Enhancing Book Recommendation with the use of Reviews

P. G. Sudasinghe

2019

Enhancing Book Recommendation with the use of Reviews

A dissertation submitted for the Degree of Master of Business Analytics

P. G. Sudasinghe

University of Colombo School of Computing

2019

DECLARATION

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: P. G. Sudasinghe

Registration Number: 2018/BA/033

Index Number: 18880331

[illegible] 13/09/2021

Signature: Date:

This is to certify that this thesis is based on the work of <u>Ms. P. G. Sudasinghe</u> under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by

Supervisor Name: Dr. H. A. Caldera

H. A. Caldera 13-9-2021

Signature: Date:

I would like to dedicate this thesis to

my family, friends, colleagues, and teachers who were an immense source of support and guidance throughout.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr. Amitha Caldera for his guidance and support throughout this project, coordinators Dr. Manjusri Wickramasinghe and Dr. Thilina Halloluwa for their effort in coordinating the Master of Business Analytics program and Individual Project respectively, and all the lecturers at University of Colombo School of Computing who have equipped me with the knowledge and skills required to complete this project.

ABSTRACT

Recommendation systems are a major component in current e-commerce websites and applications. There are many studies carried out to ensure that the best recommendations are provided to the user and conversion rate is increased. These techniques usually utilize historical transaction data and user ratings. While most of such websites also provide the capability to review the products bought by the users, the content of these reviews usually does not play a major role in recommendations made to the users.

Goodreads is the world's largest website for readers and book recommendations. A user can keep track of their reading as well as review, rate and recommend books to other users. Book recommendations are also made automatically by Goodreads based on the books a user has already read and rated. As a review is much more expressive than a single rating and tend to explain the user's decision for a rating, it is reasonable to expect that incorporating reviews will improve the recommendation process. This study attempts to address this by combining sentiment analysis of the user reviews with the recommendation process in Goodreads.

To achieve the above goal, the constructed recommender system utilizes LightFM, a Python library facilitating popular recommendation algorithms for implicit and explicit feedback. LightFM enables item and user metadata to be incorporated into traditional matrix factorization algorithms. The item metadata that is utilized in this scenario are the sentiment scores obtained through user reviews of each book.

The above recommender system performs better than pure collaborative filtering algorithms such as k-nearest neighbor and SVD for the same Goodreads dataset as evinced by the better AUC score of the LightFM model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Recommender Systems are software tools and techniques that provide suggestions for items that may be of interest to a user (Ricci, et al., 2011). These suggestions or recommendations may relate to various commercial and entertainment decision making processes. For example, a Recommender Systems (RS) may suggest products to the users of an e-commerce website such as Amazon, TV shows to users of Netflix, an entertainment provider, or a new book to users of GoodReads, a book recommendation website.

Recommender Systems came into being as a result of the observation that individuals regularly depend on recommendations made by others in making routine decisions; a person may decide on a book to read based on a suggestion from a friend or decide to watch a film after reading a review by a film critic.

The exponential growth and variety of information available on the Internet and the rapid introduction of new e-business services frequently overwhelmed users leading to the pressing need of a way to address this information overload. A RS aims to attempts to solve this by pointing the users to new or previously unseen items that maybe relevant to the users' current task. Information such as the user's preferences and behaviour are utilized to achieve this task.

One of the main features in websites and mobile applications mentioned above (Amazon, Netflix, eBay etc…) is facilitating users to rate and review items they have used. Recommender Systems utilize these ratings along with other interaction and behaviour data to come up with and finetune recommendations for the users.

Items are also often reviewed along with their rating by the users explaining the user's opinion or sentiment. Sentiment Analysis, also referred to as opinion mining, is the process of using natural language processing, computational linguistics, and many other techniques to automate the understanding of sentiments or opinions, often within user-generated content (UGC) as product reviews. The purpose of this is to understand people's position, attitude or opinion towards a certain entity or event and further to classify their polarity (Puschmann & Powell, 2018).

The advent of social media platforms has enabled their ever-increasing number of users to express their opinions through text, images, video, and audio freely. Consumer websites such as amazon, eBay and AliExpress and online travel companies like TripAdvisor, Booking.com and Agoda also facilitate sharing user opinions and ratings of goods and services they provide. All these have provided a rich and varied resource for sentiment analysis.

One of the most important advantages of sentiment analysis for a business is that it enables the companies to understand aggregated user feedback. This includes analysis and understanding customer feedback and target marketing. As a review is a lot more expressive than a rating, they could be utilized to explain the underlying dimensions of the users' decision on the rating (Sachdeva & McAuley, 2020).

Taking into account the above uses, if reviews are also taken into account when a suggestion is made to the user regarding a certain item; be it a book, TV show or a product to purchase; one may reasonably expect that the recommendation process will improve.

Both Sentiment Analysis and Recommender systems have been studied thoroughly as separate areas. There have been several works that explore the possibility of combining these two areas to achieve better recommendations in the past. These works will be studied, and the varying results of these approaches will be discussed in depth in the subsequent chapters.

1.1 Motivation

Most Recommender Systems at present generally depend on user ratings and do not consider reviews. Therefore, the review content is largely underutilized in the recommendation process.

Goodreads (Amazon.com, 2007) is considered world's largest site for readers and book recommendations. At present there are two main methods where a user receives recommendations for a book. First, the readers (users) can be recommended books explicitly by their friends. Secondly, there is a separate function that will list recommended books for the logged in user. This largely depends on the genre the reader has marked as interested in or the books marked as currently-reading. The automatic recommendation process utilizes this information and the user ratings of previous books read by the user. However, the reviews a large number of readers leave are not taken into account when recommending books to readers.

Goodreads has a scalar rating of 1- 5 stars each rating with the following definitions.

1 – “I didn’t like it”

2 – “It was ok”

3 – “liked it”

4 – “really liked it”

5 – “it was amazing”

Each book in Goodreads has a summary of its ratings (Figure 1) and reviews displayed.

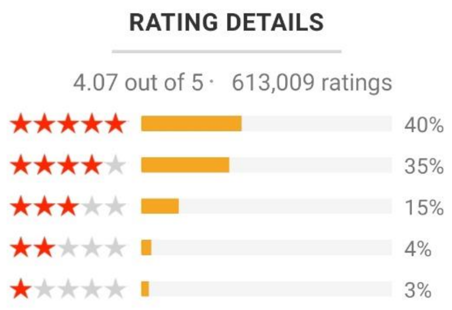


Figure 1: Rating summary of a book in Goodreads

A user can rate a book from 1 – 5 and additionally leave a text review as well. As mentioned earlier, a review is much more expressive and contains a lot more information than a mere rating and would indicate the readers’ opinion better.

1.2 Statement of the problem

This study attempts to improve book recommendations made to the users with the use of reviews based on the assumption that reviews contain much more information that may be useful in the recommendation process. As a mechanism of summarizing the information given in reviews, sentiment analysis will be used. The resulting sentiment scores will be used to improve the recommendation process.

1.3 Research Aims and Objectives

1.3.1 Aim

The ultimate goal of this study is to provide a better user experience to the end user by improving the book recommendations provided.

1.3.2 Objectives

The objective of this study is to first investigate the existing studies conducted in this area to identify the feasibility of combining sentiment analysis with item recommendation, shortcomings of the existing methods and possible improvements. At this stage the best recommender system model to incorporate review data is decided.

Secondly, utilizing the knowledge gained from the above study to investigate how user reviews may help to improve the Recommender Systems and implement a feasible solution.

Finally, this solution is evaluated, and inadequacies and potential improvements identified.

1.4 Scope

This study investigates how sentiment analysis of user reviews will augment recommendations made to the end user. The data utilized in this study are publicly available at the following URI:

https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home (Wan & McAuley, 2018)

Above datasets contain records of user reviews and ratings that may be used to implement and test the proposed solution.

The studies conducted by several researchers in this area has provided sufficient proof that utilizing this method may indeed improve recommendations. These studies and their outcomes will be summarized in Chapter 2.

VADER (Valence Aware Dictionary and Sentiment Reasoner) is used to perform sentiment analysis on the book reviews in the first phase of this study. Then, LightFM, a python library that facilitates building hybrid recommender systems incorporating user and item features will be used to build a recommender system. The performance of this will be compared against a few well-known collaborative filtering algorithms to test the assumption that the additional sentiment features do indeed improve the recommendations.

1.5 Structure of the Thesis

The next chapter of this thesis will contain the literature review of the existing studies done on the same area as well as discuss the theoretical background of the tools and technologies that will be used throughout this study. The subsequent chapter will contain the methodology and implementation details. The next chapters will detail results of the study and its evaluation, conclusion, and future work respectively.

CHAPTER 2
LITERATURE REVIEW

This chapter provides an introduction to the theories and tools used within this study. The first section will discuss recommender systems and their current status, and the subsequent section will discuss the progression of sentiment analysis and its current status. Finally, studies that utilize reviews and their sentiments into the recommendation process will be discussed.

2.1 Recommender Systems

Recommender Systems, also commonly referred to as Recommendation Systems, are software tools, algorithms and technologies that are used to provide item suggestions to users of certain systems. Here, “item” is the general term used to denote what the system recommends to users. They maybe products available for purchase in an e-commerce system, music available in a streaming service or a news article in a news website. These recommendations are personalized such that, the set of items suggested differ from user to user. Recommender systems are extensively used in various domains; for example, e-commerce (Amazon, Aliexpress, eBay), e-tourism (Tripadvisor, Booking.com, Expedia) and e-library (Amazon, Goodreads).

Service providers, such as e-commerce websites, will invest in state-of-the-art recommendation systems for various reasons. In a commercial RS, the most important function is to increase the sales. In a streaming service such as Youtube or Netflix, the goal is to increase the number of views; i.e. increase the conversion rate or the number of users that accept the recommendation and consume an item compared to users simply browsing through items in the absence of a recommending mechanism (Ricci, et al., 2011). Further advantages of RSs include increasing user satisfaction, increasing sales of diverse items, increasing user fidelity, and better understanding or customer needs and requirements.

RS also help users find appropriate items of good quality easily. They may also provide a context to the suggestions made according to the user’s preferences; for example, Goodreads will suggest new books to users if the user has read a book of the same genre or marked such a book as “to read” which denotes an interest in the genre.

For RS to perform its function, three main types of information are generally required.

Items, or the objects to be recommended that may be of varying complexity, value, or utility. Books, movies, and news are examples for low complexity items while insurance policies, digital cameras and mobile phones are examples of high complexity items.

In order to personalize recommendations, RS also require information about Users, which is another type of information. Users of an RS can have many diverse characteristics and unique goals. Information about users' past behaviour, preferences and characteristics will need to be represented and manipulated in an appropriate format to gain the best advantage from Recommendation systems.

Transactions are the interactions between Users and Items. This includes the ratings a user may give a certain product which will take a variety of forms such as (Schafer, et al., 2007):

1. Numerical ratings (e.g.: 1- 5 stars)
2. Ordinal ratings, such as "strongly agree, agree, neutral, disagree, strongly disagree"
3. Binary ratings in which the user is simply asked to decide if a certain item is good or bad.
4. Unary ratings only indicate that a user has observed or purchased an item, or otherwise rated the item positively.

