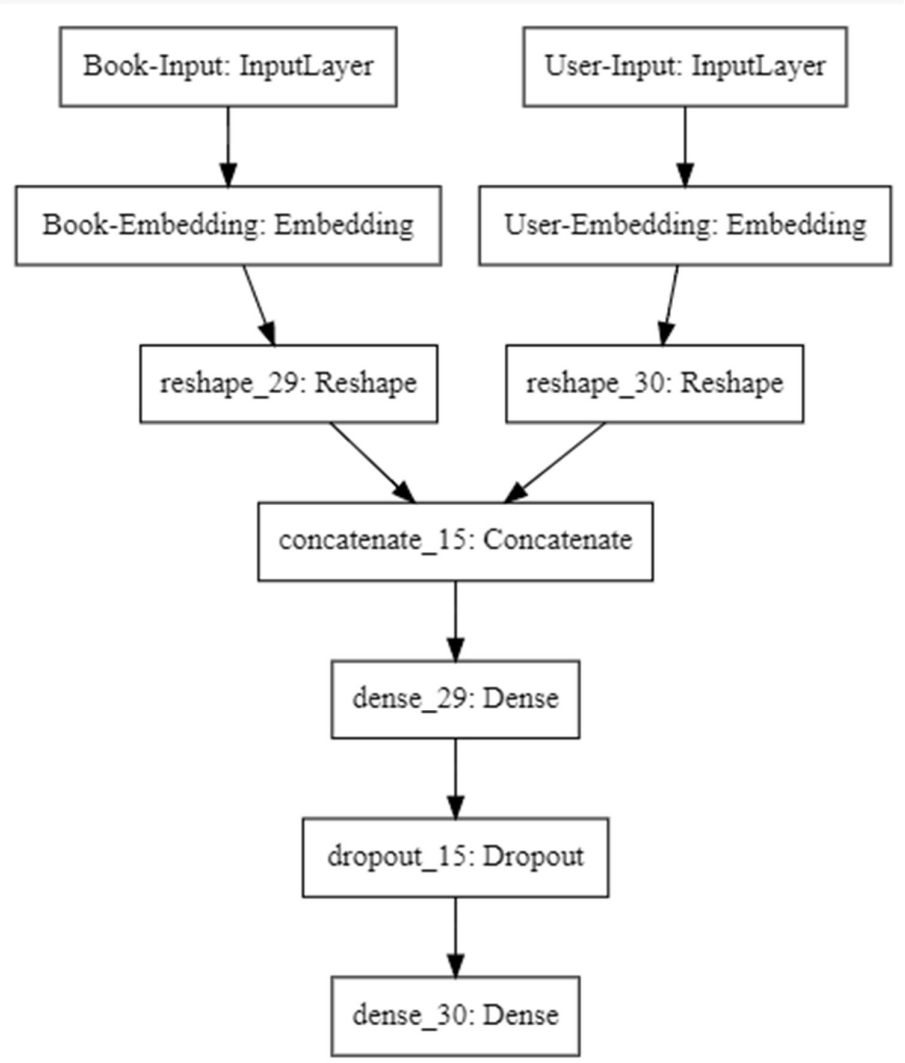

Figure 8.3: Block level summary of denser model

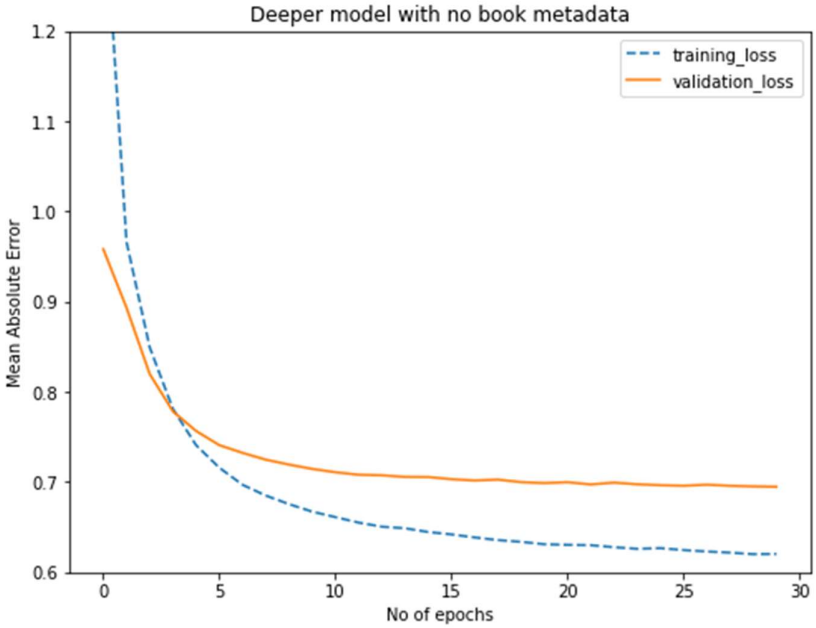

Figure 8.4: MAE Vs No of epochs

The dataset was prepared by combining multiple publicly available dataset including book crossing dataset and UCSD Book Graph dataset. The final dump contained book author and genre (style) as book metadata. Above model only accounts the collaborative side of recommendation with no consideration for cold start. In order to provide recommendation for users having provided rating to low number of books, incorporating book metadata along with collaborative approach is crucial. Figure 8.5 shows the model summary for hybrid recommender system.

