Using a Hybrid Recommender to Mitigate Filter Bubble and Item Cold Start Problem in Movie Recommendations

Dawit Nerea

Department of Data Science, The University of Wisconsin – Eu Claire

DS 785: Capstone

Dr. Tracy Bibelnieks

May 1, 2022

Abstract

Recommendation engines are readily used to provide online customers with relevant recommendations. Among the different types of recommendation systems used by major online marketers are content-based recommenders and collaborative-filtering models. Both models are among the most well-known traditional recommendation models. Having said that, these approaches do not come without drawbacks. Specifically, content-based recommendations are known to give users item recommendations that resemble their past consumption history with limited diversification which is also known as a filter bubble. On the other hand, because collaborative-filtering models depend on ratings provided by users, these models are ineffective when applied to datasets with a minimal user to item interactions or more commonly called a cold start problem. This case study demonstrates how to use a hybridized model of the two models mentioned above to mitigate their shortcomings. The results from the comparison of all three models for this paper show that a hybridized model outperforms a content-based recommender by giving more diverse recommendations as measured by diversity as a metric. Similarly, results show that items that lack user interaction can be included in the set of recommendations using a hybridized model compared to a collaborative-filtering model as measured by coverage of total recommendations for the two models. This paper first covers the introduction to all the models, objectives, and the motivation for the project followed by a literature review. Following that, the methodology of building all three models, results of the analysis, implications, and future recommendations are covered in detail.

Keywords*:* content-based recommender, collaborative-filtering model, hybrid model, filter bubble, cold start problem, diversity, coverage.

Contents

Abstract 2
List of Tables 5
List of Figures 6
Chapter 1: Introduction 7
Overview 7
Problem Statement 8
Statement of the Purpose 9
Project Objectives 9
Significance of the Study 10
Limitations 11
Conclusion 11
Chapter 2: Literature review 12
A brief introduction to recommendation systems 12
Limitations of current recommendation applications 12
Overcoming item based cold start using content-based recommendation systems 13
Overcoming filter bubbles using collaborative filtering 15
Importance of product diversification 16
Using a hybridized approach 18
Conclusion 20
Chapter 3: Methodology 21
Research design 21
Data set overview 22
Data processing 22
TD-IDF (Term Frequency-Inverse Document Frequency) 23
Cosine similarity 27
Content-based recommender 30
User profile 31
Content Analysis 31
Retrieving Recommendations 32
Measuring the performance of a content-based recommender 32
Collaborative filtering model 34
Memory-based collaborative filtering 34

Model-based collaborative filtering 35
K nearest neighboring (KNN) 35
KNN implementation using the surprise package for python 38
Hyperparameter tuning using a grid search 39
Using coverage as a metric to evaluate collaborative filtering 40
Building a hybrid model 41
Sequential steps in the comparison of the three models 42
Steps in comparing models on diversity Vs coverage 43
Conclusion 45
Chapter 4: Findings and results 46
Introduction 46
Findings 47
Comparing a Content-based and the collaborative-filtering models 47
Comparing the hybrid model Vs content-based recommender 50
Comparing the hybrid model with the collaborative-filtering model 51
Assessing diversity in varying levels of data sparsity 53
Conclusion 55
Chapter 5: Summary, Recommendations, Conclusion 57
Introduction 57
Summary of findings 57
Improving diversity 58
Combating item-based cold start problem 58
Recommendations 58
Textual analysis from multiple sources 59
Simulate the ratio of content-based and collaborative filtering recommendations 59
Resolving user-based cold start along with item-based cold start 60
Using deep learning instead of a content-based recommender 61
Using metrics such as serendipity for comparison 62
Devise a method to gather the users' experience 62
Conclusion 62
References 64
Appendix 69

List of Tables

Table 1 *TD-IDF scores* 26

Table 2 *Cosine similarity for a sample of movies in the movie's dataset* 30

Table 3 *Coverage values for hybrid and collaborative-filtering models* 52

Table 4 *Diversity values for the hybrid model and content-based recommender* 54

List of Figures

Figure 1 *Content-Based recommendation* 14

Figure 2 *Item-based collaborative filtering recommendation design* 15

Figure 3 *Rate of recommendation for a total of N recommendations* 17

Figure 4 *Heatmap for TF-IDF scores of words in a sample of movies* 27

Figure 5 *Visual representation for cosine similarity of 4 sample vectors* 28

Figure 6 *Item-based recommendation summary* 37

Figure 7 *Hybridized model design* 42

Figure 8 *Mean similarity score distribution plot* 48

Figure 9 *Popularity density plot for content-based recommendations* 49

Figure 10 *Popularity density plot for collaborative-filtering based recommendations* 49

Figure 11 *Mean similarity score distribution for Hybrid and content-based recommenders* 51

Figure 12 *Percentage difference in coverage between hybrid and collaborative models* 53

Figure 13 *Hybrid model Vs content-based recommender diversity scores* 55

Chapter 1: Introduction

Overview

Product recommendation is at the heart of good customer experience for companies such as Netflix, Amazon, and many others who thrive on providing the best customer experience (Rocca, 2021). Customer satisfaction is one of the core principles of all business models and companies implement various strategies to maintain a high standard of customer service on which the success of the company depends (Isinkaye et al., 2015). With the explosion of technology, customers now more than ever rely on their smart devices to interact with different services ranging from entertainment to online retail companies to satisfy their respective needs.

The accelerated use of the internet calls for the implementation of big data methods such as recommendation engines to gain deeper insights into the trends and traits of behaviors customers exhibit during their online activities. Therefore, recommendation systems have become a focal point for most businesses as they have the power to galvanize the resources available by companies and convert them into meaningful ways upon which customers can interact with the companies and the items they provide. Recommendation systems are a wide topic, and the research around different approaches continues to expand.

In this chapter, the overall objective of the case study will be discussed along with the relevance of the study when applied in the real world. The key points discussed in this chapter are outlined below.

- **Part 1** will cover the main obstacles facing recommendation systems.
- **Part 2** will cover the main purpose behind using a hybridized model to mitigate the obstacles mentioned in part 1.

- **Part 3** will outline the objective of the case study and why content-based and collaborative-filtering models will be hybridized for this study.
- **Part 4** will detail the relevance of the study for both companies and customers who use an online platform.
- **Part 5** will mention some possible limitations of this case study.

Problem Statement

Recommendation systems come in different formats depending on what item is being recommended and which platform the algorithms are being implemented on. Almost all approaches to recommendation systems come with advantages and disadvantages where one approach tries to resolve the downfalls of another approach. For this reason, it is important to identify which approach is most reasonable for a given company whereby the disadvantages of the method chosen will not significantly impact the productivity of the said company.

Items online come with different variables, both quantitative and qualitative, that enable recommendation systems to make statistical computations on. Having said that, no dataset comes with a complete set of variables to enable recommendation algorithms to make perfect predictions as to which items to recommend for users. The lack of data significantly impacts recommendation systems that rely on ratings and reviews to make accurate predictions. This is more generally known as *data sparsity* (Chen, 2021). This is especially true for new items or users that lack enough historical data such as ratings or preferences upon which future recommendations can be made on. In technical terms, this is known as the *cold start problem*. The cold start problem can be user-based or item-based depending on which historical data is missing from the dataset (Grčar et al., 2006).