2.1.1 Recommendation Tools and Technologies

To perform recommendation, the recommender systems should first predict if an item would be of interest to a particular user. In order to achieve this, the system must be capable of predicting the utility or usefulness of items or comparing the usefulness of several times and then decide which item needs to be recommended to the user. Various types of recommender systems have been studied to address this requirement. The taxonomy of recommender systems explained in (Burke, 2007) and (Ricci, et al., 2011) is listed below.

1. Collaborative Filtering (CF)

   In the simplest form (Schafer, et al., 2007) this approach recommends items to a user that other users with similar interests liked in the past. This only requires past information about rating profiles of different users. The similarity in interest is calculated based on the similarity in the rating history of the users.

   There are two main approaches to CF. Memory-based or neighborhood-based approach simply utilizes ratings of other users to predict the target user's ratings, whereas model-based methods assume an underlying generative model that explains the user-item interactions. This approach attempts to identify this model with the use of machine learning and probabilistic methods.

2. Content-based (CB)

   These recommender systems utilize two data sources to make recommendations: the features associated with products and the ratings that a user has given them. Content-based recommenders only require rating history of the target user. First a model or a profile on the user interests is created by analyzing a set of documents about the items that were previously rated by the target user. The recommendation process consists of matching the attributes of items against the CB recommender systems. In essence, CB recommender systems treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on product features.

3. Demographic

   A demographic recommender system categorizes and provides recommendations based on the demographic characteristics such as location, gender, or age of the user. Recommended products can be produced for different demographic classes, by combining the ratings of users in those classes. These type of recommender systems are frequently used in the Marketing domain.

4. Knowledge-based

   A knowledge-based recommender recommends items based on inferences about a user's needs and preferences. This requires specific domain knowledge about features of items and how these features meet the users' needs. Case-based recommender systems are one subcategory of knowledge-based systems where it estimates how much the user's needs match the items using a similarity metric. The other subcategory, constrain-based recommender systems, depend on the predefined explicit rules when matching customer requirements and items.

5. Community-based

   Also called Social recommender systems, these are based on the observation that people tend to rely more on recommendations made by friends or family rather than those made by anonymous individuals. The expansion of social media has raised the interest on these recommender systems as it is much easier to obtain details about an individual's connections through them.

6. Hybrid recommender systems

   The above discussed types of recommender systems have their advantages and disadvantages. Hybrid recommender systems were introduced with the intention of combining the strength of more than one recommender system and eliminating the weaknesses of those when used individually (Bansal, 2019).

There are several ways recommender systems can be combined together; Weighted, Switching and Mixed Hybrid Recommender Systems are examples for these.

2.1.2 Current Status

Apart from the main types of RS mentioned above, more avenues have been explored to perfect recommendation further. The following are some of the more recent technologies used that aims to achieve this objective (Bansal, 2019).

1. Genetic Algorithm Based Recommender Systems (GARS)

   Genetic Algorithm is an evolutionary approach utilized for optimizing an objective function where the selection strategy is applied on the solution candidates to a particular problem to ensure a better evolving solution. Studies have been carried out where the Genetic Algorithm is used in recommender systems to optimizing similarity function and clustering.

2. Deep Learning Based Recommender Systems (DLRS)

   Deep learning has proven to be an excellent approach in recommender systems in recent time due it's feature learning capabilities, inherent feature extraction and accuracy. Various deep learning techniques such as multilayer perceptron, deep factorization, recurrent neural networks, and convoluted neural networks have been studied and used to construct robust RS.

   The drawbacks of DLRS includes the large amount of data that is required and the "black box" nature of deep learning networks that limits the explainability of the hidden layers.

2.1.3 LightFM

LightFM (Kula, 2015) is a hybrid matrix factorization model which represents users and items as latent vectors (embeddings) similar to traditional collaborative filtering models. In addition to this, similar to a content-based model, it also utilizes linear combinations of embeddings of content features to describe each product and user. Thus, LightFM combines the advantages of content-based and collaborative filtering methods to:

1. Perform as well as content-based systems in cold-start and low-density scenarios.

2. Performs as well as traditional collaborative filtering methods when interaction data are available.

When both features and interaction data are available, LightFM performs significantly better than both CF and CB methods.

The LightFM model's prediction for user u and item i is given by the dot product of user and item representations, adjusted by user and item feature biases:

$$r_{ui} = f(q_u.p_i + b_u + b_i)$$

Here, q_u is the latent representation of user u and p_i is the latent representation of item i. b_u, b_i refers to the bias terms of user u and item i respectively. The latent representations of user and item are calculated by the summation of their feature vectors. The model learns user and item embeddings (latent representations) for user and items such that, they encode user preference over items using stochastic gradient descent methods.

There are many functions suitable for f such as identity functions or sigmoid function (when predicting binary data). LightFM implements four loss functions called logistic loss, Bayesian Personalized Ranking pairwise loss (BPR), Weighted Approximate-Rank Pairwise loss (WARP) and k-OS WARP.

If the user and item feature sets only contain indicator variables, the LightFM model reduces to the standard Matrix Factorization model.

2.2 Sentiment Analysis

According to the Cambridge Dictionary, a sentiment can be defined as a thought, an opinion, or an idea based on a feeling about a situation; or a way of thinking about an entity. Sentiment Analysis, also known as opinion mining, seeks to understand the sentiment or opinions expressed in various user generated content (UGC) such as text, audio and video posted online in platforms such as social media. The purpose of this is to determine people's position, attitude or opinion towards a certain entity or event (Puschmann & Powell, 2018) and further to classify their polarity.
Sentiment analysis is considered a branch of computational linguistics, which, since its inception in 1950s has been concerned with understanding and machine translation of natural languages. However, computational sentiment analysis only rose to prominence in early

2000s. The reason for this was the lack of sufficient and accessible volumes of natural language data that expressed opinions, sentiments and emotions rather than objective facts (Puschmann & Powell, 2018). However, the numerous free social media platforms available now (i.e. Facebook, Twitter, and Instagram) have allowed their users to express their opinions through text, images, video, and audio easily. Consumer websites such as Amazon, eBay and AliExpress and online travel companies like TripAdvisor, Booking.com and Agoda also enable sharing user opinions and ratings of goods and services provided by them. This provides an abundant resource of data that explains the minds of consumers eliminating one of the barriers early sentiment analysis techniques had.

Liu and Zhang (Liu & Zhang, 2012) further explain Sentiment analysis as the automatic identification of the entity, aspect, opinion holder and aspect's sentiment given a sentiment or an opinion. Entity is a product, service, an individual, organization or a topic about which is opinion is made. This is also referred to as the opinion target. An instance or a facet of an entity is an aspect.

The arrival of Web 2.0 and popularization of social media and its corresponding applications, as mentioned earlier, allowed millions of users to express their opinions and attitudes about various topics online. The resulting high-volume, high-variety, and high-velocity data was a catalyst for the automated sentiment analysis and its popularity at present.

Most well-known brands currently utilize some form of sentiment analysis tool across social media to gauge the opinions of users. By monitoring social media sites such as Twitter, a brand could detect sentiments of a certain user-base and react accordingly. Trends could also be tracked over time easily. By interacting with users proactively and responding to public opinion favourably would inarguably provide a brand an edge over its competitors.

Sentiment analysis could also aid managers and decision makers of a company in identifying how their or their competitors' brand and company reputation evolve over time. Another advantage is that this will help companies avoid potential public relations issues as real time sentiment analysis would facilitate identifying these issues sooner.

Perhaps the most important advantage of sentiment analysis for a business is that it enables the companies to understand aggregated customer feedback. This includes analysis and understanding Net Promoter Score (a tool that can be used to measure the loyalty of customer relationships of a business), customer feedback and target marketing. Further, using sentiment analysis to understand customer queries in an automated manner will increase customer care efficiency, ultimately leading to less customer dissatisfaction and customer churn.

Tourism is another service industry where sentiment analysis has great potential. Sentiment analysis on online user generated content has several advantages over the traditional data collection via surveys, interviews and questionnaires. These methods of obtaining customer opinions have several characteristic disadvantages. Surveys may reflect an inherently positive assessment due their investment in their travel while questionnaires only capture information about several pre-determined aspects (Alaei, et al., 2017). Social media sites and mobile applications such as Instagram, in contrast, present means to gather authentic and unsolicited opinions of travelers.

2.2.1 Sentiment Analysis Tools and Technologies

There are several techniques and algorithms used in sentiment analysis. The following diagram (Figure 2) summarizes the current such technologies (Rokade & Kumari D., 2019). There are two main categories of sentiment analysis; Lexicon Analysis calculates the polarity (e.g.: positive, negative, or neutral) with the usage of semantic orientation of words or phrases of a text document. A drawback in Lexicon analysis is that it does not consider the context. The other category, Machine Learning, involves building models from labelled data on a specific topic in order to find the orientation of a document belonging to the relevant topic. Both these methods have been used in a multitude of different domains such as politics, marketing, health, resulting in varying outcomes.

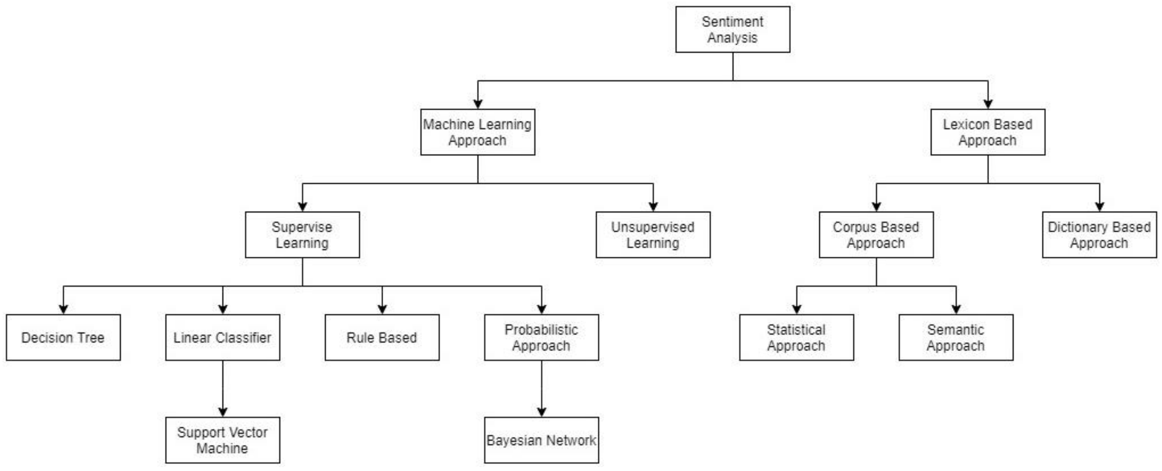


Figure 2: Types of Sentiment Analysis Approaches

Machine Learning Approach

Supervised learning and unsupervised learning algorithms are utilized to conduct the sentiment analysis. Supervised learning requires labelled datasets which are then used in algorithms such as Decision Tree, Support vector machines and Bayesian networks. In unsupervised approach, such labelled data are not available.

Both these approaches require training and testing datasets. For Supervised learning, the training dataset contains input feature vectors and their corresponding class labels. A classification model is developed using this training vector, which is subsequently tested using the testing dataset (Neethu & Rajasree, 2013).