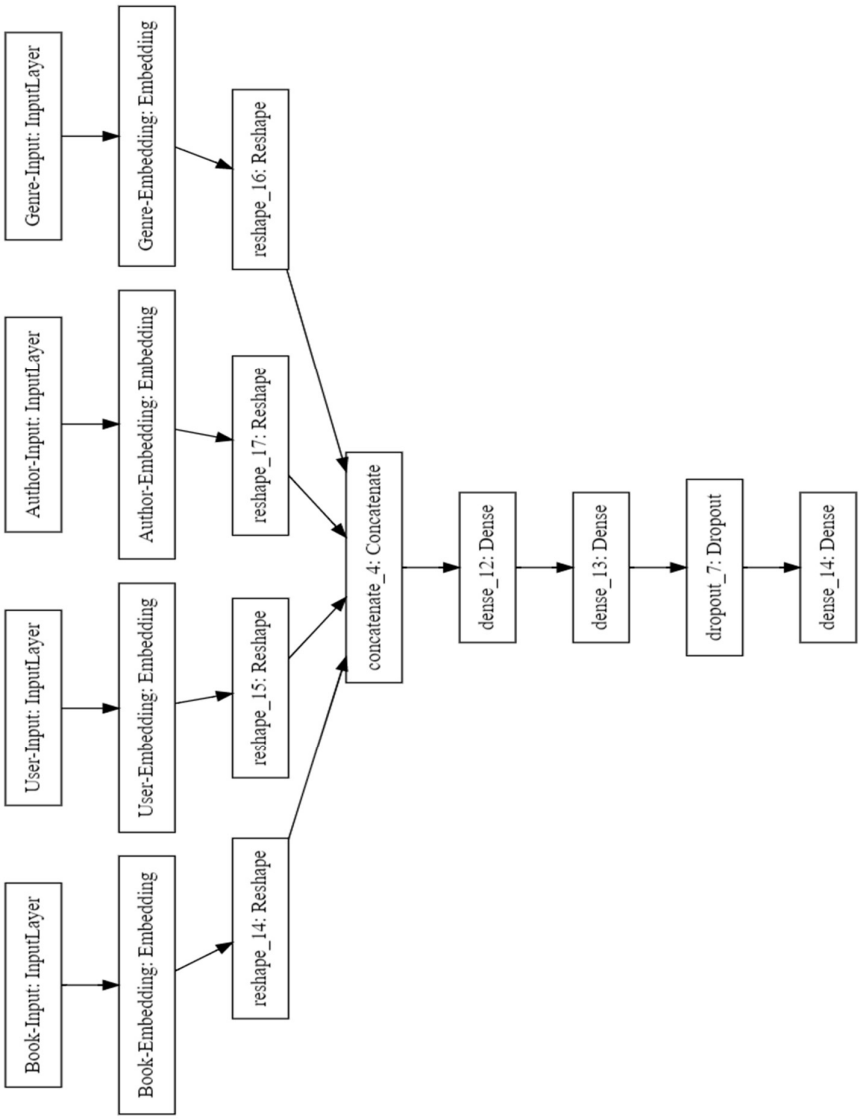

Figure 8.5: Block level summary of hybrid model

8.1.1. Hyperparameter tuning on book recommender model

A hyperparameter is a parameter whose value is set before the learning process begins. The outcome of deep learning model varies significantly with the values chosen for the hyperparameters. For the hybrid recommender model, embedding size (latent factor), batch size, learning rate and regularization stood out to be difference making parameters. The outcome of model was examined on different values of each of these parameters to see how they affect validation and test accuracy of the model.

8.1.1.1. Effect of Embedding Size

Number of latent factors (embedding size) plays a significant part in low rank matrix factorization. Embedding size corresponds to the number of columns and rows in the smaller matrices obtained after matrix factorization. Fig 8.6 and Fig 8.7 show the effect of embedding size on training and validation set. It can be concluded that with higher number of latent vector, the model converges quickly in the training phase but the performance gets poor as seen with embedding size of 20 in this case. The better value of embedding size was found to 5 as it showed descent performance in both training and validation set.