Another challenge seen in the field of recommendation systems is what is known as a *filter bubble* whereby recommendation engines give highly specialized recommendations to users (Le, 2018). This is not only detrimental to the company as their ability to expose more products is limited, but it also limits users to consume similar products and thereby not allowing them to broaden their tase by exploring other options.

Statement of the Purpose

Given that there are numerous types of recommendation systems, it is advisable to look for ways to combine different algorithms to offset some of the shortcomings mentioned above. This case study aims to do this by combining two types of recommendation systems, namely a content-based recommender, and a collaborative filtering recommender. Such an approach is called a *hybrid* approach whereby the strengths of the two algorithms will assist in giving more appropriate recommendations to users.

Project Objectives

To see how hybridization works, this case study will focus on two recommendation systems, *content-based* and *collaborative filtering recommendations*. The reason why the two recommendation systems were selected is they complement each other's advantages and disadvantages (Berkay, 2022). This makes them a perfect candidate to build a hybridized algorithm based on the two models. Content-based recommenders solely depend on qualitative data such as item description and therefore algorithms that depend on this approach suffer from a filter bubble scenario whereby users are limited to consuming items with similar descriptions. On the other hand, recommenders that solely depend on collaborative filters depend on mainly quantitative data such as ratings to give recommendations. This in turn makes collaborative

filtering recommenders suffer in scenarios where the dataset does not have enough numerical historical data about either users or items (Gupta and Dave, 2019).

The hybridized model will help in dealing with the data sparsity and filter bubble scenarios mentioned above by inheriting the strengths of the two independent approaches. Specifically, the content-based recommender will have the ability to prevent item-based cold start problems by relying solely on attributes of items irrespective of when they were introduced to the algorithm. On the other hand, collaborative filters enable users to explore wider options due to their ability to galvanize the rating trends of similar users or items that share similar sentiments from users. Therefore, the main goal of this case study is to go in-depth regarding the two fundamental problems, data sparsity, and filter bubbles, and how the hybridized approach can help mitigate them. The summarized objectives of the case study will be as follows.

- Build a content base recommendation system to mitigate item-based cold start problems.
- Build a collaborative filter recommendation system to mitigate the filter bubble to give wider options in recommendations for users.
- Hybridize both models together to mitigate both item-based cold start problems and filter bubbles.

Significance of the Study

Given that recommendation systems are an actively growing field of study, this study will provide a fundamental understanding of the science and on which aspects of the science there remains a need for improvement. Furthermore, a breakdown of each approach mentioned above will reveal technical details that can spark new insights on how to improve existing recommendation systems. Hybridized models are a perfect representation of how two algorithms can be merged to drive better performance and understanding how to build them enables one to

gain a detailed understanding of the two recommendation system approaches that constitute them.

Furthermore, this case study will enable readers to identify which datasets require which types of recommendation systems. For instance, not all datasets require hybridized approach if for example abundant historical data is available and models with just a collaborative approach could suffice. Such insight requires the intimate knowledge of both how the models operate, and which shortcomings could raise a problem depending on a given data set.

Limitations

This case study will be using a hybridized model derived from collaborative filtering and content-based recommenders. It is only possible to partially deal with a cold start problem using this approach meaning it is only possible to mitigate an item-based cold start problem. This means the cold start problem cannot be completely avoided in cases where we have new users introduced to the algorithm. Furthermore, all models built using the approach proposed above will rely on linear computation to draw similarities between items and between users. Therefore, when applied to a significant amount of data, the computation required might be very expensive.

Conclusion

In this chapter, the overall problem statement is discussed by highlighting the shortcomings of the pure models. The purpose and objectives for this case study are discussed within the context of the problem statement on how a hybridized model will help mitigate item-based cold start problems and filter bubbles. Furthermore, the significance of the study and possible limitations are also highlighted. A literature review regarding the topic will follow in chapter 2.

Chapter 2: Literature review

A brief introduction to recommendation systems

The field of recommendation systems has become a major topic within the field of data science due to a massive explosion of online activities. The number of companies that utilize the internet has increased dramatically and the need to have efficient interaction with customers has become an integral part of companies' business models. Major companies such as Netflix have reaped the benefits of investing in powerful recommendation engines that enable them to recommend items that satisfy the taste of their customers at a higher rate.

The most widely used recommendation systems, according to Banik (2018), from which other engines are derived are knowledge-based RS, content-based RS, and collaborative filtering RS. Researchers are also making progress on recommendation systems based on deep learning in combination with traditional methods to build more sophisticated engines. This chapter will discuss some of the common limitations of individual models, proposed techniques by researchers to overcome these limitations, and how a hybridized approach has been used in the past and compared with individual models.

Limitations of current recommendation applications

Recommendation engines are interactive by nature and rely on user behavior to gain insight for future recommendations. This means, that in instances where the engine lacks abundant data on either users' or items' past performance such as ratings or reviews, the recommendations provided will be inaccurate. The lack of sufficient data in the field of recommendation systems is known as *data sparsity*. Data sparsity is common for platforms that add new items regularly such as Netflix and Amazon. Researchers have devised numerous ways to overcome the impact of data sparsity which in most cases requires gathering valuable

variables from item descriptions and comparing them to historical user behavioral trends. According to Isinkaye et al (2015), data sparsity is one of the major difficulties for recommendation systems due to the exponential growth of both internet users and the addition of new items to the internet. Furthermore, researchers are working on ways to broaden the recommendations given to users so that users are allowed to try items they would not normally go for or in other words to overcome *filter bubbles*.

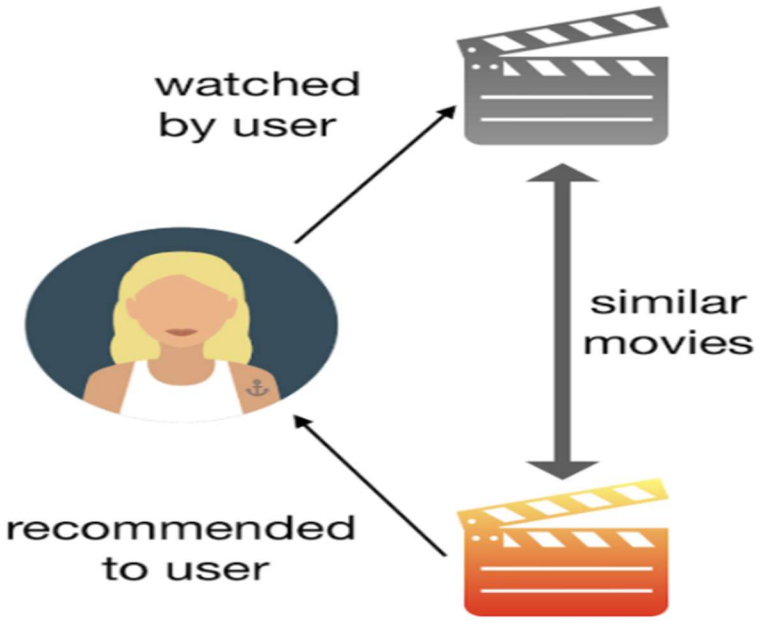
Overcoming item based cold start using content-based recommendation systems

As mentioned in chapter 1, an example of a data sparsity scenario happens when a given dataset lacks enough information such as ratings on the items it contains. This is more specifically known as an *item cold-start problem*. Having said that, most new items introduced to a website have descriptive data such as overview and metadata that can be used as an alternative variable by recommendation engines.