Lexicon-based Approach

Lexicon-based sentiment analysis involves calculating the sentiment from the semantic orientation of the word of phrases in the text. This is based on the insight that the polarity of a document can be found with the aid of the polarity of the words that compose the said text. This approach utilizes sentiment dictionaries or lexicons such as WordNet or SentiWordNet (Rokade & Kumari D., 2019).

However, there are challenges to this approach that arise from the intrinsic complexity of natural languages. There are many studies carried out in addressing this as well as other challenges in lexicon-based approaches.

There are two main types of Lexicon-based sentiment analysis: Dictionary-based and Corpus-based.

1. Dictionary-based Approach

   This is the simplest method of lexicon-based sentiment analysis. Polarity is calculated using the presence of signaling sentiment words, also referred to as seeds, in the text. The polarity of each word can be determined using predefined dictionaries which contain positive and negative words and their synonyms and antonyms. The performance of this sentiment analysis method depends heavily on the dictionary that will be used.

2. Corpus-based Approach

   Unlike in the dictionary-based approach, in addition to a seed list and their sentiment labels, the context of the words in the form of syntactic patterns, is available as well. This solves the problem of words with context specific sentiment orientations that is not addressed in the previous method.

   There are two methods in corpus-based analysis: Statistical approach and Semantic approach. The former determines the polarity of an unknown word by calculating the relative frequency of co-occurrence with another word as it is observed that similar opinion words mostly appear together in a corpus and the latter assigns similar sentiment values to semantically close words.

2.2.2 Levels in Sentiment Analysis

As mentioned earlier, in Sentiment Analysis opinions are classified as positive, negative, or neutral. This is can be carried out in three levels.

1. Document Level

   At this level the entire document is considered as a single source or an individual entity in sentiment analysis. For example, the sentiment of an entire review will be classified where the document in question is the single review. The biggest challenge observed within this level of analysis is that not all the sentences of the document may be subjective. The accuracy depends on how well each sentence is extracted and individually analyzed.

2. Sentence Level

   The documents in the corpus is divided into sentences. This approach involves two steps; the first is categorizing a sentence as objective and subjective. The former will have no opinion attached and may contain only factual information. The latter will contain opinions which may be classified as positive or negative.

   The sentence level polarity can be determined using a grammatical syntactic approach, which takes the grammatical structure into account using part of speech tags. The same can also be achieved through a semantic approach.

3. Aspect/Feature Level

   This level is concerned with identifying aspects of a target entity and estimating the polarity of each mentioned aspect. This task is further broken down in to two subtasks. The first is Aspect Extraction, is an information extraction task where aspects of the entity are identified. This could be achieved by identifying and filtering highly frequent phrases in the text(document) with the aid of specific rules or determining the aspects in advance and finding them in the documents.

   The next subtask of Aspect Based Sentiment Analysis is sentiment classification. Once the aspects are identified, the sentiments for each aspect is grouped to arrive at the final polarity of for that aspect.

   This type of sentiment analysis provides the finest degree of sentiment analysis compared to document level and sentence level analysis

2.2.3 VADER – Valence Aware Dictionary and Sentiment Reasoner

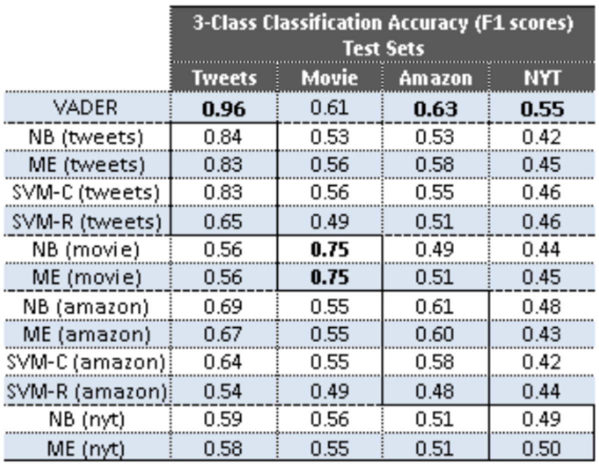
VADER (Gilbert & Hutto, 2014) is a rule-based tool available for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. VADER is available in the Natural Language Toolkit (NLTK) package. This can be applied directly to unlabeled text data which is an added advantage.

This works well on social media style text and also generalizes well to multiple domains. It is constructed from a generalizable, valence based, human-curated gold standard sentiment lexicon, thus does not require training data. This sentiment lexicon is sensitive to both polarity and intensity expressed in text.

Apart from the lexicon, VADER also uses five generalized rules based on grammatical and syntactic cues that indicate changes to the sentiment intensity, thus incorporating word-order sensitive relationships between terms. These heuristics include Punctuation which may increase the magnitude of the sentiment, Capitalization which may emphasize certain words, Degree modifiers (also known as intensifiers, booster words or degree adverbs) and Contrastive conjunctions such as ‘but’ which indicates a shift in the sentiment polarity. Finally, VADER also examines the tri-gram preceding a lexical feature to ensure negations are also identified correctly.

VADER performs best in three out of the four datasets the authors have used to test the model as the below set of results show (Figure 3). The authors also claim that this is fast enough to be used with streaming data and does not severely suffer from a speed-performance tradeoff.

Therefore, it can be assumed that out of the ready-to-use tools available for sentiment analysis at present, VADER is an excellent choice for sentiment analysis in review data.



	3-Class Classification Accuracy (F1 scores) Test Sets			
	Tweets	Movie	Amazon	NYT
VADER	**0.96**	0.61	**0.63**	**0.55**
NB (tweets)	0.84	0.53	0.53	0.42
ME (tweets)	0.83	0.56	0.58	0.45
SVM-C (tweets)	0.83	0.56	0.55	0.46
SVM-R (tweets)	0.65	0.49	0.51	0.46
NB (movie)	0.56	**0.75**	0.49	0.44
ME (movie)	0.56	**0.75**	0.51	0.45
NB (amazon)	0.69	0.55	0.61	0.48
ME (amazon)	0.67	0.55	0.60	0.43
SVM-C (amazon)	0.64	0.55	0.58	0.42
SVM-R (amazon)	0.54	0.49	0.48	0.44
NB (nyt)	0.59	0.56	0.51	0.49
ME (nyt)	0.58	0.55	0.51	0.50

Figure 3: Three-class accuracy (F1 scores) for each machine trained model (and the corpus it was trained on) as tested against every other domain context

2.3 Sentiment Analysis in Recommendation

In (Sachdeva & McAuley, 2020), the writers discuss to which extent reviews are useful for recommendation. They discuss the two approaches reviews have been used in the recommendation process up to now; as explanations for the recommendations made by the system or conversely, as ratings are considered much more expressive than a rating, they used to learn the laten features to perform better Matrix Factorization (MF).

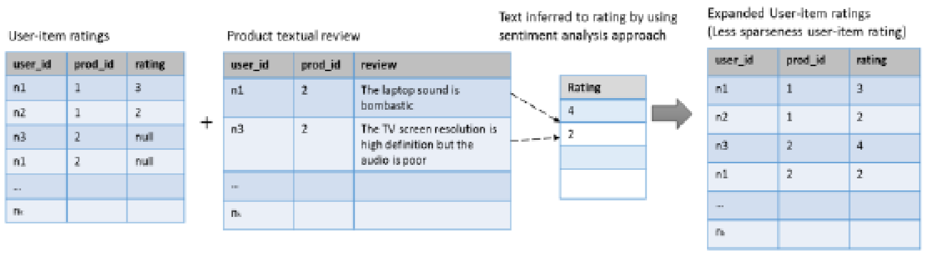
In this study, the writers have used the reviews as text instead of sentiment orientation, polarity or any other features that maybe extracted from the reviews. They conclude that while reviews may be considered important in recommendation, in cold start conditions they serve better as a regularizer rather than as more data to extract better recommendations.

Singh et al in 2011 have presented another research has explored a content-based recommender system with sentiment analysis to improve recommended movies (Singh, et al., 2011). This method was applied on a dataset of 2000 movies including the name, description, genre and 10 user reviews each from IMDB. These collected data was them transformed into term vectors.

First a content-based filtering was applied to acquire a list of movies that a particular user may be interested in. The cosine-similarity for the created vectors were calculated and the movies over a certain threshold were considered for this list. In the next step each movie in this list has been labelled as positive or negative based on the reviews. The authors have employed an unsupervised semantic orientation approach that computes sentiment of documents based on aggregated semantic orientation values of selected opinionated POS tags in it using Pointwise Mutual Information (PMI). The final recommendations contain only the movies that were labelled as positive.

This study handles the cold start problem by asking new users information on their interested genres as creating a vector with that information. The authors claim the hybrid methodology they have used provided them with recommendations of high accuracy and quality. Further, the final recommendations tended to also be rated sufficiently high overall.

A similar approach has been introduced in (Osman, et al., 2019) where the authors aim to eliminate domain sensitivity by elevating contextual information in conventional sentiment analysis and utilize this in a electronic product recommendation system. The writers believe that merging ratings with the review data could address the data sparsity problem as well.


+
Text inferred to rating by using sentiment analysis approach

User-item ratings

user_id	prod_id	rating
n1	1	3
n2	1	2
n3	2	null
n1	2	null
...		
nₖ		

Product textual review

user_id	prod_id	review
n1	2	The laptop sound is bombastic
n3	2	The TV screen resolution is high definition but the audio is poor
...		
nₖ		

Rating
4
2

Expanded User-item ratings (Less sparseness user-item rating)

user_id	prod_id	rating
n1	1	3
n2	1	2
n3	2	4
n1	2	2
...		
nₖ		

Figure 4:Merging ratings and textual reviews can improve data sparsity (Osman, et al., 2019)

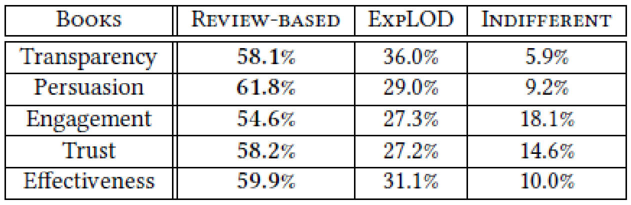
In this study, the authors have compared a standard collaborative filter recommender system, a collaborative filter system enhanced with sentiment ratings and a third system that incorporate contextual sentiment ratings. They have concluded that the recommender system with contextual sentiment ratings perform best in that domain as it addresses the issue of ambiguous wording in reviews.

A multi-criteria recommender system that exploits aspect-based sentiment analysis of user's reviews combined with collaborative filtering has been introduced in (Musto, et al., 2017). This study centers on restaurant datasets of several sources such as Yelp and TripAdvisor. First the aspects and relevant sentiments are extracted from the reviews using a framework referred to as SABRE (Sentiment Aspect-Based Retrieval). The sentiment scores are calculated using CoreNLP (which utilizes deep learning techniques) and AFINN which is a lexicon-based algorithm. These sentiment scores are then considered the ratings given by the users and utilized in user-based and item-based Collaborative Filtering algorithm to produce recommendations.