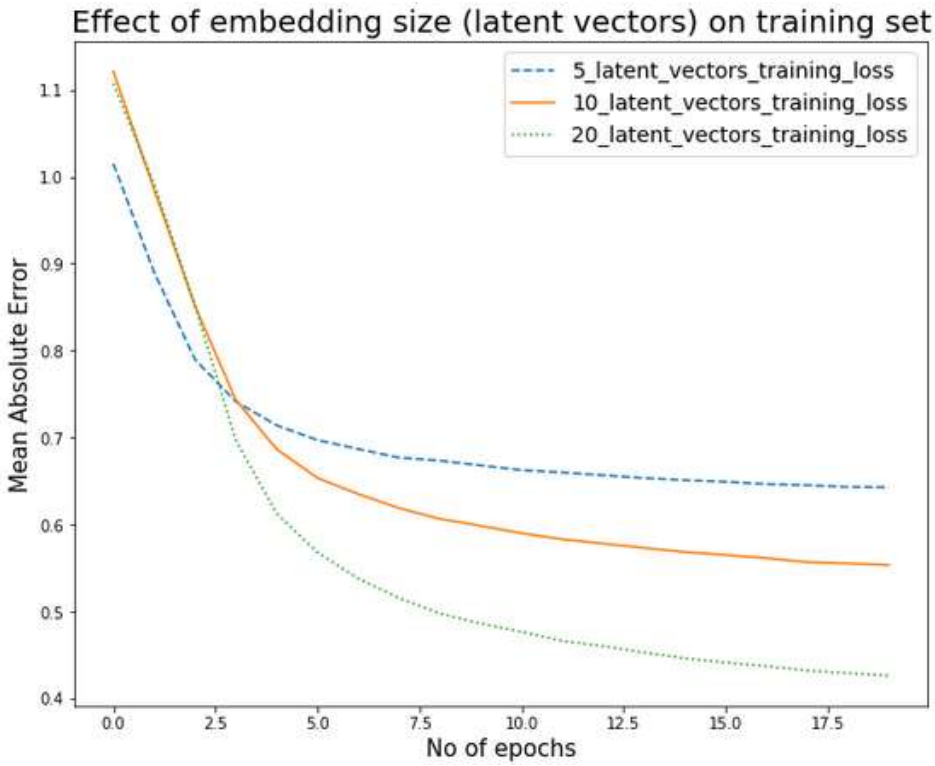

Figure 8.6: Effect of embedding size on training set

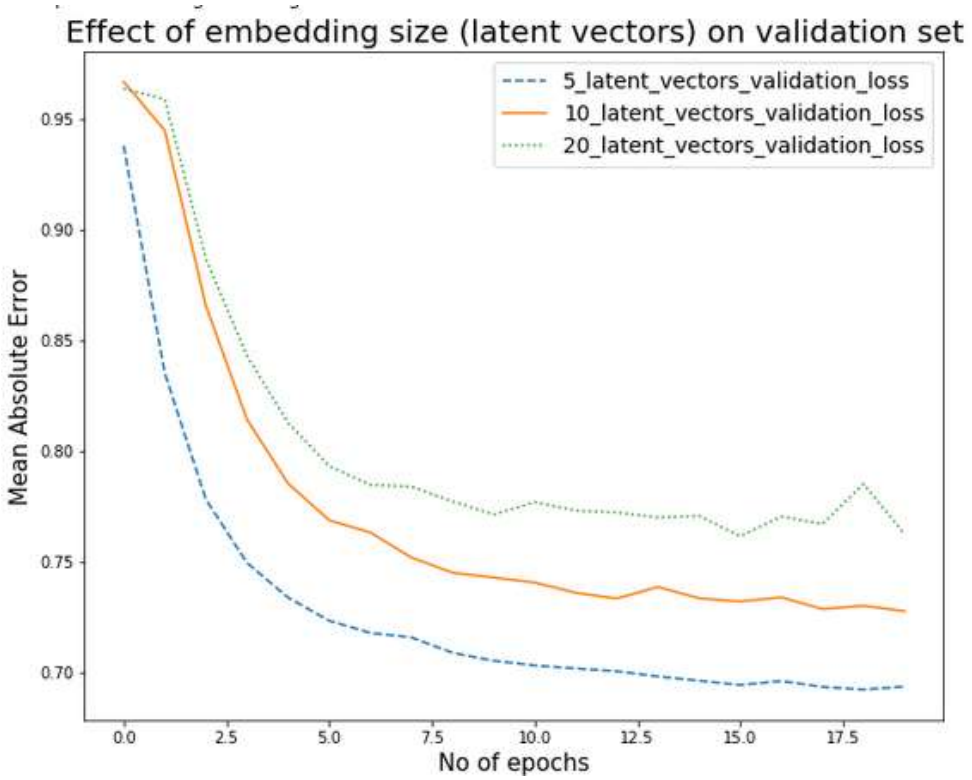

Figure 8.7: Effect of embedding size on validation set

8.1.1.2. Effect of Batch Size

Batch size refers to the number of samples that are going to be read before updating the parameter. For instance, let's say there are 1,000 training samples and batch size is set up to 100. The algorithm takes first 100 samples (from 1st to 100th) from the training dataset, calculates total cost and gradients, and finally updates all the parameters. Next it takes second 100 samples (from 101th to 200th) and train network again. This procedure is repeated until all the samples are propagated through the network. Figure 8.8 and 8.9 shows the MAE plotted against the number of epochs for batch size of 32, 64 and 128. Significant amount of effect were not seen on these values of batch sizes. However, the batch size of 64 produced much lower MAE in the validation set compared to others. Therefore, batch size of 64 was preferred.

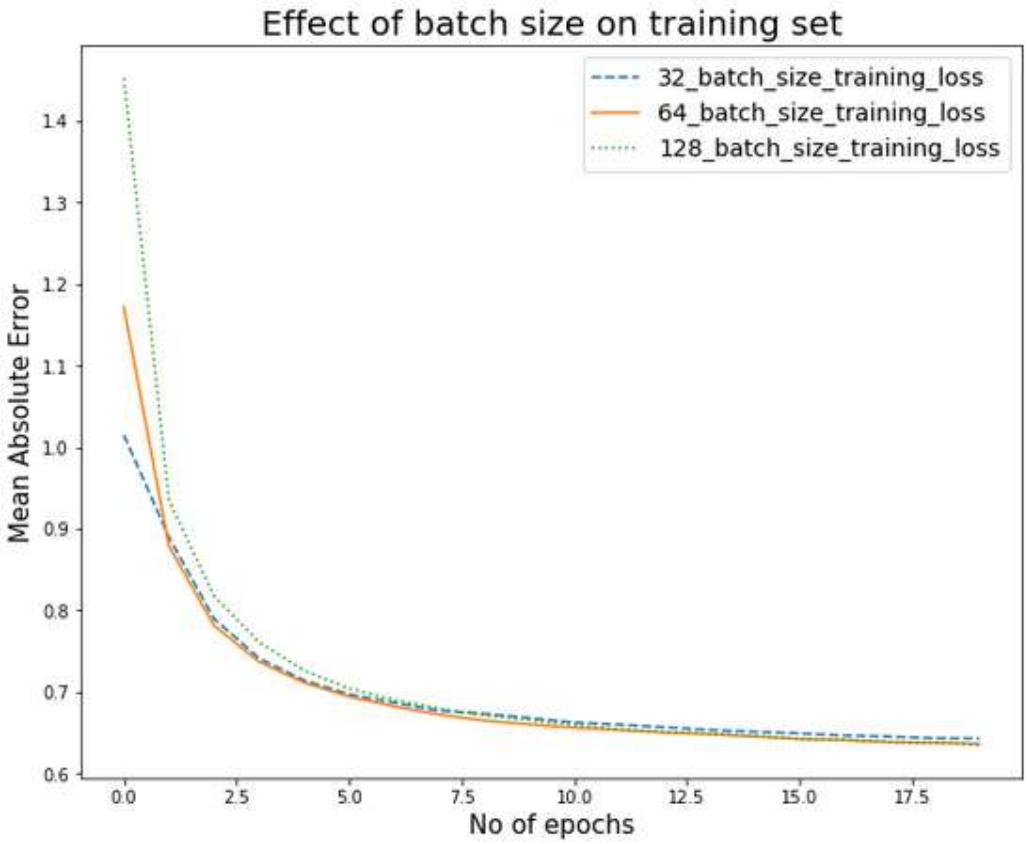

Figure 8.8: Effect of batch size on training set

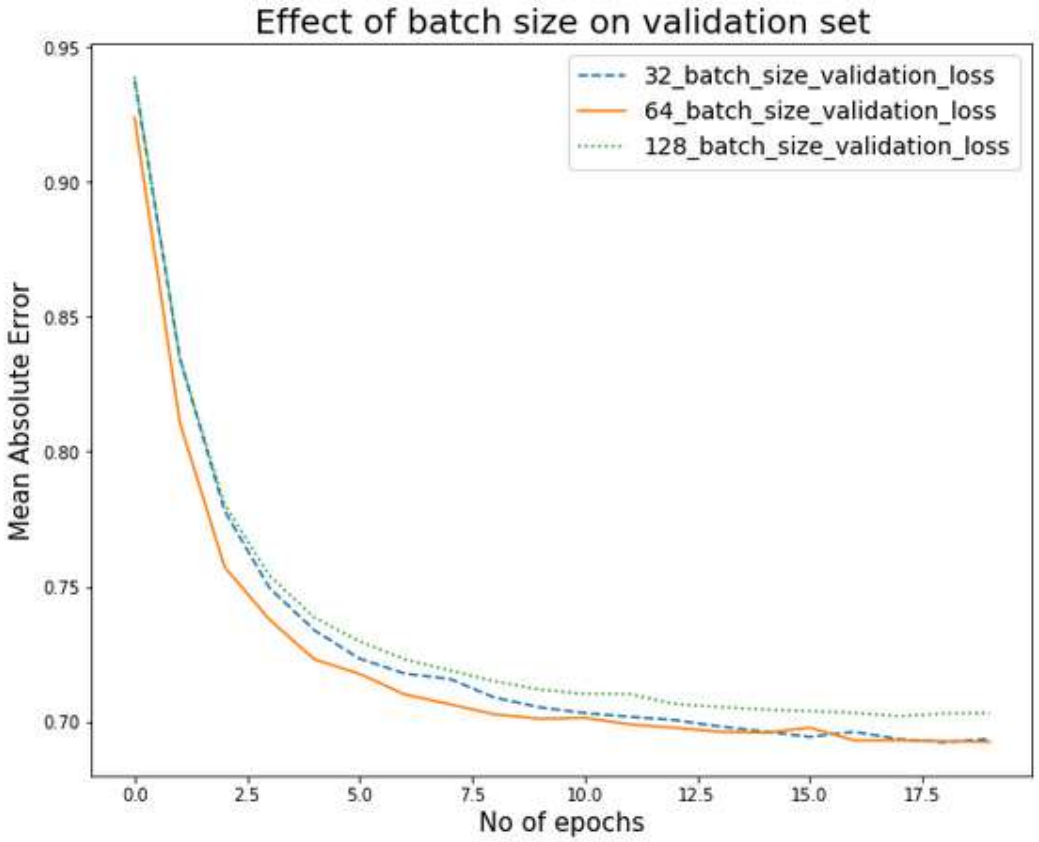

Figure 8.9: Effect of batch size on validation set

8.1.1.3. Effect of learning rate

The learning rate defines how quickly a network abandons old beliefs for new ones. If the learning rate is too low, the convergence will take too long. And if the learning rate is too high, the model will oscillate or diverge instead of converging. Therefore, it is essential to choose the right learning rate for optimum speed and performance.