According to Kula (2015), the approach above is called content-based recommendation because it relies on the contents in the description of an item which are given in a textual format. The author demonstrates that a content-based approach relies on past data from users that enables the engine to formulate similarities and differences with new items based on attributes such as descriptions of an item when reviews and ratings are not available. As shown in Figure 1, content-based recommenders use a feedback loop using past consumption behavior and item description with no rating data to make future recommendations (Grimaldi, 2019). Having said that, in the absence of historical data from users, this approach cannot be implemented on its own showing the reason why a content-based approach can only deal with one-half of the cold start problem.

Figure 1

Content-Based recommendation



Note: Reprinted from Emma Grimaldi. (2018) How to build a content-based movie recommender system with Natural Language Processing. https://tinyurl.com/2fzdk9sm

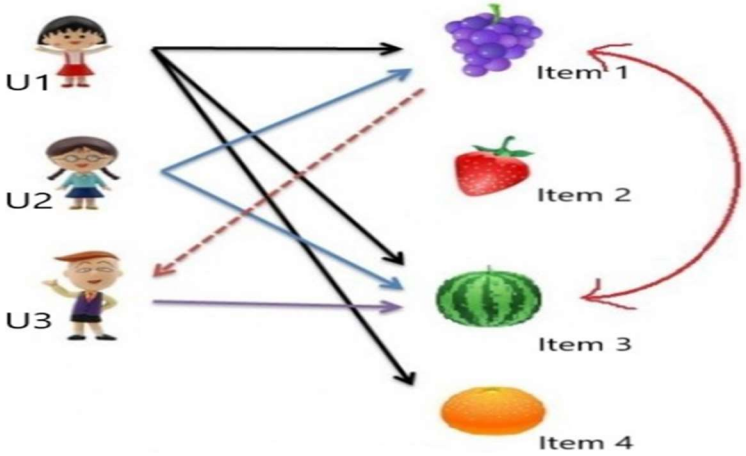
To draw similarities between what users have already consumed and new items added to websites, content-based recommendation engines must find a way to compare these items independent of ratings and reviews. According to Banik (2018), one of the most effective ways of drawing such a comparison when it comes to recommending movies to users is to use the textual description of items such as movies' overviews, genre, and actors involved in the movies, and so on. This requires the transformation of textual data to numerical data from which mathematical similarities can be computed upon. The author also demonstrates the use of TF-IDF and cosine similarity as methods that can be used to achieve this. These methods will be discussed in detail in chapter 3 when building a purely content-based recommender before hybridization with a collaborative-filtering model.

Overcoming filter bubbles using collaborative filtering

In content-based recommendation systems, there is no networking available to see how other users are interacting with items online. An over-reliance on content-based recommenders can lead to customer fatigue because of no surprising recommendations being included to users. On the other hand, collaborative filtering allows for many like-minded users to influence one another in their consumption patterns as shown in Figure 2. According to Pinela (2018) and as shown in Figure 2, *item-3* and *item-1* as similar items due to how customers have interacted with them in the past. Based on this similarity, for a subset of customers with overlapping purchase history that has liked both items, the remaining customers within that group will then be recommended the item they have not consumed yet which in this case is accomplished by recommending item 3 to the user 3. This opens the door for users to identify items they would not have otherwise discovered. Therefore, collaborative filtering can be seen as one possible way of broadening the diversity of recommendations to help mitigate a filter bubble situation.

Figure 2

Item-based collaborative filtering recommendation design



Note: Reprinted from Carlos Pinela. (2017) Recommender Systems, User-Based and Item-Based Collaborative Filtering. https://blog.clerk.io/collaborative-filtering

Research by Zhang (2012), who conducted a study to see how different sets of recommendations affect users' behavior demonstrates the value of an element of surprise in recommendations made to users. The study uses an advanced application of collaborative filtering while using *Novelty* and *Serendipity* as metrics of measurement for recommending items outside the usual recommendations for users. It demonstrates how to overcome "safe" recommendations and gives customers a chance to explore different options than their usual consumption patterns. Users were able to rate their level of satisfaction for two different models, one full auralist model, and a basic auralist model. The full model has high novelty and serendipity scores compared to the basic model. Upon evaluation of the accuracy of the two models, the basic model outperforms the full model when making predictions about how users would rate a song. Nevertheless, when users were made to rate their level of satisfaction between the model, the majority picked the full model. This is a great example to demonstrate that, when it comes to recommendations accuracy is not always indicative of customers' satisfaction and that models that explore more options can yield better results.

Banik (2018) illustrates that collaborative filters can be categorized under two major recommendation engine branches namely, *memory-based*, and *model-based* collaborative filters. An introduction to a memory-based approach and an application of model-based collaborative filtering will be used for this case study and will be discussed in detail in chapter 3.

Importance of product diversification

In the section above, diversification is explained as it relates to customer fatigue. Another reason for the diversification of product recommendations besides customer satisfaction is for the sake of diversifying sales for companies. The diversification of sales allows companies to

reinvent themselves with new products and increase their revenue by not depending on a few items at a time for their success.

Research from Yin et al. (2012) illustrates the concept of a long tail phenomenon in recommendation systems. The author states the danger of relying on recommendation systems that are going to keep rating the popular items highly while avoiding recommending unpopular items and the impact of such an approach on *sales diversity.* To understand this visually, Yasar (2018) illustrates the rate of recommendations for a sequence of recommendations made for users as shown in Figure 3. As seen in the figure, items on the left side of the plot are recommended more often for a given recommendation list than items in the tail region of the plot. The most frequently recommended items can be seen as the most popular out of the total recommendations that can be made for a given user. The author emphasizes the importance of optimizing the threshold and the long tail of the recommendations by increasing the diversification of recommended items.

Figure 3

Rate of recommendation for a total of N recommendations

Note: Reprinted from Kadir Yasar. (2018) Recommender Systems: What Long-Tail tells? https://medium.com/@kyasar.mail/recommender-systems-what-long-tail-tells-91680f10a5b2

Most companies aim to utilize the internet for their success along with the quality of products they manufacture using recommendation systems. If the proper strategy is not followed on how to give all items of the company exposure to customers, the resources spent on product diversification will be a liability limiting a company's potential to expand its horizon.

Product diversification in recommendations is not a blind approach whereby users are exposed to as many products as possible without any rhyme or reason. Recommending items that are too diverse can lead to disengagement from customers. A publication from Ge et al (2010) states that trust is a fundamental element of customers' engagement with a given platform based on recommendations that are made for them. The authors also state that recommending items without adequate analysis of what customers like and purely aimed toward product diversification can ultimately lead to distrust of customers towards recommendation engines.

To combat recommendations that may increase diversity but might not yield a good outcome when it comes to the trust of customers, this project will base all recommendations on either users' past usage history or recommendations based on like-minded users for a given user. In other words, all recommendations based on this approach will yield results that balance results from a specific niche of recommendations for a user and suggestions from other users that have similar patterns to introduce diversity without overlooking the trust and engagement of users.

Using a hybridized approach

Hybridized recommendations are algorithms that utilize two or more recommendation systems in collaboration. In most hybrid approaches, content-based, and collaborative filters are used in conjunction to combat the disadvantages of both approaches. Although computationally more expensive than their component, hybridized models have proven to be very reliable to make better recommendations by helping us overcome the limitations mentioned above

simultaneously. This project will focus on the use of a hybridized model to combat data sparsity and introduce product diversification in the list of recommendations made for users.