The authors confirm that when tested against several baseline methods, the multi-criteria method performs better and overcomes issues in single criteria approaches.

A hybrid recommender system that exploits aspect-based sentiment analysis of user's reviews combined with collaborative filtering has been introduced in (Musto, et al., 2019). The reviews ordinarily contain evidence about the aspects of an item that impressed the reviewer. The authors of this paper have come up with a method to exploit such information to generate a natural language justification that supports the recommendations provided by the recommender system that may induce a user to try the said item.

Here for each recommended item, the reviews are analyzed to extract distinguishing aspects that describe the item. This is achieved through a Part-Of-Speech (POS) tagging algorithm that extract representative nouns. These aspects are then ranked according to the relevancy and finally generate the justification utilizing a template-based structure. The results of this system were evaluated through user study measuring transparency, persuasion, engagement, trust, and effectiveness. The following figure contain the results of the Review-based method discussed this in paper and a comparison against a different method of justifying recommendation called ExpLOD.



Books	Review-based	ExpLOD	Indifferent
Transparency	58.1%	36.0%	5.9%
Persuasion	61.8%	29.0%	9.2%
Engagement	54.6%	27.3%	18.1%
Trust	58.2%	27.2%	14.6%
Effectiveness	59.9%	31.1%	10.0%

Figure 5: Results of experiment for Books domain (Musto, et al., 2017)

As the above studies show, it can be expected that integrating reviews and sentiments expressed in reviews are more likely to improve the performance of the recommender system than otherwise. All the above discussed research suggests further experimentation and study utilizing different datasets of different domains as future work.

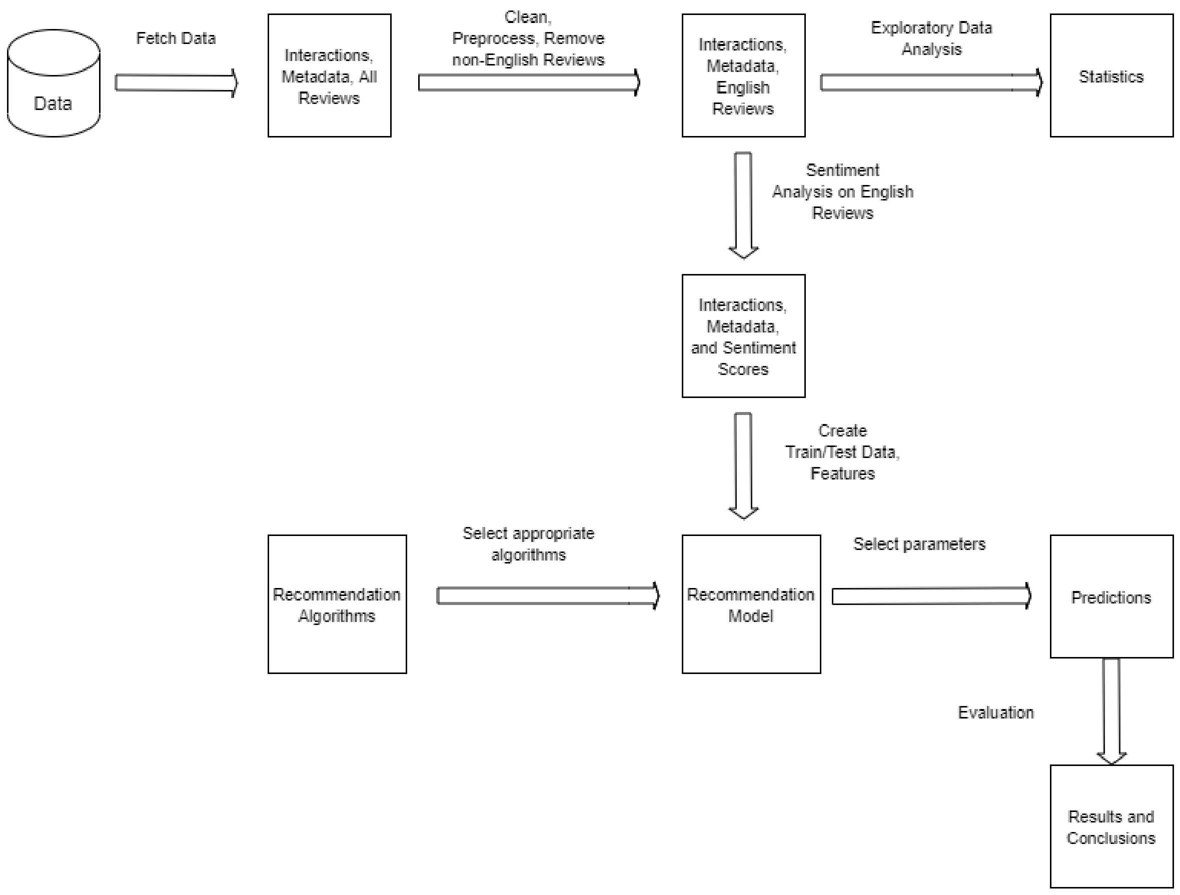
Therefore, utilizing such a method in book recommendations is worth investigating and one can optimistically expect better recommendation results.

CHAPTER 3
METHODOLOGY

This chapter contains the design process and methodologies employed in the attempt to enhance recommendations with review information. This will detail the data acquisition and processing, sentiment analysis and finally the construction of the recommendation system.

3.1 Workflow

The process followed throughout this study is shown in Figure 6.



Data
Fetch Data
Interactions, Metadata, All Reviews
Clean, Preprocess, Remove non-English Reviews
Interactions, Metadata, English Reviews
Exploratory Data Analysis
Statistics
Sentiment Analysis on English Reviews
Interactions, Metadata, and Sentiment Scores
Create Train/Test Data, Features
Recommendation Algorithms
Select appropriate algorithms
Recommendation Model
Select parameters
Predictions
Evaluation
Results and Conclusions

Figure 6: Workflow

The data used in this study is first cleaned and processed before a preliminary exploratory data analysis is performed. The nature of available data and its attributes are detailed in the subsequent sections of this chapter. Then, non-English reviews are removed, and sentiment scores are calculated that yields a dataset of user-book interactions, book metadata including sentiment scores for each book.

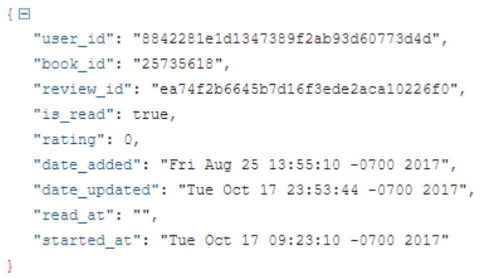
After investigating suitable recommendation algorithms, LightFM is selected as the most appropriate tool for this study due to its ability to feed item feature information to the recommendation model. The dataset obtained in the preceding steps are used to create a LightFM model which is then used to make recommendations.

The final step of this study evaluates the created model to observe its advantages over existing pure collaborative filtering methods and identify the future improvements that could be made.

3.2 Data

This study utilizes a publicly available dataset which contain user interaction, user review and book metadata information from Goodreads (Wan & McAuley, 2018). The dataset is categorized into several genres for ease of handling. This study will utilize the data in the Poetry category.
Goodreads Poetry dataset has 3 separate data files: information on books (metadata of items), user interactions with books, and review information.

Interactions



```
{ ⊟
   "user_id": "8842281e1d1347389f2ab93d60773d4d",
   "book_id": "25735618",
   "review_id": "ea74f2b6645b7d16f3ede2aca10226f0",
   "is_read": true,
   "rating": 0,
   "date_added": "Fri Aug 25 13:55:10 -0700 2017",
   "date_updated": "Tue Oct 17 23:53:44 -0700 2017",
   "read_at": "",
   "started_at": "Tue Oct 17 09:23:10 -0700 2017"
}
```

Figure 7: Example record of Goodreads Interactions

Interaction dataset contains 2701068 interactions between readers and poetry books and is in JSON format(Figure 7). The following variables are available in interactions:

'user_id', 'book_id', 'review_id', 'is_read', 'rating', 'review_text_incomplete', 'date_added', 'date_updated', 'read_at', 'started_at'

```
Read interactions:  1281024
Non-zero ratings :  1197248
is_read == True  :  1281024
is_read == False :  1420044
Read but unrated (is_read = True and rating = 0) books: 83776
```

A summary of ratings of all books and books marked as read are depicted in Figure 8. In this study only interactions with non-zero ratings are considered.

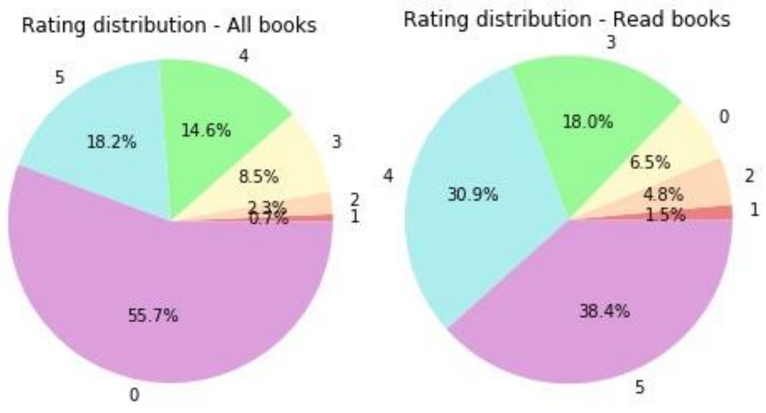


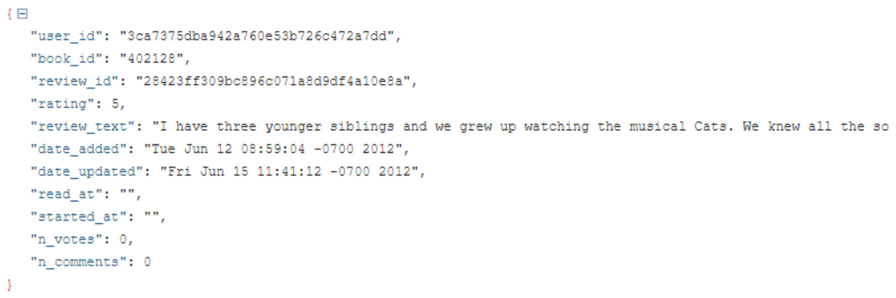
Figure 8: Ratings distribution

Reviews

This dataset contains 154555 records, each with a review for a poetry book. The following attributes are available.