In this project, since the Batch Gradient Descent (BGD) optimization algorithm (which is a modification of BGD) was used, the learning rate changes as the training progresses. The learning rate (per parameter) adapts to the parameter. However, it is necessary to choose the base learning rate.

Typical values of learning rate (0.1, 0.01 and 0.001) were chosen to see how they affect the performance of the model. The result is shown in Figure 8.10 and 8.11. It can be seen from the graph that the optimal learning rate is around 0.001 as the learning rate higher than this value started show severe spikes in the training set.

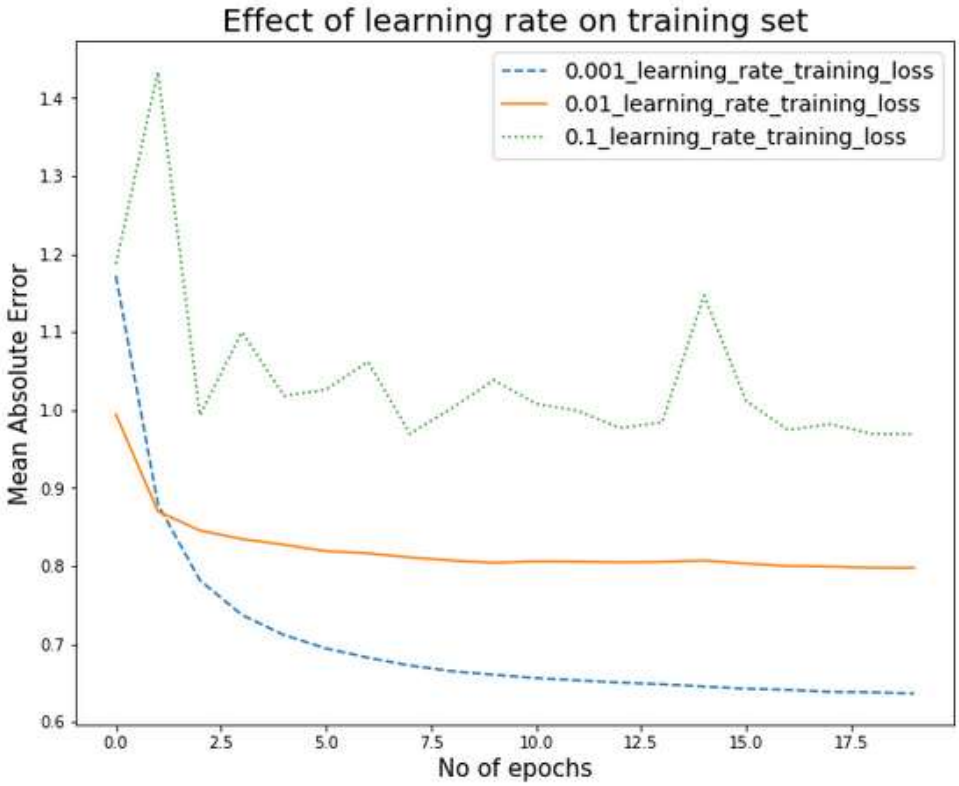

Figure 8.10: Effect of learning rate on training set

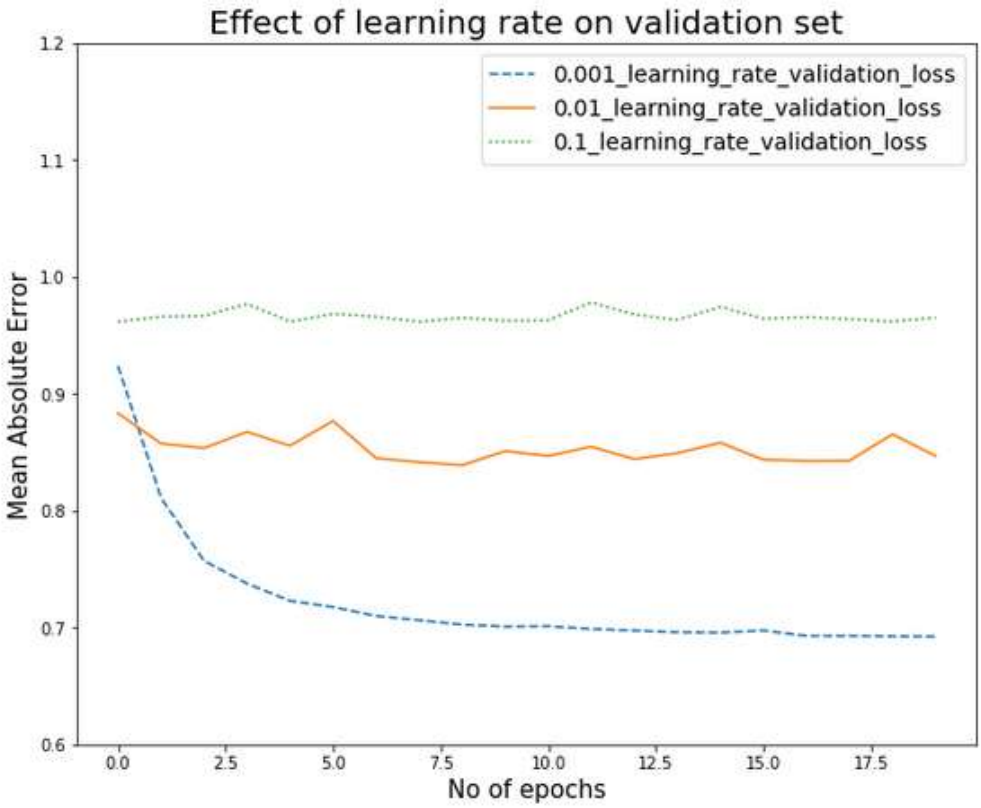

Figure 8.11: Effect of learning rate on validation set

8.1.1.4. Effect of regularization

Regularization is a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting. Figure 8.12 and 8.13 show the graphs for MAE plotted against regularization parameter l2. The graphs show that the performance of the model is similar for regularization value of 10^{-5}, 10^{-7} and 10^{-8} although regularization value of 10^{-5} showed slight improvement over the others in the validation set.