Even if this project won't be focused on the accuracy of predictions made by hybrid models, it is still important to highlight that hybrid models have been proven to make accurate rating predictions. This is shown in research done by Lee (2004), who demonstrated the effectiveness of the hybridized approach using mean absolute error (MAE) and rank scoring (RSM). The study was conducted on a dataset containing 7,291 users and 1,628 movies. The accuracy in the prediction of ratings by different models including a hybridized model was evaluated. The performance in measurement was done for a varying number of users and showed that hybridized models outperformed individual models for any given user. Most importantly, the growing number of users exponentially increased the accuracy of the hybridized approach compared to other models in the study. This finding is relevant given the exponential increase of online customers across the globe. In another study by Claypool (1999), hybridized models were measured against traditional recommendation systems on a fixed number of users but across a span of 20 days. The study used inaccuracy as a metric for the recommendation of newspapers. As the number of days increased the performance of the hybridized model was found to be significantly better than the remaining models.

Related to this project as far as mitigating data sparsity is a study from Pandey (2021). The study conducts a comparative approach to measure the performance of a purely collaborative-filtering model with a hybridized model. At a 99.28% data sparsity, the hybrid model shows a lower RMSE of 0.11 compared to the purely collaborative-filtering model with an RMSE value of 1.07. Having said that, the author makes the argument that, due to the nature of a content-based recommender that relies on item attributes instead of any rating data, the

hybrid recommender remains unaffected by the data sparsity value. This is because the recommendations made using this approach first rely on content-based recommendations and then the output of this method is processed using collaborative filtering. Such an approach might be effective for dealing with data sparsity but will not be ideal when trying to mitigate filter bubbles, but the research is still relevant in showing how hybridized models can be arranged to aim at and tackle a specific problem. For this project, the focus will be to tackle both data sparsity and filter bubbles. For this reason, the recommendations made from the two individual models will be combined in a way that will optimize the two goals of the project without one model preceding the other in the recommendation process.

All the studies above are relevant in highlighting the importance of hybridized models for the increasing number of online users, the adaptability of the model when being used for an extended period, and tackling a specific problem such as data sparsity. This shows that hybridized models are used to achieve a specific objective by tailoring the model-building process to achieve those goals. A similar approach will be used in this case study to achieve the project objectives.

Conclusion

So far in this chapter, the complementary nature of a content-based recommender and a collaborative-filtering model is shown to illustrate why they are ideal candidates for hybridization. This is true given the project objectives to mitigate item-based cold start problems and filter bubbles. Studies that highlight the significance of a hybridized approach compared to the pure models are discussed in detail. In the following chapters, an application of each model to overcome the limitations discussed in this chapter will be discussed. Methodology for developing the recommendation engines will be given in chapter 3.

Chapter 3: Methodology

Research design

In this chapter, the details of building a hybridized model using content-based and collaborative filtering techniques are discussed in detail. Below is a summary of the main steps involved in the methodology of building a hybridized model and a framework of how this chapter is structured.

- **Part 1** will cover the methods involved in the data processing step of the project including the transformation of data for model use. The use of TD-IDF and cosine similarity will be introduced in conjunction with how these methods are applied in content-based recommendation systems.
- **Part 2** will cover the methodology behind building a content-based recommender using users' profiles and processed data from step 1. Following this, ***diversity*** will be discussed as a measure of how well a model can give recommendations that are dissimilar and will be used to measure the performance of the content-based recommender.
- **Part 3** will cover the fundamental concepts behind a collaborative filtering model such as the K nearest neighbor (KNN) algorithm and its applications. The methodology of building a KNN-based collaborative recommender will be illustrated. Furthermore, ***coverage*** will be discussed as a metric of evaluation for the collaborative model and how it was applied to the movie dataset.

- **Part 4** will discuss the methodology of building a hybrid model by combining the models from parts 2 and 3.

Data set overview

This project uses a subset from the Movie Lens dataset with a total of 3000 movies. The dataset contains many attributes for each movie, but the main ones considered for this project were the *id, title,* and *overview* for each movie in the dataset. Data were uploaded into a jupyter notebook from an excel document and essential columns were filtered out to be analyzed for the project using python. The attribute *overview* contains the essential description of each movie or can be said to be a summary of how each movie unfolds. In addition to the original dataset, a rating dataset was generated for 400 users using a rating scale from 1 to 5 for each movie in the original dataset. The content-based recommender will heavily rely on the *overview* column to make predictions on which movie to recommend to each user. On the other hand, the rating dataset is used to train the collaborative filter model. The methodology of these steps will be discussed in subsequent sections.

Data processing

Data processing is a crucial step in this project as the results will be utilized in different steps in the model building process and analysis of the resulting recommendations. The initial step in the data processing step is the conversion of textual information in the data set into numerical information. This is applied to the *overview* column as mentioned in the previous section. The first step involved in the data processing step is the Term Frequency-Inverse Document Frequency (TF-IDF) method used to determine the numerical representation of each word in the column *overview.* The next step uses cosine similarity to draw similarities between each vector generated using the previous step. The results from these steps will be used in two

portions of this project. Initially, similarity scores will be used to make content-based recommendations using a comparative analysis of existing movies in the dataset with historical data from users' consumption history. Following this, the output of this analysis will be used to compute the diversity in the recommendations made to users from both content-based and collaborative filtering models. The details of data processing using TF-IDF and cosine similarity are given below.

TD-IDF (Term Frequency-Inverse Document Frequency)

In short, TF-IDF is a technique that allows the statistical computation of what words in a document are most important within the context of a collection of multiple other documents. In the case of this project, TF-IDF is applied to the words in the *overview* column of each movie in the dataset. After the application of TF-IDF, movies that have similar words graded as more significant compared to other words in their overviews can be said to be more similar than other movies in the dataset. This lays the foundation for mathematically computing the similarity or difference of items or in this case movies to one another.

According to Scott (2021), and as mentioned above, the higher the Tf-IDF score for a given word in a text, the higher its significance for that text. For example, if there are a total of 10 texts in a document and only one of those texts contains the word "*aliens*", the word "*aliens"* according to the TF-IDF approach is much more relevant within the context of the document compared to other words and will be scored higher for the text containing that word. This is also an indication that the word is rare and therefore significant for the text that contains it in describing the intended information. On the other hand, words such as "*and",* which appear in almost all textual data, will be given lower scores.

TF-IDF is a multiplication of two independent mathematical computations, the term frequency (TF) and the inverse document frequency (IDF). The mathematical computation of the term frequency (TF) simply shows the number of times a word appears in a document divided by the total number of documents in the data.

$$Tf(w,d) = \frac{w\ frequency\ in\ d}{Total\ number\ of\ words\ in\ d} \quad (1)$$

- *Tf* = Term frequency
- *w* = word in a document (text)
- *d* = document

TF is an independent computation done on a single document and has no information regarding the relevance of the word within the context of many other documents or in this case overviews of other movies. This can be accomplished using the inverse document frequency (IDF) method.