['user_id', 'book_id', 'review_id', 'rating', 'review_text', 'date_added', 'date_updated', 'read_at', 'started_at', 'n_votes', 'n_comments']

The reviews may be of any language.



```
{
   "user_id": "3ca7375dba942a760e53b726c472a7dd",
   "book_id": "402128",
   "review_id": "28423ff309bc896c071a8d9df4a10e8a",
   "rating": 5,
   "review_text": "I have three younger siblings and we grew up watching the musical Cats. We knew all the so
   "date_added": "Tue Jun 12 08:59:04 -0700 2012",
   "date_updated": "Fri Jun 15 11:41:12 -0700 2012",
   "read_at": "",
   "started_at": "",
   "n_votes": 0,
   "n_comments": 0
}
```

Figure 9: Example record of Goodreads reviews

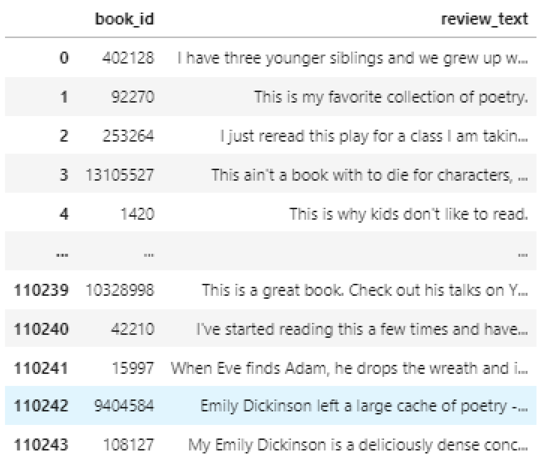
Book Metadata

The metadata contains all relevant details about books. Each book is identified by a unique ‘book_id’. In addition to book_id, the following data are also available:

['isbn', 'text_reviews_count', 'series', 'country_code', 'language_code', 'popular_shelves', 'asin', 'is_ebook', 'average_rating', 'kindle_asin', 'similar_books', 'description', 'format', 'link', 'authors ', 'publisher', 'num_pages', 'publication_day', 'isbn13', 'publication_month', 'edition_informatio n', 'publication_year', 'url', 'image_url', 'ratings_count', 'work_id', 'title', 'title_without_series']

3.2.1 Data Preprocessing

As mentioned above, the book reviews made by the users may be of any language. As only English reviews are utilized for this study, non-English reviews have to be filtered out.
Langdetect (Danlik, 2021) python tool was used in order to recognize non-English text in the reviews and filter them out. This is a python implementation of Nakatani Shuyo's language-detection library (Nakatani, 2010). This tool can identify 49 languages with 99.8% accuracy. The experiments conducted by the creators claim that this could identify English language with 100% precision, which recommends this tool as an adequate approach to filter out non-English reviews from the dataset.
At the conclusion of this step, 110244 reviews were identified as English.



	book_id	review_text
0	402128	I have three younger siblings and we grew up w...
1	92270	This is my favorite collection of poetry.
2	253264	I just reread this play for a class I am takin...
3	13105527	This ain't a book with to die for characters, ...
4	1420	This is why kids don't like to read.
...	...	...
110239	10328998	This is a great book. Check out his talks on Y...
110240	42210	I've started reading this a few times and have...
110241	15997	When Eve finds Adam, he drops the wreath and i...
110242	9404584	Emily Dickinson left a large cache of poetry -...
110243	108127	My Emily Dickinson is a deliciously dense conc...

Figure 10: English Reviews

3.3 Sentiment Analysis

The next step in the process is to perform sentiment analysis on the English reviews acquired in the previous step. This was performed with the use of VADER tool available in NLTK that was earlier described in Chapter 2.
VADER sentiment analyzer outputs four scores given a text; positive, negative, neutral, and compound. The compound score calculated by summing the valence scores of each word in the lexicon, adjusted according to the rules and then normalizing this sum between -1 and +1. -1 is the extreme negative while +1 is the extreme positive. A text can be classified as either positive, negative, or neutral according to the below criterion:

Sentiment Polarity	**Condition**
Positive	compound score >= 0.05
Neutral	(compound score > -0.05) and (compound score < 0.05)
Negative	compound score <= -0.05

Table 1: Sentiment Polarity according to VADER

As the below pie chart (Figure 11) shows, majority of the reviews were positive, i.e., the sentiment score calculated by VADER is larger than 0.05.

Table 2 depicts a sample of the reviews available in the Goodreads Poetry dataset along with the calculated compound sentiment intensity, the polarity. The rating corresponding to this review is also listed there.

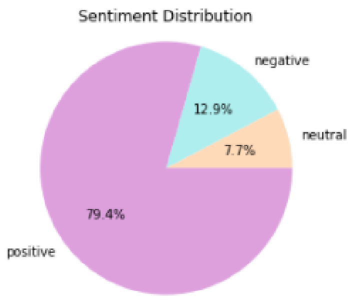
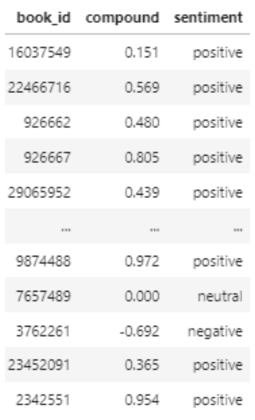


Figure 11: Sentiment distribution for all reviews

Review Text	**compound**	**sentiment**	**rating**
I have three younger siblings and we grew up w...	0.9766	positive	5
This is my favorite collection of poetry.	0.4588	positive	5
I just reread this play for a class I am takin...	0.9062	positive	5
This ain't a book with to die for characters, ...	0.9702	positive	5
This is why kids don't like to read.	-0.2755	negative	1
Odysseus is such an arrogant, power-hungry man...	-0.4939	negative	3
Beyond the Words is an anthology of poetry and...	0.9973	positive	4
Thoughts of a Pure Mind by Calvin Bland is a c...	0.9326	positive	0

Table 2: Review texts, sentiment polarity and ratings

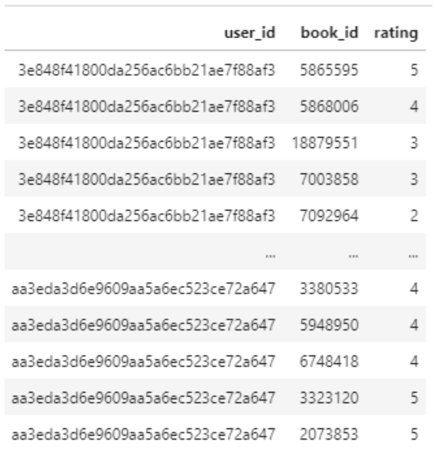
After sentiment scores per review are calculated, the average sentiment score per book was calculated. Figure 12 shows the calculated compound scores per each book available in the Book Metadata dataset.



book_id	compound	sentiment
16037549	0.151	positive
22466716	0.569	positive
926662	0.480	positive
926667	0.805	positive
29065952	0.439	positive
...	...	...
9874488	0.972	positive
7657489	0.000	neutral
3762261	-0.692	negative
23452091	0.365	positive
2342551	0.954	positive

Figure 12: Sentiment scores and polarity per book

At the end of this step, the dataset contains 265072 user-book interactions and metadata on 5667 books. Each user has at least rated 20 books, while each book is rated by at least 20 users. Each user is identified by "user_id" (e.g.: '561130041c7cbc45193e38b5cd9eea83') and each book is identified by "book_id" which is an integer value (Figure 13).



user_id	book_id	rating
3e848f41800da256ac6bb21ae7f88af3	5865595	5
3e848f41800da256ac6bb21ae7f88af3	5868006	4
3e848f41800da256ac6bb21ae7f88af3	18879551	3
3e848f41800da256ac6bb21ae7f88af3	7003858	3
3e848f41800da256ac6bb21ae7f88af3	7092964	2
...	...	...
aa3eda3d6e9609aa5a6ec523ce72a647	3380533	4
aa3eda3d6e9609aa5a6ec523ce72a647	5948950	4
aa3eda3d6e9609aa5a6ec523ce72a647	6748418	4
aa3eda3d6e9609aa5a6ec523ce72a647	3323120	5
aa3eda3d6e9609aa5a6ec523ce72a647	2073853	5

Figure 13: User-book interactions

3.4 Recommendation

In the studies conducted up to now, various researchers have tried different methods in combining sentiment polarity or sentiment scores and recommendation algorithms. As detailed in Chapter 2, more researchers have suggested that using additional information such as sentiments tend to work better when they are used alongside interaction and rating data, instead of relying purely on sentiment data.

There are many recommendation algorithms that have been introduced in the past few years. In this study, the focus is to identify if output of the above sentiment analysis step could be utilized to improve recommendation of books.

As a benchmark, two simple recommendation systems were created using K-Nearest

Neighbor and SVD based Collaborative Filtering algorithms, available in Surprise Python library (Hug, n.d.) using the interactions dataset. No additional features were considered here.

k-NN Collaborative Filtering

Using a k-NN CF approach (surprise.prediction_algorithms.knns.KNNBasic), an item-based recommender system was created. In k-NN collaborative filtering algorithm the prediction for a particular user *u* and item *i* is calculated with the use of similarity between two items.

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)}$$

Here *sim(i,j)* refers to the cosin similarity between item i and j and r_{uj} is the rating user *u* has given item *j*.

The best parameters that minimized the RMSE were k = 33 and min_k =3. Here k is the maximum number of neighbors taken into account for aggregation, while min_k is the minimum number of neighbors considered. If there are not enough neighbors, the prediction is set to the global mean of all ratings. The ROC curve and the AUC score of 0.61 are depicted in Figure 14.

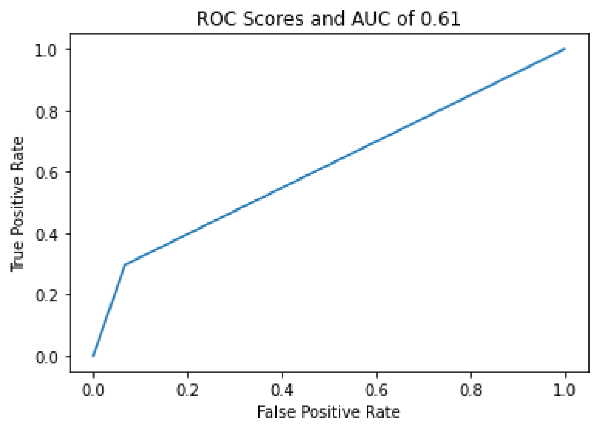


Figure 14: k-NN Based CF - ROC Curve

SVD Collaborative Filtering

The second recommender system that was created was based on the SVD algorithm available in Surprise (Hug, n.d.) under Matrix Factorization-based algorithms, which yielded the below ROC curve and an AUC value of 0.6 (Figure 14) for the minimum RMSE value of 0.82.

In SVD CF, the prediction for $\widehat{r_{ui}}$ is calculated as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Here, μ is the average rating of all items. Terms b_u and b_i are the bias terms for user *u* and item *i*, where each refers to the average ratings given by user *u* and average rating of item *i* minus μ respectively. q_i and p_u represents each item and user.

The AUC score of this CF algorithm was 0.6 as depicted in Figure 15.

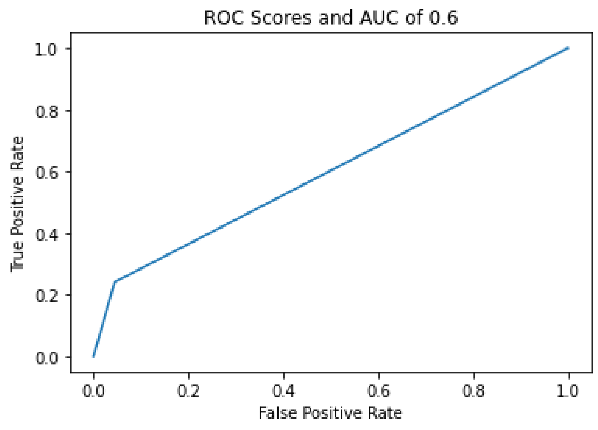


Figure 15: SVD ROC Curve

3.4.1 LightFM Model

As described in Chapter 2, LightFM is a Python library that facilitates combining CF filtering approach with item and user feature to provide better recommendations. As the sentiment score per each book could be considered a feature of a book, in this study, sentiment scores would be utilized.