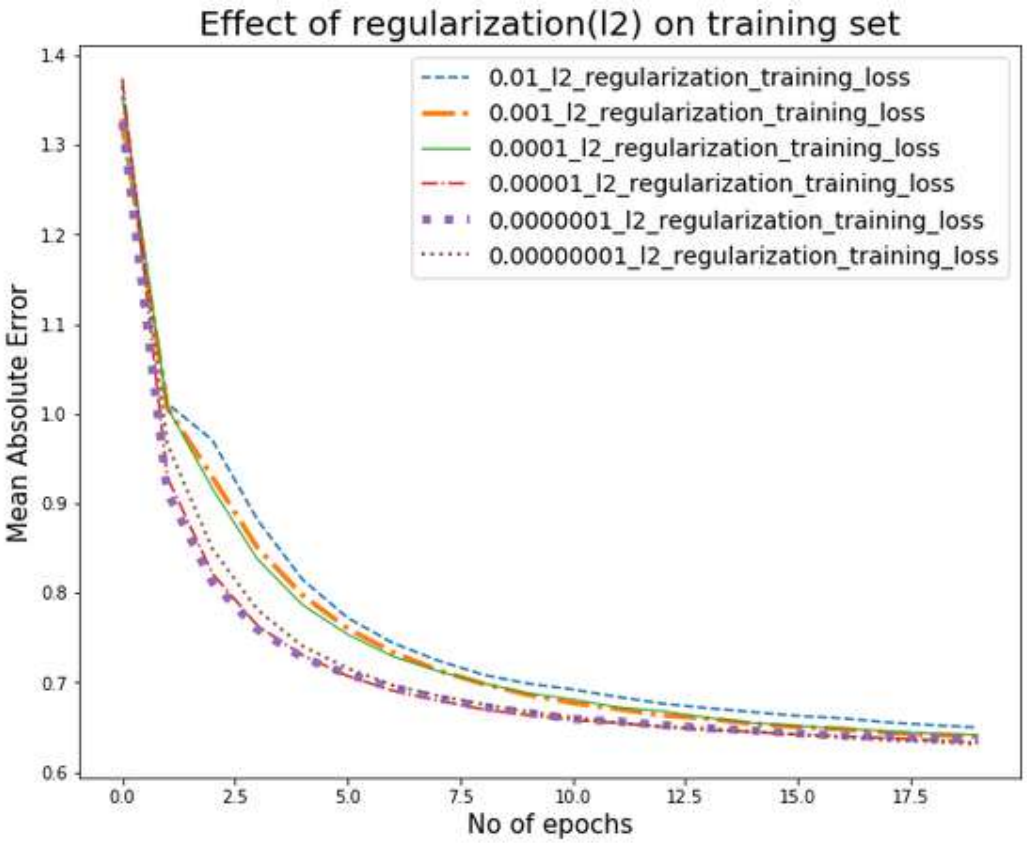

Figure 8.12: Effect of regularization on training set

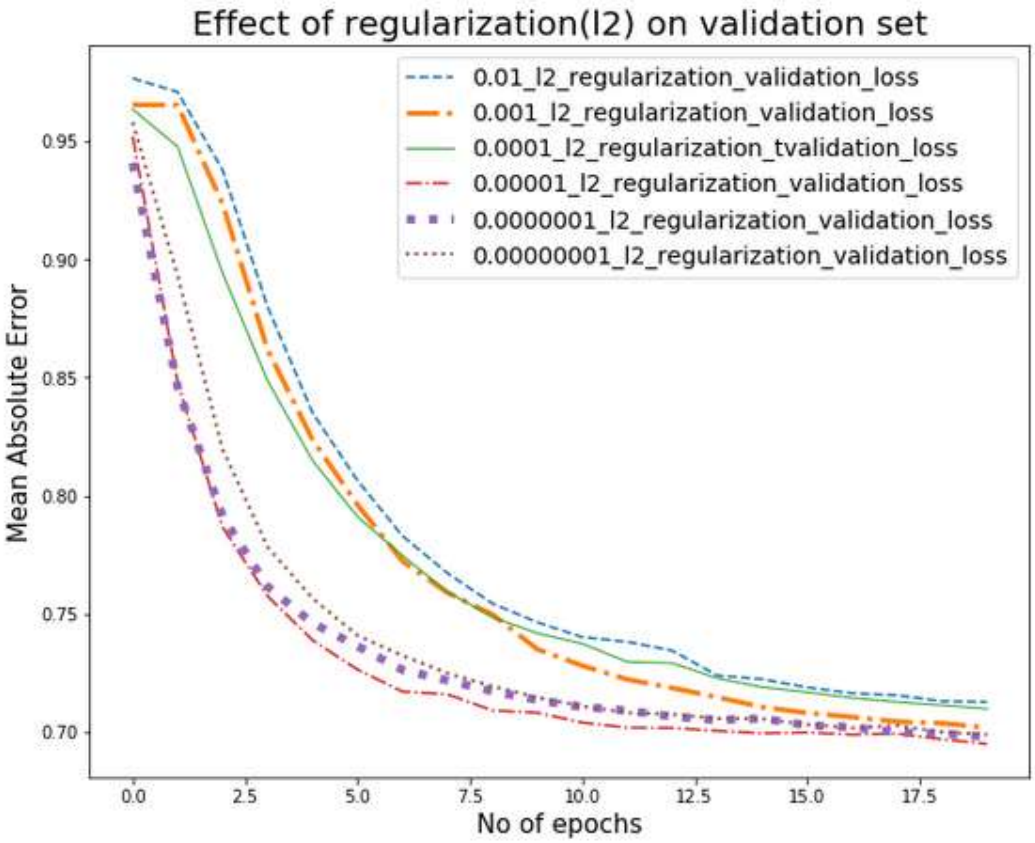

Figure 8.13: Effect of regularization on validation set

Even though the right practice to tune the network is to try out different values of a single hyperparameters keeping others constant and repeating the process, the final value thus gained may not be the best solution after all the hyperparameters have been chosen. In this case, the model showed improved performance for regularization value of 10^{-7} as opposed to 10^{-5} obtained when keeping rest of the parameters constant. Therefore, the model provided at 18 epoch, the best MAE of 0.7092 and 0.6955 in validation and test set respectively.