The concept of IDF is best illustrated by Falk (2019) who relates the idea to the purchase history of customers. Simply put, he states that if a customer buys a popular item, the information regarding that purchase does not hold enough power to determine how much that customer likes the item. On the other hand, if a customer purchases an item that only a few others have purchased when compared to the total purchase history of other customers, the information of this purchase holds much weight in determining how much the customer values the purchase. When translated into textual analysis, IDF uses the same approach to see how many documents in a dataset contain a word present in a single text. From this, it can be inferred that the fewer documents containing that word, the higher the significance of the word for the documents containing that word.

When computing IDF, one takes the natural log of the number of documents divided by the number of documents containing that specific word. One can easily see that a word that appears in all the documents will have a value of zero as the natural log of 1 will quate to 0. Falk (2019) summarizes the formula for IDF and a translated version of the formula for textual analysis is given below.

$$IDF = \log\left(\frac{total\ number\ of\ documents\ d}{number\ of\ document\ containing\ w}\right) \quad (2)$$

- IDF = Inverse document frequency
- *d* = document
- *w* = word in a document (text)

The cross-product of the two computations above as shown below gives the overall TF-IDF score for a word in a document. The TF-IDF score ranges from 0 to 1 making it easily interpretable for comparative analysis.

$$TFIDF = \frac{w\ frequency\ in\ d}{Total\ number\ of\ words\ in\ d} \times \log\left(\frac{total\ number\ of\ documents\ d}{number\ of\ document\ containing\ w}\right) \quad (3)$$

For the dataset in this project, TF-IDF is applied to the *overview* column as mentioned above. Python's scikit-learn library contains a TF-IDF vectorizer than enables the vectorization of each document in the overview column of the dataset. Having said that, vectorization of all the words such as conjunctions in each review can be counterproductive and time-consuming. The analysis of irrelevant stop words can also skew the result of the computation. Therefore, the stop

word feature of the TF-IDF vectorizer enables the removal of stop words before the analysis takes place. The first 100 words of each review are analyzed for this step of the project. The output of the analysis gives a matrix with rows as the movies and the TF-IDF score for 100 words in each overview as columns as shown in Table 1.

Table 1

TD-IDF scores

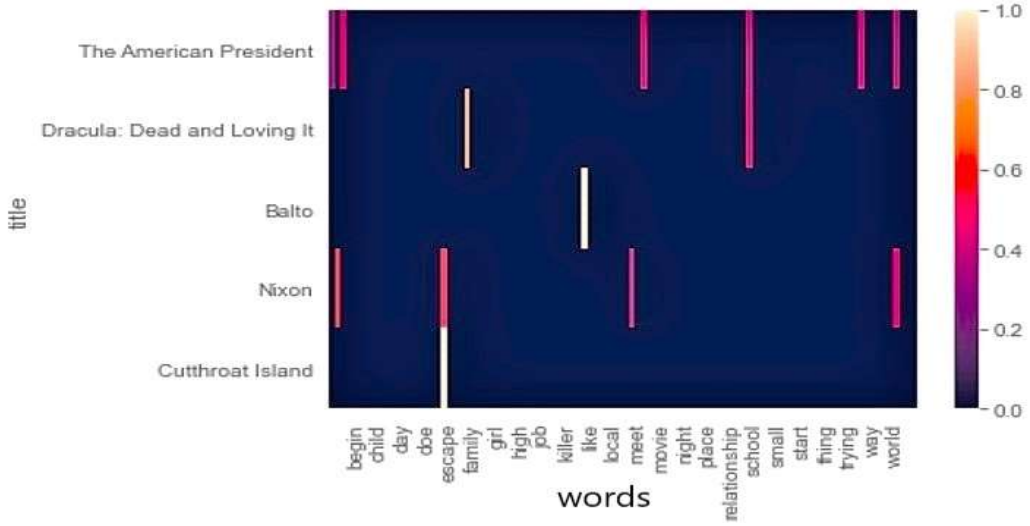
title	american	attempt	beautiful	begin	best	boy	brother	child	city	...	want	war	way	wife	woman	work	world
Jumanji	0.510952	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.344089
Grumpier Old Men	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.000000	0.000000	0.0	0.000000
Waiting to Exhale	0.610904	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.444395	0.000000	0.392696	0.0	0.000000
Father of the Bride Part II	0.504329	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.229489	0.000000	0.0	0.000000

Note: This table made using the pandas library in Python shows a snapshot of the TD-IDF score of the words in the overview column for four movies in the movie dataset.

To see how the similarity of movies can be inferred from Table 1 visually, Figure 4 was made using Python's seaborn library to show a color scheme for a sample of movies depending on how each word is scored using TF-IDF. Movies with similar color arrangements will have similar TF-IDF vectors compared to other movies in the dataset. Furthermore, one can see the significance of a word for two movies in the dataset compared to other movies based on the lightness of the color assigned for a given word for each movie. For example, one can see from the plot that the world "*escape*" is more significant for the movies *"Cutthroat Island"* and "*Nixon*" than it is for movies *"Balto"* or *"The American President"*. Within the same context, the same word is shaded darker for "*Nixon*" than *"Cutthroat Island"*, showing the word has more significance for the latter.

Figure 4

Heatmap for TF-IDF scores of words in a sample of movies



Note: Figure shows the similarity for a sample of movies using a heatmap. Each word on the x-axis is scored differently for each movies on the y-axis. The degree of resemblance or difference for the color arrangment for any two given movies is indicative of their overall similarity or difference based upon the discription given in each movie's overviews.

In this stage of the project, each movie now has a numerical vector representation that can be used to draw similarities among other movies. Figure 4 is used for more of an illustration of how each vector can be compared visually. The next step is to compute this similarity numerically whereby each movie has a score representing its similarity with every other movie in the dataset. Cosine similarity will be used in this project as a method to compute similarity among movies. Details regarding the application of cosine similarity will be discussed below.

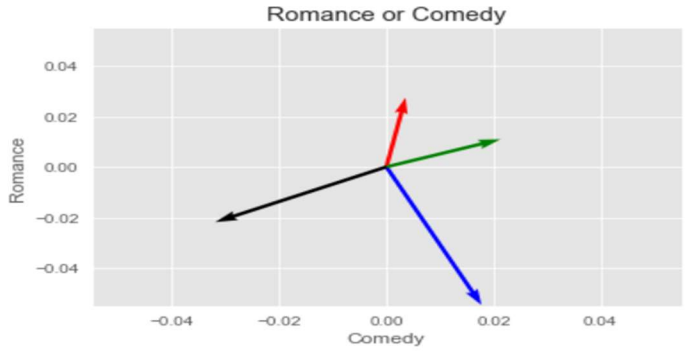
Cosine similarity

After identifying which words are significant to a specific movie, the next step is to compute similarities and differences for all the movies in the dataset. This is done using vector comparison using cosine similarity where the vectors are the TF-IDF scores of the main words

inside the overview of each movie. To illustrate the concept behind cosine similarity, Banik (2018) describes the TF-IDF vectors obtained from the previous step as directions in a multidimensional space. Using cosine similarity, vectors pointing toward a similar direction will be given a higher score compared to vectors pointing in opposing directions given that all vectors originate from the same point. This is visually illustrated in Figure 5. In the figure, two genres of movies, comedy, and romance, are being compared for 4 different vectors in space. Among the four different vectors, the most dominantly romantic movie is the black vector while the blue vector can be said to be a comedy movie. Having said that, the black vector is also pointing downwards in the direction of comedy, and this can be an indication that it is a romantic comedy and not purely a romantic movie.