To create a model, first the required user item interaction data and item features must be created in a way that LightFM model understands. The subsequent sections of this chapter will explain how this is done in detail.

3.4.2 LightFM Data Preparation

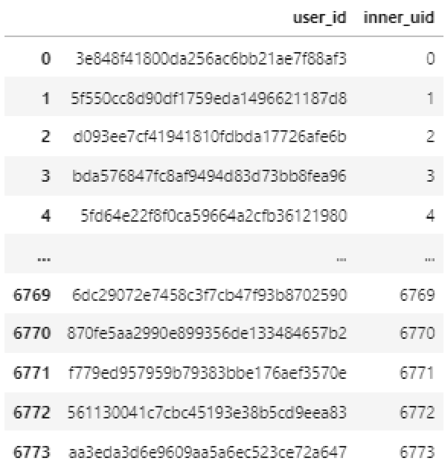
LightFM requires the user item interactions to be in the form of a numpy.float32 coo_matrix of shape [n_users, n_items]. User and item features should be in the form of a numpy.float32 csr_matrix of shape [n_items, n_item_features].

To obtain these types of interaction and feature matrixes, LightFM has provided a class called Dataset. Once a Dataset object is created, fit() method is used to create user/item_id and feature mappings. The build_interactions() and build_item_features() methods available in this class will be used to create the interaction and feature matrices as desired.

To obtain item_features in this study, books and their relevant sentiment scores are provided to the Dataset.

LightFM creates inner user and item IDs to be utilized in training, testing and predictions. These user id and item id mappings can also be obtained from the Dataset class. These inner IDs are of type numpy.int32. Figure 16 depicts the user id mappings converted to a dataframe object.

Data obtained as above is then be split into Training, Testing and Validation datasets using the random_train_test_split function provided by LightFM. This method does not ensure that all items and users with interactions in the test set also have interactions in the training set.



	user_id	inner_uid
0	3e848f41800da256ac6bb21ae7f88af3	0
1	5f550cc8d90df1759eda1496621187d8	1
2	d093ee7cf41941810fdbda17726afe6b	2
3	bda576847fc8af9494d83d73bb8fea96	3
4	5fd64e22f8f0ca59664a2cfb36121980	4
...	...	...
6769	6dc29072e7458c3f7cb47f93b8702590	6769
6770	870fe5aa2990e899356de133484657b2	6770
6771	f779ed957959b79383bbe176aef3570e	6771
6772	561130041c7cbc45193e38b5cd9eea83	6772
6773	aa3eda3d6e9609aa5a6ec523ce72a647	6773

Figure 16: User ID Mappings

3.4.2 LightFM Model Creation

LightFM model has the following parameters that could be set upon initialization. A few important parameters are listed below.

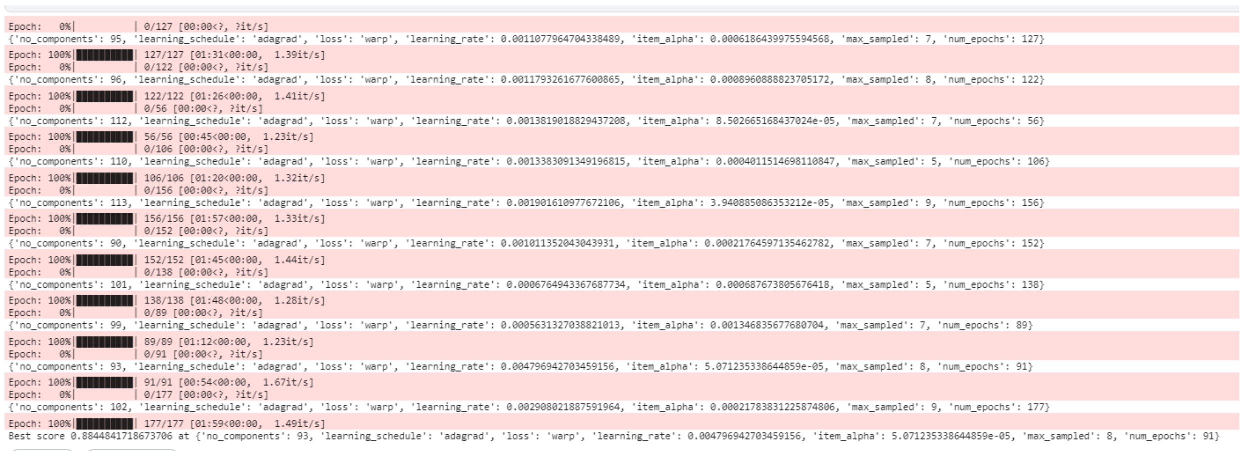
Parameter	Description	Best Parameter Value
no_components	the dimensionality of the feature latent embeddings.	93
learning_schedule	one of (‘adagrad’, ‘adadelta’).	‘adagrad’
loss	one of (‘logistic’, ‘bpr’, ‘warp’, ‘warp-kos’): the loss function.	‘warp’
learning_rate	initial learning rate for the adagrad learning schedule. rate initial learning rate for the adagrad learning schedule.	0.004797

item_alpha	L2 penalty on item features	5.071e-05
max_sampled	maximum number of negative samples used during WARP fitting	8

Table 3: LightFM Model parameters

To obtain the model with the best AUC score, the above hyperparameters have to be properly tuned. Figure 17 shows the output of such a process. The best values obtained in this study are also available in the Best Parameter Value column in the table above (Table 3).

Further, as the best AUC value is 88.4%, it appears to be better than the simple k-NN and SVD based collaborative filtering recommender systems that were created with Surprise.



```
Epoch:   0%|          | 0/127 [00:00<?, ?it/s]
{'no_components': 95, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.0011077964704338489, 'item_alpha': 0.0006186439975594568, 'max_sampled': 7, 'num_epochs': 127}
Epoch: 100%|██████████| 127/127 [01:31<00:00,  1.39it/s]
Epoch:   0%|          | 0/122 [00:00<?, ?it/s]
{'no_components': 96, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.0011793261677600865, 'item_alpha': 0.0008960888823705172, 'max_sampled': 8, 'num_epochs': 122}
Epoch: 100%|██████████| 122/122 [01:26<00:00,  1.41it/s]
Epoch:   0%|          | 0/56 [00:00<?, ?it/s]
{'no_components': 112, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.0013819018829437208, 'item_alpha': 8.502665168437024e-05, 'max_sampled': 7, 'num_epochs': 56}
Epoch: 100%|██████████| 56/56 [00:45<00:00,  1.23it/s]
Epoch:   0%|          | 0/106 [00:00<?, ?it/s]
{'no_components': 110, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.0013383091349196815, 'item_alpha': 0.0004011514698110847, 'max_sampled': 5, 'num_epochs': 106}
Epoch: 100%|██████████| 106/106 [01:20<00:00,  1.32it/s]
Epoch:   0%|          | 0/156 [00:00<?, ?it/s]
{'no_components': 113, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.001901610977672106, 'item_alpha': 3.940885086353212e-05, 'max_sampled': 9, 'num_epochs': 156}
Epoch: 100%|██████████| 156/156 [01:57<00:00,  1.33it/s]
Epoch:   0%|          | 0/152 [00:00<?, ?it/s]
{'no_components': 90, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.001011352043043931, 'item_alpha': 0.00021764597135462782, 'max_sampled': 7, 'num_epochs': 152}
Epoch: 100%|██████████| 152/152 [01:45<00:00,  1.44it/s]
Epoch:   0%|          | 0/138 [00:00<?, ?it/s]
{'no_components': 101, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.0006764943367687734, 'item_alpha': 0.000687673805676418, 'max_sampled': 5, 'num_epochs': 138}
Epoch: 100%|██████████| 138/138 [01:48<00:00,  1.28it/s]
Epoch:   0%|          | 0/89 [00:00<?, ?it/s]
{'no_components': 99, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.0005631327038821013, 'item_alpha': 0.001346835677680704, 'max_sampled': 7, 'num_epochs': 89}
Epoch: 100%|██████████| 89/89 [01:12<00:00,  1.23it/s]
Epoch:   0%|          | 0/91 [00:00<?, ?it/s]
{'no_components': 93, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.004796942703459156, 'item_alpha': 5.071235338644859e-05, 'max_sampled': 8, 'num_epochs': 91}
Epoch: 100%|██████████| 91/91 [00:54<00:00,  1.67it/s]
Epoch:   0%|          | 0/177 [00:00<?, ?it/s]
{'no_components': 102, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.002908021887591964, 'item_alpha': 0.00021783831225874806, 'max_sampled': 9, 'num_epochs': 177}
Epoch: 100%|██████████| 177/177 [01:59<00:00,  1.49it/s]
Best score 0.8844841718673706 at {'no_components': 93, 'learning_schedule': 'adagrad', 'loss': 'warp', 'learning_rate': 0.004796942703459156, 'item_alpha': 5.071235338644859e-05, 'max_sampled': 8, 'num_epochs': 91}
```

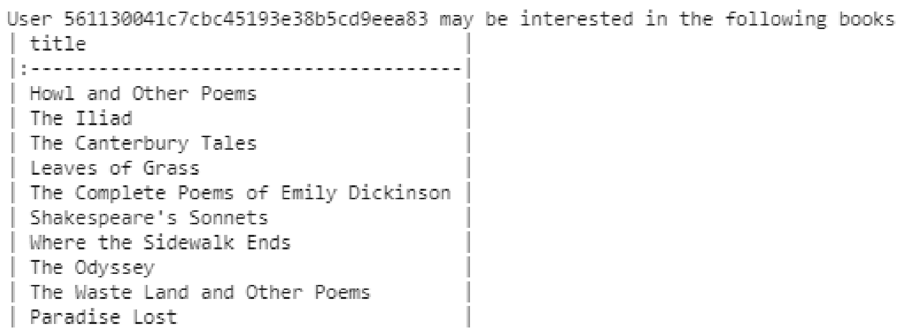
Figure 17: Parameter Tuning for LightFM model

This LightFM model can be saved using Python's Pickle module, to be deployed in a production environment and make predictions. LightFM also provides capability to add new item, user and interaction data using fit_partial() function that makes LightFM even easier to utilize in production and improve recommendation further.

3.4.3 Prediction

After the model is created, it can be used to make predictions. The predict() function will return the recommendation scores defined by the user inputs. In this study, a prediction will be made for a single user. As the predict function requires the inner IDs of user and items, the user and item mappings obtained with the Dataset object is utilized here.