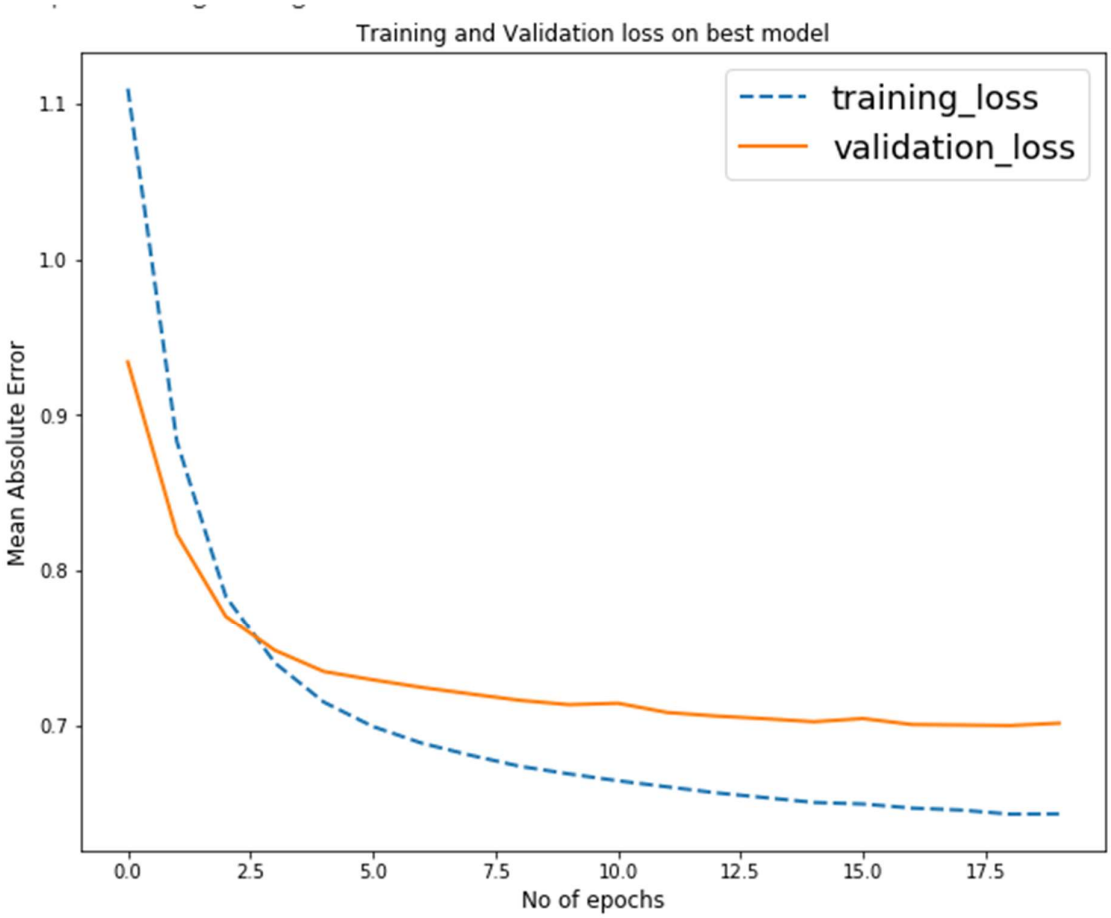

Figure 8.14: MAE Vs No of epochs for the best model

8.2. Sentiment Analyzer

After the data collection and its preprocessing the main tough job was to train and test the Deep learning model. Different types of models were trained and validated and tested on the datasets and were compared. The model with highest accuracy was used in the Project. The experiment was tested with simple RNN cell and was upgraded to Advanced RNN architectures to avoid the different kinds of hurdles. During the experiment in the Simple RNN cell the overall accuracy was only 47% which was very poor and couldn't classify the sentences properly. The next move was use of Advanced RNN architectures using LSTM which helped in solving the problem of Exploding Gradient Problem by removing Long Term Dependencies. After doing this the overall accuracy was increased to 86.25%, which was a great breakthrough for the Deep Learning Model. CNN also had

great performance and was able to classify the sentence with 85.28% accuracy. The following table shows the comparative analysis of the performance of the different models.

Summary (Epochs = 20, Hidden Layer = 2, Batch Size = 64)

Model	Train		Validation		Test	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
Simple RNN	0.693	50.35%	0.698	49.44%	0.708	47.48%
fastText	0.437	86.80%	0.392	85.09%	0.386	85.18%
CNN	0.18	88.78%	0.743	86.49%	0.339	85.28%
RNN + LSTM	0.103	89.23%	0.323	88.45%	0.293	86.25%

Table 7.7: Comparison among different models on classifying the sentiment

8.2.1. Outcomes of Upgraded RNN model

For this model different hyperparameters were tuned to observe the performance of the model in terms of the accuracy and loss. There are three partitions of the dataset. Training set, Test Set and Validation set. The model was trained in the training set, was validated with validation set and tested on the test set. The model provided different outcomes for different hyperparameter. The test accuracy was better in case of the RNN+LSTM model hence, this model was used in the project.

Effect of batch size and dropout has been represented by following graphs in terms of both accuracy and loss.