Figure 5

Visual representation for cosine similarity of 4 sample vectors



Note: The figure made in python shows the direction of 4 vectors for two movie genres, romance, and comedy. Vectors represent the TF-IDF scores while their direction in space is a representation of their cosine similarity. For example, the black vector can be said to be more of a romantic comedy than the blue arrow which is more dominantly comedy.

In cosine similarity, the main interest is not in classifying the movies into any groups but rather in computing the similarity among each one of them. From the figure above, it can be understood that vectors with the smallest angle between them will belong to a similar group of a

subgroup of movie genres than those with a wider angle between them. The implementation of this concept in the movie dataset implies that movies with similar vectors will have smaller angles between them than dissimilar movies. Even if this project uses the overview of each movie as a source for information about the movies, other textual columns such as genre can be included in this analysis.

Mathematically the cosine similarity represents the dot product of two vectors divided by the product of their magnitude. Cosine similarity scores range from -1 to 1. If the similarity score of two vectors is close to 1, that is an indication of their similarity while a negative or a value of 0 would indicate vectors that vary significantly from each other. The formula for cosine similarity is summarized below for two vectors in a multidimensional space (Banik, 2018).

$$sim(v1, v2) = \cos(\theta) = (v1 * v2) \div (||v1|| * ||v2||) \quad (4)$$

- Sim (v1, v2) = similarity score
- Cos (θ) = Cosine of the angle between v1 and v2
- *v1, v2* = vectors in a multidimensional space

As one can see from (4), when applied to each pair of vectors from the TF-ID matrix obtained in the previous step, cosine similarity will now score each movie against every other movie in the dataset as shown in Table 2. This results in a secondary matrix where all the rows and columns are the titles of each movie in the dataset. To see the relative similarity of movies, consider the movie *"waiting to exhale"* from the table which has a similarity score of 0 with *"Grumpier old men"* and a score of 0.31 with *"Father of the Bride"* showing it has more similarity with the latter. Now that all scores are collected, the data is now ready to be used in the model building process which will be discussed in the coming sections.

Table 2

Cosine similarity for a sample of movies in the movie's dataset

title title	Jumanji	Grumpier Old Men	Waiting to Exhale	Father of the Bride Part II
Jumanji	1.000000	0.064428	0.312143	0.440489
Grumpier Old Men	0.064428	1.000000	0.000000	0.084790
Waiting to Exhale	0.312143	0.000000	1.000000	0.308097
Father of the Bride Part II	0.440489	0.084790	0.308097	1.000000

Note: The table shows the cosine similarity of movies against each other in a form of a matrix. The diagonal scores of 1 show the similarity of a movie against itself.

Content-based recommender

As illustrated previously, content-based recommendation solely relies on the attributes of the items being recommended such as the review of movies and users' past consumption history to make predictions. In recommendation engines used by companies such as Netflix, the consumption history of users is stored, and many analytical computations are performed to determine an appropriate candidate or cluster of items from which recommendations can be made. For this project, the rating dataset generated in the previous step is used to determine which movie will be selected as a candidate for other recommendations. The rating dataset contains how well the 400 users in the dataset have rated each movie they have previously watched. The components of building a content-based recommender can be summarized as follows in three individual steps.

- Step 1 Obtaining user profile

- Step 2 Content Analysis
- Step 3 Retrieval of recommendations

User profile

This project uses the *rating* dataset to generate a user profile. Many approaches can be appropriate to determine what set of movies for a user's past consumption history should be used. Having said that, the method used for this step is to aggregate all the movies rated by all users in the dataset and from the grouped items select the top-rated movie for each user.

The highest-rated movies are indicative of what kind of movie a user enjoys watching making them a good candidate to base future recommendations upon. After collecting the *ID* of the highest-rated movies, the *title* of each movie is collected from the original dataset containing that information. Following this, each *title* is passed into a function for further analysis.

Content Analysis

After retrieving the title of the highest-rated movie for each user from the original dataset, the information is passed into a function along with the original dataset, the cosine similarity matrix prepared in previous steps, and the number of recommendations needed for each user. The consecutive steps performed in the function are as follows.

- Step 1. Identify the indices of the *title* passed from the original dataset.
- Step 2. Identify the similarity score of the movie against all other moves using the similarity matrix using the indices found in the previous step.
- Step 3. Delete the similarity score of a movie against itself so that the same movie is not recommended to the user.

Retrieving Recommendations

After finding the similarity score of the movie against all other movies, the final step is to sort all scores in order. Finally, depending on how many recommendations are needed, give the top- N recommendations as an output. Recommendations for all users can be performed using a for loop in python and passing the titles gained in the first step into the recommender function one by one. The number of recommendations retrieved for each user in this project is 10 movies for a total of 400 users, making a total of 4000 recommendations.

Measuring the performance of a content-based recommender

There are different ways to measure the performance of a recommender. Having said that, given the objective of this project, the *diversity* amongst the top-N recommended items is used to measure the performance of the model. The reason for this is illustrated in previous chapters and is given as one of the main limitations of content-based recommendations, the *filter bubble* scenario.

As mentioned earlier, movies with similar TD-IDF scores for specific words will point in a similar direction in the multidimensional space. Content-based recommendation solely relies on textual data and so one can expect most items that are recommended to have similar attributes. This results in recommendation items that are too similar to one another. To see the performance of the content-based recommender, the *diversity* of recommendations is used as a metric to assess the degree of diversity among recommended items. Kaminskas and Bridge (2017) describe diversity as the average pairwise distance between all items in the list of recommendations for a user. Or simply put diversity is the average dissimilarity between a set of recommended items and its formula for the movie dataset can be summarized as follows.

$$Diversity(R) = (\sum i \in R \sum j \in R \backslash \{j\}\, dist(i,j)) \div (|R(|R| - 1|)$$

(5)

- R = set of top -N recommendations
- *i, j*= all possible pairwise combinations of all movies in the top- N recommendations

Given that this project uses cosine similarity to compute the similarity of movies to each other, another approach is to compute diversity indirectly from the similarity scores of each movie. Diversity and similarity can be seen as two different approaches to telling the same story and the formula showing that relationship is given below for a set of N recommendations.

$$Diversity(N) = 1 - sim(N)$$

(6)

- *N*= Top recommendations for a given user
- *Sim(N)* = Mean similarity score of movies in a set of top – N recommendations.

The approach given in (6) is only applicable to a set of recommendations for a single user. To compute the diversity for the total number of users, in this case, 400, the average similarity score must be computed for all users and then averaged again. The steps of this computation can be summarized as follows.

- Step 1. For each set of recommendations, retrieve the similarity score for each movie against each other using a cosine similarity matrix. This step will measure each movie against the other within the set, meaning all pair-wise combinations of movies will be analyzed.
- Step 2. Get the mean similarity score for each set by dividing the sum of the results from step 1 by the number of unique combinations in a set resulting in the similarity scores for each set.
- Step 3. Add all similarity scores for each set of recommendations made and divide the result by the number of users the recommendations are made to.

- Step 4. Finally, subtract the average similarity score of all recommendations made to all users from 1 to obtain the *diversity* of the recommendations.

The results obtained from this computation will be discussed in comparison with other models in chapter 4.