As the output of the predict method does not differentiate between already read books, they are first filtered out. The prediction will then contain the books the given user will likely be interested in as depicted in Figure 18.



```
User 561130041c7cbc45193e38b5cd9eea83 may be interested in the following books
| title                                 |
|:--------------------------------------|
| Howl and Other Poems                  |
| The Iliad                             |
| The Canterbury Tales                  |
| Leaves of Grass                       |
| The Complete Poems of Emily Dickinson |
| Shakespeare's Sonnets                 |
| Where the Sidewalk Ends               |
| The Odyssey                           |
| The Waste Land and Other Poems        |
| Paradise Lost                         |
```

Figure 18: Top 10 recommendations for a user

CHAPTER 4
EVALUATION AND RESULTS

Recommender systems can be evaluated with the use of metric-based methods or human judgement methods. As the study is utilizing already available data, this proposed recommender system is evaluated on its accuracy using metric based methods.
LightFM library provides several methods that can be utilized to measure the performance of the created model. The following sections explains these further.

4.1 Area Under Curve (AUC)

ROC Curve or Receiver Operator Characteristic curve is a graph that plots the False Positive Rate against the True Positive Rate. The area under this curve is called the AUC score. In recommendation systems, AUC measures the quality of overall ranking. AUC can be interpreted as the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example. The closer AUC score is to 1, the more accurate the ordering given by the model is.

LightFM provides evaluation methods to calculate the AUC; the best score obtained in this study is 88.4% (0.884). This is higher than the AUC scores of both k-NN and SVD recommender systems that were used for comparison. Therefore, it can be said that the LightFM model with sentiment scores as features performs much better than pure collaborative filtering methods for this dataset.

4.2 Precision@k

Precision@k (precision at k) gives the faction of known positives in the first k positions of the ranked list of results. A perfect score is 1. LightFM precision_at_k function returns a numpy array containing precision@k scores for each user.

e.g.: [0. 0. 0. 0. 0.2 0. 0. 0.1 0.1 0.1 0. 0. 0. 0.1 0. 0.1 0. 0. 0.2 0.]

The mean of this array is taken as the precision@k value. If there are no interactions for a given user, then the returned precision will be 0.

This measure does not consider the overall ranking but only focuses on the ranking quality of the top k of the list. Therefore, this measure is highly dependent on the data. For example, if there is only one positive item for a particular user at k = 5, the maximum score for that user for precision@k will be 0.2. For the validation dataset used in this study; the precision@10 value ranged from 0.0 to 0.6 for each user. Out of 6451 users only 2362 users had precisions above 0, which indicates that the remaining 4089 did not have any interactions for precision to be measured.

For the LightFM model created in this study,

when k = 10

Precision@10 for the train dataset	=	0.28
Precision@10 for the test dataset	=	0.05
Precision@10 for the validation set	=	0.05

4.3 Recall@k

Recall at k is the number of positive items in the first k positions of the ranked list of recommended results divided by the number of positive items in the test period. In other words, recall@k is the proportion of relevant items found in the top k recommendations. A perfect score is 1.0. Similar to precision_at_k, LightFM's recall_at_k also returns a numpy array containing recall@k scores for each user. If there are no interactions for a given user having items in the test period, the returned recall will be 0.

For the LightFM model created in this study:

when k = 10

Recall@10 for the train dataset	=	0.10
Recall @10 for the test dataset	=	0.13
Recall @10 for the validation set	=	0.14

According to the Figures 19 and 20, Precision tends to be higher for smaller k values; highest value being 0.05 when k is 1. Recall value tends to increase with the k value going up to 0.28 when k is 50.

While both precision@k and recall@k value appear to be rather low, one reason for this maybe the sparsity of the data. While it was ensured that each user had rated at least 20 books in the initial dataset, when splitting this dataset into three separate sets (80%, 10%, 10%) using train_test_split() no special consideration was taken to ensure that each user had interactions in all three datasets. This assumption is further proven by the fact that when testing the model with 20% of the dataset (while 80% is set aside for training), the precision@k improved.

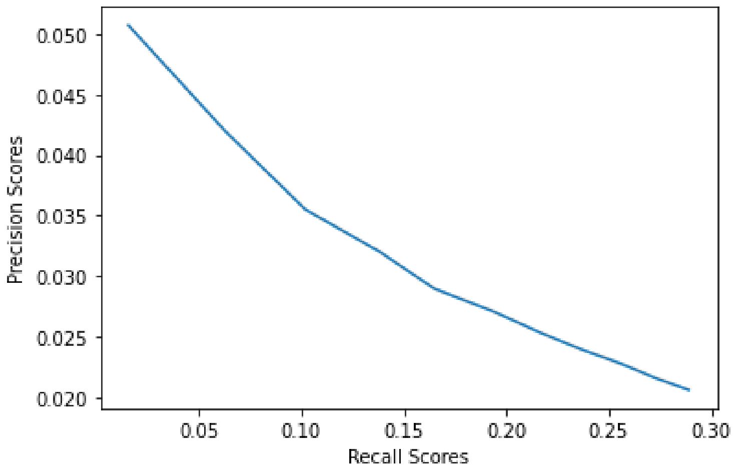
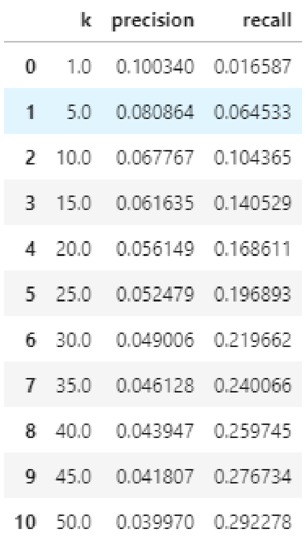


Figure 19: Precision@k and Recall@k values



	k	precision	recall
0	1.0	0.100340	0.016587
1	5.0	0.080864	0.064533
2	10.0	0.067767	0.104365
3	15.0	0.061635	0.140529
4	20.0	0.056149	0.168611
5	25.0	0.052479	0.196893
6	30.0	0.049006	0.219662
7	35.0	0.046128	0.240066
8	40.0	0.043947	0.259745
9	45.0	0.041807	0.276734
10	50.0	0.039970	0.292278

Figure 20: Precision@k and Recall@k values at different 'k' values for larger testing set

4.4 Accuracy of VADER

From the sentiment analysis performed with VADER, out of 110244 reviews, 8335 reviews which had a rating of 5 or 4 (which corresponds to “It was amazing”, “I really liked it”) were classified as “negative”. Number of reviews which had a rating of 1 (“I didn’t like it”) but classified as “positive” were 1017.
It may indicate that approximately 8.4% of the sentiment polarity were miscalculated. The accuracy of the sentiment polarity calculation can be deduced as 91.6%.
However, further in-depth analysis with human intervention is needed to verify or refute this, as it is possible that even with a positive rating, the reader may have described some negative aspects of the book in the review, or vice versa.

As elaborated in detail above, the results of this study indicate that interaction data combined with sentiment intensity scores of reviews could be used to improve recommendation process. The possible improvements to this study and conclusions on the results that are obtained will be discussed in the next chapter.

CHAPTER 5
CONCLUSION AND FUTURE WORK

Both recommendation algorithms and sentiment analysis techniques are used in abundance in various applications at present. The study presented in this report, attempts to combine the two approaches to obtain better recommendations for the user as a way of improving the services provided to the user. This chapter contains the conclusions arrived at, at the end of this study and the possible future work that may further improve and attest that sentiment analysis could in fact be of use in recommendations.

5.1 Conclusion

At the sentiment analysis phase of this study, VADER proved to have 91% accuracy which may provide us with reasonable confidence that the sentiment scores calculated for each review, and subsequently each book are in fact correct. In the initial publication on VADER (Gilbert & Hutto, 2014), authors claimed fairly good F1 scores for Tweets (Posts on Twitter), Movie and Product reviews and New York Times articles as well which is another indication for the reasonable accuracy of the sentiment analysis results.

With the results of the recommendation model presented in the preceding chapter, it may be rationally concluded that sentiment scores can be used as additional features in the recommendation process and that hybrid models do have an advantage over pure collaborative filtering methods provided by packages such as Surprise. The overall model and rankings provided by the LightFM model are better than that of pure collaborative filtering methods as proven by the Area Under the Curve (AUC) scores of both approaches.

However, the final model did not present high precision@k and recall@k scores for test and validation datasets. As mentioned in the previous chapter, while the initial dataset contained books and users that had at least 20 interactions per entity, no especial effort was given to ensure users had interactions in training and testing sets when splitting the dataset for testing and training. As the LightFM document itself claims, this may create a partial cold-start problem. While the item features in the form of sentiment scores may slightly alleviate this problem, this does not seem to be enough to provide higher precision and recall scores for top k items.

5.2 Future Work

One of the biggest challenges in Natural Language Processing thereby sentiment analysis is the constant evolution of natural languages. As languages evolves quickly with English language acquiring words from other languages and non-standard or unofficial forms of words (e.g.: Internet slang), lexicons will have to updated frequently. This is especially true of the latter as most reviewers, be it for products, movies, or books, tend to use Internet slang fairly often unlike professional critics.

Therefore, one obvious future enhancement to the approach discussed in this thesis is using more sophisticated sentiment analysis algorithms to calculate sentiment scores. This method would have to allow for constantly evolving languages, as well as other ubiquitous words, phrases that is used on the Internet. The latter may also include emojis, GIF (Graphics Interchange Format) and image reactions in reviews.

Another future enhancement for this study will be to include more features per book and measure the precision and recall values. Such features can be extracted from the book metadata available in the same source listed in Appendix A (UCSD Book Graph - (Mengtin, 2017)). Attributes such as Language Code, Authors, and Publication Year are a few examples for the possible features.

In this study only the overall sentiment was calculated. Further investigations in to whether Aspect Based sentiments could be utilized in this approach may also yield interesting results. This will require selecting the best aspects to be used in the LightFM item features as well.

To further verify the results presented in the study and improve the performance further, this approach will need to be tested for larger datasets as this study is only conducted for the poetry books. Utilizing more data, ideally for books of different genres, will solidify the findings of this study. The genre or genres of the books of the new dataset can also be utilized as another feature of books.

In this study only items features have been included in the LightFM models. As another improvement, it is also possible to add user features to further personalize recommendations.

REFERENCES

Alaei, A. R., Becken, S. & Stantic, B., 2017. Sentiment analysis in tourism: Capitalising on Big Data. *Journal of Travel Research,* 58(2), p. 175–191.

Amazon.com, 2007. *Goodreads.* [Online]
Available at: https://www.goodreads.com
[Accessed 20201].

Bansal, S., 2019. A Study of Recent Recommender System Techniques. *International Journal of Knowledge and Systems Science,* 10(2), pp. 13-41.

Burke, R., 2007. Hybrid web recommender systems. In: P. Brusilovsky, A. Kobsa & W. Nejdl, eds. *The Adaptive Web. Lecture Notes in Computer Science.* s.l.:Springer, Berlin, Heidelberg, pp. 377-408.

Danlik, M., 2021. *langdetect 1.0.9.* [Online]
Available at: https://pypi.org/project/langdetect
[Accessed 2021].

Gilbert, E. & Hutto, C. J., 2014. *VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text..* s.l., The AAAI Press.

Hamilton, W. L., Clark , K. & Lesko, J., 2016. Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora.

Hug, N., n.d. *Surprise - A Python scikit for recommender systems.* [Online]
Available at: http://surpriselib.com/
[Accessed 2021].