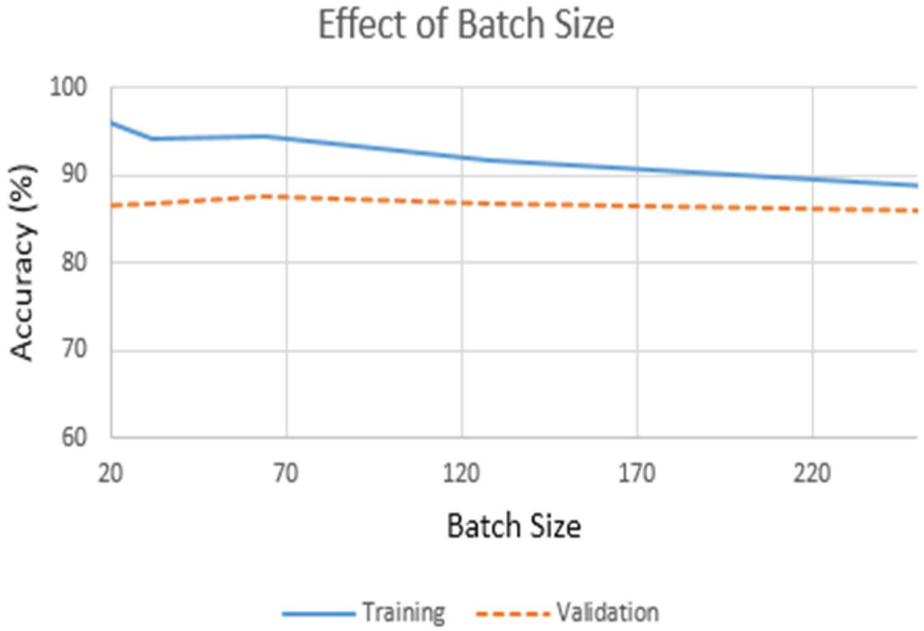

Figure 8.15: Effect of batch size on accuracy

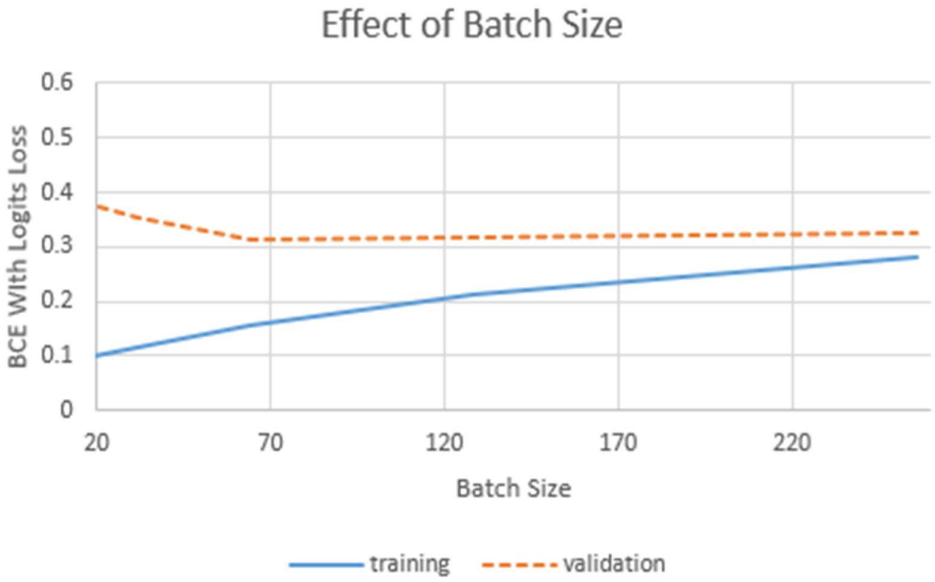

Figure 8.16: Effect of batch size on loss

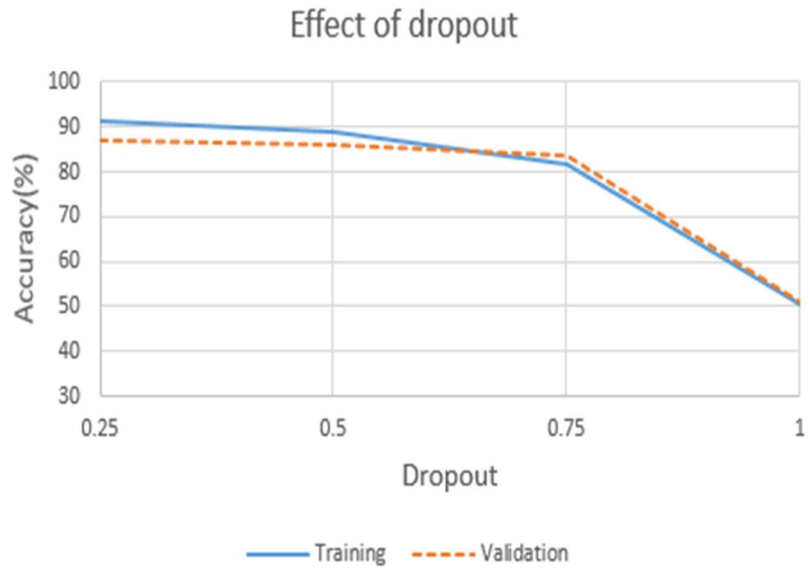

Figure 8.17: Effect of dropout on accuracy

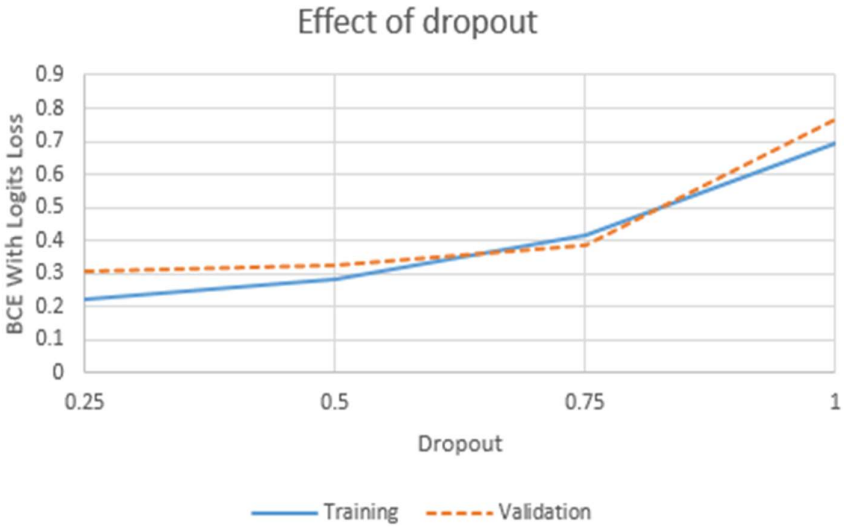

Figure 8.18: Effect of dropout on loss

After going through numerous training and testing phases, following set of parameters provided the best result.

Number of Epochs : 20

Batch Size : 64

Number of Filters : 100

Optimization Algorithm: Adam

Learning Rate : 0.001 (default for Adam)

Momentum : 0.9 (default for Adam)

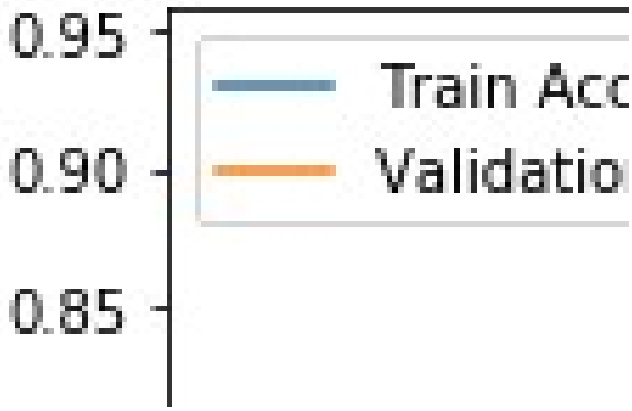

Figure 8.19: Epochs Vs Accuracy on validation set

8.2.2 Confusion matrix for evaluation

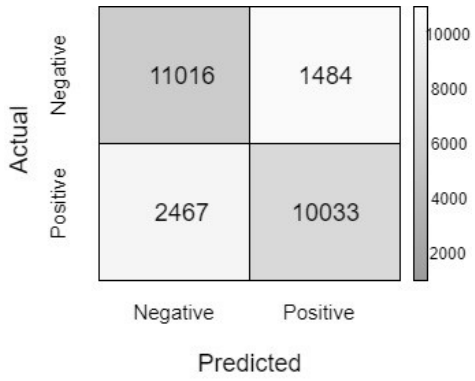

Figure 8.20: Confusion matrix for evaluation

The confusion matrix was plotted for test data sets which shows the number of sentences, classified truly or badly. The pure negative sentence is considered as 0 and pure negative sentence is considered as 1. The positivity of any sentence increased as the softmax output increases from 0 to 1. The below figure shows details of the sentences classified.