Collaborative filtering model

Unlike content-based recommendations which rely on a single user's consumption history to make predictions, collaborative filtering relies on the rating pattern of other users or rating pattern for items to make accurate predictions. Collaborative filtering can be classified into two main categories namely, *memory-based,* and *model-based collaborative filtering.* This project focuses on model-based collaborative techniques, but the understanding of a memory-based collaborative recommendation is essential to grasping the fundamental application of this approach.

Memory-based collaborative filtering

Memory-based collaboration can further be divided into *user-based* and *item-based* filtering approaches (Saluja, 2018). User-based collaborative filtering is an approach that uses the similarity among all users in a database by analyzing their rating patterns. This can be performed using cosine similarity as discussed in previous topics. In the real world, this approach is computationally expensive to perform due to the growing number of users on eCommerce sites and streaming services. On the other hand, item-based collaborative filtering seeks to find similar items based on the pattern of ratings they received from users. Computationally, this approach is less expensive compared to a user-based filtering approach for the simple reason that there are almost always fewer items on a given platform than there are users on a given platform.

Model-based collaborative filtering

Model-based collaborative approaches are much more robust and computationally less expensive than memory-based collaborative approaches. Model-based filtering can also be subdivided into clustering and matrix factorization algorithms. This project will use a clustering algorithm based on K nearest neighboring (KNN) to make predictions of user ratings on items from which recommendations can be made. But first, it is important to see the fundamental principles of how a KNN algorithm works.

K nearest neighboring (KNN)

Algorithms can be classified into two general categories, supervised and unsupervised machine learning approaches. Unsupervised machine learning uses methods to see hidden patterns in the distribution of variables within a given dataset without having to predict a response variable. On the other hand, unsupervised learning techniques make use of the patterns of variables in a dataset to make predictions of a response variable. KNN is one of the supervised learning methods developed to either classify a categorical variable into a class or make quantitative predictions to an unknown point in a multidimensional space depending on the distribution of other predictor variables in the dataset. KNN uses *feature similarity* to measure the closeness of an unknown point to other K numbers of points in the dataset. To compute the feature similarity between two points, KNN measures the Euclidian distance between the two points, and the formula for the computation of a Euclidean distance is given below.

$$d = \sqrt{[\,(x2 - x1)2 \; + \; (y2 - y1)2]}$$

(7)

- d = distance

- x1, x2 = coordinates of point 1
- y1, y2 = coordinates of point 2

After computing the Euclidean distance, the KNN algorithm then considers the value assigned for K as a measure of how many points should be considered for the prediction of the value for the unknown point in space. In other words, K is the measure of how many of the closest points to an unknown point *P* should be considered. The larger the value of K the more expensive the computation will be. Consecutively, for the collaborative filtering model, the value of K indicates the number of items considered to predict a user's rating of a given movie in the dataset.

The basic steps of a KNN-based algorithm can be summarized as follows.

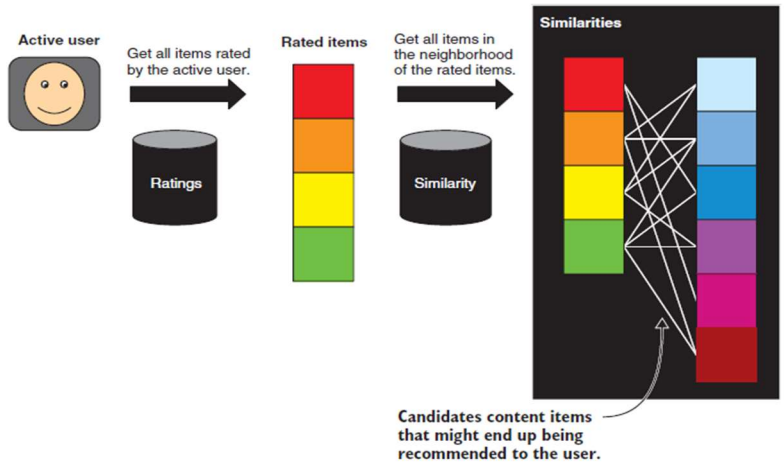
- Step 1. Compute the Euclidean distance between an unknown point and all other points in the dataset.
- Step 2. Choose a value of K for the number of points required to make classification or regression predictions.
- Step 3. Compute the mean of the closest K points in the dataset to the unknown point based on the Euclidian distance computed in step 1 and assign the mean of the known points as a new value to the unknown response variable.

When translated into item-based collaborative filtering using KNN, Figure 6 shows the general stepwise approach of recommending potential items to a user. As illustrated in the figure, a user's rating history for a given set of items is compared to other items in the dataset to compare how others have rated the same products. Items that are found to have similar ratings or in other words are neighbors to the original items will be selected as possible recommendations

for a user. For example, if *user x* rates a *movie x* highly, and other users on the same platform have rated the same movie, *movie x*, highly, movies watched by this group of users can potentially be good recommendations for user x.

Figure 6

Item-based recommendation summary



Note: Reprinted from *Practical Recommender Systems* (p. 198), by Kim Flank, 2019, Manning Publications Co. Copyright 2019 by Kim Flank.

The predictive accuracy of a KNN algorithm can be measured using a root mean squared error metric (RMSE). RMSE is the squared root of the residuals obtained from the average of the difference between actual observations in the dataset and the predicted values using the fitted model. In other words, RMSE equates to how well the observed data points are contracted around the fitted regression line for a given model or in other words the standard deviation of residuals. The formula for RMSE can be summarized as follows.

$$RMSE = \sqrt{(\sum i = (1, N)(xi - xp)^2/N)} \quad (8)$$

- *RMSE* = Root mean squared error
- *N* = total number of data points
- *xi* = Actual observations in the dataset
- *xp* = estimated predicted values of Xi

Given that RMSE is a measure for error, a predictive model with a smaller RMSE value is more powerful than a model with a higher RMSE value. Methods on how to obtain a better fit for the movie dataset will be discussed in the following sections.

When training a KNN algorithm it is very important to not use all the datasets at a given time so that the model can be tested on a dataset it was not trained to predict in the first place. This helps in avoiding overfitting of the model which may result in a low accuracy when the algorithm is applied to a new dataset. This can be achieved by using a method called *cross-validation.* Cross-validation is a process of leaving out a portion (fold) of a dataset in the training portion of an algorithm to be used at the end in assessing the predictive ability of the model.

KNN implementation using the surprise package for python

For this step in the project, the predictor variables (user id and item id) in the training dataset are used to make the predictions of the response variable (ratings) using python's surprise package. A surprise package is a powerful tool that can allow us to perform all the computations mentioned in the section above while using RMSE to determine the predictive ability of the model.

To build the KNN-based collaborative filtering model, the dataset *rating* was loaded into a reader object and a cross-validation of 5 was performed on the whole dataset to make predictions for each iteration of the model training process. At the same time, a K value of 4 was randomly used in the first step of the model building process. The dataset *rating* has the ratings

of all 400 users for all 3000 movies in the dataset, but as mentioned above a portion of the dataset is left for each iteration of training the model goes through. At the end of each iteration, an RMSE was given as a measure of the error in prediction for that iteration. This process was repeated 5 times as the cross-validation value (CV) was set to 5 in the initial step of the training process. After the model training process, a final score of an RMSE value of 1.58 was obtained by averaging the RMSE value from each iteration. The steps of the model building process can be summarized as follows.