Kula, M., 2015. Metadata Embeddings for User and Item Cold-start Recommendations. *CoRR,* Volume abs/1507.08439.

Liu, B. & Zhang, L., 2012. A Survey of Opinion Mining and Sentiment Analysis. In: *Mining Text Data.* s.l.:s.n., pp. 415-463.

McAuley, J. & He, R., 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *WWW.*

Mengtin, W., 2017. *Goodreads Datasets.* [Online]
Available at: https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home
[Accessed 2020].

Musto, C., de Gemmis, M., Semeraro, G. & Lops, P., 2017. *A Multi-criteria Recommender System Exploiting Aspect-based Sentiment Analysis of Users' Reviews.* Como Italy, s.n.

Musto, C., Lops, P., de Gemmis, M. & Semeraro, G., 2019. *Justifying Recommendations through Aspect-based Sentiment Analysis of Users Reviews.* Larnaca Cyprus, s.n.

Nakatani, S., 2010. *Language Detection Library for Java.* s.l.:s.n.

Neethu, M. S. & Rajasree, R., 2013. *Sentiment analysis in twitter using machine learning techniques.* Tiruchengode, India, s.n., pp. 1-5.

Osman, N. A., Noah, S. A. M. & Darwich, M., 2019. Contextual Sentiment Based Recommender System to Provide Recommendation in the Electronic Products Domain. *International Journal of Machine Learning and Computing,* 9(4).

Puschmann, C. & Powell, A., 2018. Turning Words Into Consumer Preferences: How Sentiment Analysis Is Framed in Research and the News Media. *Social Media + Society.*

Ricci, F., Rokach, L. & Shapira, B., 2011. *Recommender Systems Handbook.* s.l.:Springer New York Dordrecht Heidelberg London.

Rokade, P. P. & Kumari D., A., 2019. Business intelligence analytics using sentiment analysis-a survey. *International Journal of Electrical and Computer Engineering (IJECE),* 9(1), pp. 613-620.

Sachdeva, N. & McAuley, J., 2020. *How Useful are Reviews for Recommendation? A Critical Review and Potential Improvements.* Virtual Event, China, ACM SIGIR.

Schafer, J. B., Frankowski, D., Herlocker, J. & Sen, S., 2007. *Collaborative Filtering Recommender Systems.* Springer-Verlag, Berlin Heidelberg.

Singh, V. K., Mukherjee, M. & Kumar , G., 2011. *Combining a Content Filtering Heuristic and Sentiment Analysis for Movie Recommendations.* Bangalore, Springer, Berlin, Heidelberg.

Wan, M. & McAuley, J., 2018. *Item Recommendation on Monotonic Behavior Chains.* Vancouver, Association for Computing Machinery, p. 86–94.

APPENDIX A: DATASETS

Poetry

Download Links:

- goodreads_books_poetry.json.gz (*36,514* books)
- goodreads_interactions_poetry.json.gz (*2,734,350* interactions)
- goodreads_reviews_poetry.json.gz (154,555 detailed reviews)

Books

Download Links:

Complete book graph: goodreads_books.json.gz

Author Information: goodreads_book_authors.json.gz

Work Information: goodreads_book_works.json.gz

Book Series: goodreads_book_series.json.gz

Fuzzy Book Genres: gooreads_book_genres_initial.json.gz

Shelves

Download Links:

- Complete ***229m**** interactions in 'csv' format (~4.1g): goodreads_interactions.csv
- User IDs: user_id_map.csv
- Book IDs: book_id_map.csv

Reviews

- Complete 15.7m reviews (~5g): goodread_reviews_dedup.json.gz
- Review subset (~1.38m reviews) with parsed spoiler tags: goodreads_reviews_spoiler.json.gz
- Spoiler subset with original review text: goodreads_reviews_spoiler_raw.json.gz

APPENDIX B: SOURCE CODE

Identifying English Reviews with Langdetect

```
from langdetect import detect, DetectorFactory

df = pd.read_json('../input/poetry-reviews/goodreads_reviews_poetry.json', lines=True)

for index, row in df.iterrows():
    try:
        if (detect(str(row['review_text'])[0:200]) != 'en'):
            df.drop(index, inplace=True)
    except Exception:
        df.drop(index, inplace=True)
df.to_csv('English_reviews.csv',index=False)
```

Sentiment Analysis with VADER

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import pandas as pd

df["compound"] = 0
df["neg"] = 0
df["neu"] = 0
df["pos"] = 0

sid = SentimentIntensityAnalyzer()

for index, row in df.iterrows():
    try:
        scores = sid.polarity_scores(row['review_text'])
        df.loc[index,'compound'] = scores["compound"]
        df.loc[index,'neg'] = scores["neg"]
        df.loc[index,'neu'] = scores["neu"]
        df.loc[index,'pos'] = scores["pos"]
    except Exception:
         pass

df.to_csv("sentiment_scores.csv", index=False)
```

Data preparation for LightFM

```
class DataPrep:
    def __init__(self):
        pass

    def generate_feature_list(self, dataframe, features_name):

        """
        Generate features list for mapping

        Parameters
        ----------
            dataframe: Dataframe
                Pandas Dataframe for Books.
            features_name : List
                List of feature columns name avaiable in dataframe.

        Returns
        -------
            List of all features for mapping
        """
        features = dataframe[features_name].apply(lambda x: ','.join(x.map(str)), axis=1)
        features = features.str.split(',')
        features = features.apply(pd.Series).stack().reset_index(drop=True)

        return features

    def create_features(self, dataframe, features_name, id_col_name):
        """
        Generate features that will be ready for feeding into lightfm

        Parameters
        ----------
            dataframe: Dataframe
                Pandas Dataframe which contains features
            features_name : List
                List of feature columns name avaiable in dataframe
            id_col_name: String
                Column name which contains id of the item e.g.: "book_id"

        Returns
        -------
            Pandas Series
                A pandas series containing process features
                that are ready for feed into lightfm.
                The format of each value
                will be (user_id, ['feature_1', 'feature_2', 'feature_3'])
        """

        features = dataframe[features_name].apply(lambda x: ','.join(x.map(str)), axis=1)
        features = features.str.split(',')
        features = list(zip(dataframe[id_col_name], features))
        return features


    def get_all_feature_data(self, metadata, interactions ,features_name, id_col_name, ):

        """
        Generate lighfm data for trainig the model

        Parameters
        ----------
            metadata: Dataframe
                Pandas Dataframe containing item features
            interactions: Dataframe
                Pandas Dataframe containing user-item interactions
            features_name : List
```

```
                List of feature columns name avaiable in dataframe.
            id_col_name: String
                The item_id column name
        Returns
        -------
            List of all features for mapping
        """
        book_sentiments = metadata[metadata.book_id.isin(interactions.book_id)]

        features_list = self.generate_feature_list(book_sentiments, features_name)
        book_features = self.create_features(book_sentiments, features_name, id_col_name)


        dataset = Dataset(user_identity_features=False)
        dataset.fit(interactions['user_id'].unique(),
                    book_sentiments['book_id'].unique(),
                    item_features=features_list)

        lightfm_item_features = dataset.build_item_features(book_features)

        interactions = list(zip(interactions.user_id,
                                interactions.book_id,
                                interactions.rating))

        lightfm_interactions, lightfm_weights = dataset.build_interactions(interactions)

        user_id_mapping = pd.DataFrame(list(dataset.mapping()[0].items()),
columns=['user_id', 'inner_uid'])
        item_id_mapping = pd.DataFrame(list(dataset.mapping()[2].items()),
columns=['book_id', 'inner_iid'])


        return lightfm_item_features, lightfm_interactions, lightfm_weights,
user_id_mapping, item_id_mapping
```

Hyperparameters

```
def sample_hyperparameters():
    """
    possible hyperparameter choices.
    """
    while True:
        yield {
            "no_components": np.random.randint(80, 120),
            "learning_schedule": np.random.choice(["adagrad"]),
            "loss": np.random.choice(["warp"]),
            "learning_rate": np.random.exponential(0.001),
            "item_alpha": np.random.exponential(0.0005),
            "max_sampled": np.random.randint(5, 10),
            "num_epochs": np.random.randint(10,200)
        }

def random_search(train, test, item_features, weights, num_threads, num_samples):
    """
        Create a lighfm model given the hyperparameters, evaluates AUC score and returns
the
        best score, model and hyperparameters

        Parameters
        ----------
            train: output of a train_test_split() method or a coo_matrix
            test: output of a train_test_split() method or a coo_matrix
            item_features : csr_matrix containing item_features
            weights: weights matrix
            num_threads: number
            num_samples: number
        Returns
        -------
            best score, model and hyperparameters
    """
    for hyperparams in itertools.islice(sample_hyperparameters(), num_samples):
        print(hyperparams )
        num_epochs = hyperparams.pop("num_epochs")

        model = LightFM(**hyperparams)
        model.fit(train,
                  item_features=item_features, sample_weight=weights,
                  epochs=num_epochs, num_threads=num_threads, verbose=True)
        score = auc_score(model, test, train_interactions=train, num_threads=num_threads,
item_features=item_features).mean()
        hyperparams["num_epochs"] = num_epochs

        yield (score, hyperparams, model)
```

Create and Train Model

```
model = LightFM(no_components=93,
                learning_schedule= 'adagrad',
                loss='warp',
                learning_rate=0.004797,
                item_alpha=5.071235338644859e-05,
                max_sampled=8)
model.fit(train,
          item_features=lightfm_item_features, sample_weight=train_weight,
          epochs=91, num_threads=2, verbose=True)
```

Making Recommendations

```
class MakeRecommendations:
    """
    Make prediction given model and user ids
    """
    def __init__(self, lightfm_model,
                 books,
                 item_features,
                 interactions,
                 user_id_mapping,
                 item_id_mapping):
        self.model = lightfm_model
        self.books = books
        self.item_features = item_features
        self.interactions = interactions
        self.user_id_map = user_id_mapping
        self.item_id_map = item_id_mapping

    def _filter_already_read_books(self, user_id):
        """Drop books already read(rated) by the user_id"""

        read_book_ids = self.interactions.loc[self.interactions['user_id'] ==
user_id,'book_id']
        books_for_prediction =
self.item_id_map.loc[~self.item_id_map['book_id'].isin(read_book_ids.tolist())]

        return books_for_prediction

    def make_recommendations_per_user(self, user_ids, num_prediction=5, num_threads=1):


        inner_uid = self.user_id_map[(self.user_id_map['user_id'] ==
user_ids)].inner_uid.values[0]
```

```
        #get books already not read by the user.
        books_for_prediction = self._filter_already_read_books(user_ids)

        score = self.model.predict(
                            int(inner_uid),
                            books_for_prediction['inner_iid'].values.tolist(),
                            item_features=self.item_features)

        books_for_prediction = books_for_prediction.copy()
        books_for_prediction['recommendation_score'] = score
        books_for_prediction = books_for_prediction.sort_values( by='recommendation_score',
ascending=False)[:num_prediction]

        print("User {} may be interested in the following books".format(user_ids))
        books_for_prediction =
self.books.loc[self.books['book_id'].isin(books_for_prediction['book_id'])][['title']]
        print(books_for_prediction.to_markdown(index=False))

        return books_for_prediction
```