True Positive (TP) = 10033
True Negative (TN) = 11016
False Positive (FP) = 1484
False Negative (FN) = 2467

Metrics are calculated below:

$$Accuracy = \frac{TP + TN}{Total} = 0.8419$$

$$Error\ rate = \frac{FP + FN}{Total} = 0.1580$$

$$Senstivity(Recall) = \frac{TP}{Actual\ Postive} = 0.8026$$

$$Specificity = 1 - Senstivity = 0.1973$$

$$Precision = \frac{TP}{Predicted\ Postive} = 0.8711$$

CONCLUSION

The development of recommender system taking every user feedback is still in active research phase. This project comprises of many different pieces: collaborative filtering using matrix factorization, content-based filtering, web application and sentiment analysis and each of them took a fair amount to time to get started. A lot of research papers were considered regarding the architecture for making a hybrid recommender system. The project work started with the development of collaborative filtering model using matrix factorization. Meanwhile, the barebones for web application was also being made with django in the server side. Several days of research were carried out to choose relational or NoSQL database. Finally, we chose mongoDB as the database for storing every type of records.

Overall this project required a lot of time and work and doing it for a major project has been worth it. However, this project is not complete as of now. Content-based filtering is yet to be incorporated alongside existing model so as to develop a hybrid recommender system that will be able to tackle cold start problem. In addition, the sentiment analysis of book reviews are to be carried out and proper communication between django server and server on the raspberry pie is to be made so as to decide when to accept the ISBN number sent by the barcode interfaced with the raspberry pie.

9 LIMITATIONS AND FUTURE ENHANCEMENTS

The major limitations of this project are listed below:

- The recommender model used in this project is not highly scalable. There is a certain limit to the number of users that this model when deployed can serve. This is because the server has been kept in a single node and the serving capability depends the computational power of that node.
- The web application built to serve book recommendation lacks a number of features such as adding books to cart and reporting which should be included in a proper e-commerce web application.

The possible areas of future enhancements for this project are:

- In order to support large user base, the model should be trained on data spreading across cluster of machines. Scalability issue can be overcame by implementing MapReduce framework.
- A feature for a user to report about the recommendation he/she has been getting can be added to the website. Such reports can then be used for future trainings.

10 LIMITATIONS AND FUTURE ENHANCEMENTS

[1] M. Pazzani, A framework for collaborative, content-based, and demographic filtering, Artificial Intelligence Review 13 (1999) 393–408

[2] Alluhaidan, A. 2013. Recommender System Using Collaborative Filtering Algorithm, Technical Library: School of Computing and Information Systems

[3] Pathak, D., Matharia, S. & Murthy, C.N.S. 2012. NOVA: Hybrid Book Recommendation Engine", IEEE International Advance Computing Conference (IACC).

[4] J.A. Horrigan. Online shopping. *http://www.pweinternet.org/2008/02/13/online-shopping*, 2008 [Retrieved July 23, 2016].

[5] F.O. Isinkaye, Y.O Flolajimi, and B.A Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261 – 273, 2015

[6] Banea C, Choi Y, Deng L, et al.: CPN-CORE: A Text Semantic Similarity System Infused with Opinion Knowledge.In: Proceeding of Second Joint Conference on Lexical and Computational Semantics. Atlanta, Georgia, USA, 2013: 221

[7] Agirre E, Banea C.: SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability[C] SemEval 2015, June.

[8] Daniel Ramage, Anna N. Rafferty, and Christopher D.Manning.: Random walks for text semantic similarity. In Proceedings of the 2009 workshop on graph-based methods for natural language processing. Association for ComputationalLinguistics, 2009: 23-31.

[9] Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder and Bojana Dalbelo Bašić. 2012.: Takelab: Systems for measuring semantic text similarity. In: Proceedings of the Sixth International Workshop on Semantic Evaluation. Association for Computational Linguistics, papers 441-448.

[10] R. Amsler.: Application of citation-based automatic classification. Technical report, The University of Texas at Austin Linguistics Research Center, 1972.

[11] M. Kessler.: Bibliographic Coupling Between Scientific Papers, Journal of the American Documentation, Vol. 14, No. 1, pp.10-25, 1963.

[12] H. Small, “Co-citation in the Scientific Literature: A New Measure of the Relationship between Two Documents,” Journal of the American Society for Information Science, Vol. 24, No.4, pp. 265-269, 1973.[12]G Salton, A Wong, CS Yang.: A vector space model for automatic indexing[J], Communications of the ACM ,1975,V(11):613-620.

[13] Rafi M, Shaikh M S.: An improved semantic similarity measure for document clustering based on topic maps[J]. arXiv preprint arXiv:1303.4087, (2013).

[14] Leacock, C., and Chodorow, M. 1998.: Combining local context and WordNet sense similarity for word sense identification. In WordNet, An Electronic LexicalDatabase. The MIT Press.

[15] Zhibiao Wu and Martha Palme.1994.: Verb semantics and lexical selection. In: Proceedings of ACL, pages 133-138.

[16] Lin, D. 1998.: An information-theoretic definition of similarity. In: Proceedings of the International Conf. on Machine Learning.

[17] Resnik, P. 1995.: Using information content to evaluate semantic similarity. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence.

[18] Goikoetxea J, Soroa A, Agirre E, et al.: Random walks and neural network language models on knowledge bases[C]. In: Proceedings of NAACL-HLT. 2015: 1434-1439.

[19] Faruqui M, Dyer C.: Non-distributional word vector representations [J]. arXiv preprint arXiv:1506.05230, (2015)

[20] Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998.: An Introduction to latent semantic analysis. Discourse Processes, 25(2-3):259-284.

[21] J. Pennington, R. Socher, and C. D. Manning.: Glove: Global vectors for word representation. In: Proceedings of EMNLP, 2014.

[22] Turney, P. 2001.: Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In: Proceedings of the Twelfth European Conference on Machine Learning.

[23] Cilibrasi, R.L. & Vitanyi, P.M.B. 2007.: The Google Similarity Distance, IEEE Trans. Knowledge and Data Engineering, 19:3, 370-383.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013

[25] Quoc V Le and Tomas Mikolov.: Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053, 2014.

[26] Martín G H, Schockaert S, Cornelis C, et al.: Using semi-structured data for assessing research paper similarity [J]. Information Sciences, 2013, 221: 245-261

[28] Muyu Zhang, Bing Qin, Ting Liu, et al.: Triple based Background Knowledge Ranking for Document Enrichment.In: Proceedings ofCOLING, 2014

[29] G Salton, A Wong, CS Yang.: A vector space model for automatic indexing[J], Communications of the ACM ,1975,V(11):613-620.