- Step 1. Prepare a reader object for the *rating* dataset and load data into the reader.
- Step 2. Declare a KNN model with a specific K value which in this case is 4.
- Step 3. Perform cross-validation with CV = 5 of the KNN model and choose RMSE as a metric of measuring accuracy.
- Step 4. Obtain RMSE value for the 5-fold cross-validation.

Hyperparameter tuning using a grid search

The value of K chosen in the previous step indicates that 4 items are used as a basis for predicting the unknown rating of a user in the dataset. But nothing guarantees that a k value of 4 is optimal in the model building process. To explore more parameters of K, a grid search was conducted using the *GridsearchCV* function that comes with the surprise package in python. This feature allows the use of different values of K in the model training process to predict which value of K gives the smallest RMSE value.

K values of (3,5) and (1,3) were selected and applied to the *GridsearchCV* function. To gain an accurate estimation for the RMSE of the final model, only 90% of the dataset was used in this step while the remaining 10% of the dataset was used to get a final RMSE value for the final model. After multiple iterations, this process gave a value of 5 as the best K value for the

KNN model. Computation of the final RMSE for this model gives a value of 1.55 when applied to the remaining 10% of the dataset that was left out in the first stage of the training process.

After obtaining the best K value from the grid search operation, the model obtained in the first step was trained again using the new parameter and used as the final model for collaborative filtering recommendations. Finally, 10 recommendations in order of their predicted rating were made using this model for the 400 users in the dataset making a total of 4000 recommendations.

Using coverage as a metric to evaluate collaborative filtering

For a collaborative filtering model, the biggest challenge is the sparsity of rating data. In the real world, new products are introduced to streaming services and eCommerce sites and lack data regarding how users have interacted with them. Collaborative filtering cannot recommend items that have zero interactions with customers due to the cold start problem explained in previous chapters.

A good way to measure the performance of a collaborative filtering model to identify this shortcoming is to measure the *coverage* of the final model. Kaminskas and Bridge (2017) describe the coverage as the total number of unique items that can be recommended from the full catalog of items that exist in the original dataset. From this perspective, coverage can be seen as the total diversity in the recommendations made for N number of users. Comparatively, individual diversity has been discussed as the diversity of recommendations made for a single user in previous sections.

The cold start problem has been described in previous chapters as problematic to consumers who do not get the recommendations, they may enjoy due to the lack of rating. But even more serious is that companies that do not investigate this problem are not able to introduce their products to their customers. Therefore, coverage has been identified for this project as a

metric to evaluate the performance of the collaborative model. A simple formula used to compute the coverage percentage for a recommender is summarized below.

$$C(i) = \frac{n}{N} \times 100 \quad (9)$$

- C(i) = coverage for items in the dataset
- n = Total number of unique recommendations made for all users
- N = total number of items in the dataset

To compute the coverage for the collaborative filtering model all recommendations made for users were collected and duplicated movies were removed from the list. The count of the final list of movies was divided by the total number of movies in the original dataset.

Building a hybrid model

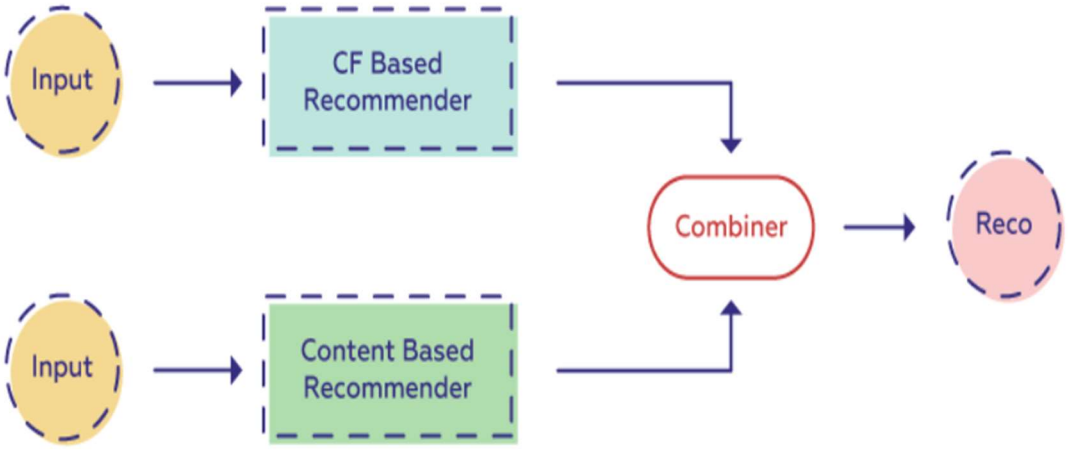
The hybrid model built for this project uses both the content-based recommender and the collaborative filtering model built in previous steps. As indicated in the objective of this project, the goal of this hybrid model is to tackle the limitations of the pure models. To achieve this goal, recommendations from both content-based and collaborative models were combined for each user in the dataset.

To maintain flexibility for the number of content-based versus collaborative recommendations, the hybrid model takes in as parameters the number of recommendations that should be considered for each pure model. This allows to vary the numbers of recommendations from each pure recommendation to see how this relates to diversity and coverage, mentioned in the sections above. But for comparative analysis, the proportion of recommendations made by each model was kept constant with 20% of recommendations made using a content-based

approach while 80% of the recommendations made using collaborative filtering. The overall design of the hybrid model for this project can better be illustrated using Figure 7 obtained from Sciforce (2021) who highlights the approach of hybridization where the results from the two models do not precede each other but are simultaneously computed.

Figure 7

Hybridized model design



Note: The figure illustrates the overall design for the hybrid model used for this project. The proportion of recommendations made using a content-based approach was 20% and 80%of recommendations were made using collaborative filtering. Reprinted from How a recommender system recommends, by Sciforce, 2021. https://medium.com/sciforce/inside-recommendations-how-a-recommender-system-recommends-9afc0458bd8f

The methodology of measuring these metrics for the recommendations made by the hybrid recommender is identical to the steps discussed in previous sections for the content-based and collaborative recommenders. The results of this analysis will be covered in chapter 4.

Sequential steps in the comparison of the three models

To assess all the models built at this stage of the project, a function was built to compute the mean similarity for each set of recommendations by a model, the overall similarity score for

total recommendations, diversity, coverage, and titles of all the movies recommended by each model was built.

Initially, a comparison with the content-based and collaborative-filtering model was performed to assess how each model is different from the other within the context of this project, *filter bubbles,* and *data sparsity*. This comparison does not include the hybrid model but was done to highlight the strength and weaknesses of each pure model. Furthermore, this comparison gives the reason why the two models were picked for this project for the complementary nature of their strengths and weaknesses.

Following this, the hybrid model was compared with the purely content-based recommender and collaborative-filtering models respectively. More specifically, the hybrid model was compared with the content-based recommender for diversity and compared with the collaborative-filtering recommender for coverage to highlight how the weaknesses of the two models can be mitigated using this approach.

Steps in comparing models on diversity Vs coverage

Comparing the diversity of the models is very straightforward compared to comparing models based on coverage. It was mentioned that the rating data generated for 400 users contains how all users have rated all 3000 movies in the dataset. The application of this data to the collaborative-filtering model does not highlight how the item-based cold start problem severely impacts its performance. For this reason, it was essential to generate new data for varying levels of data sparsity where some users have not rated a set of movies in the movies data set. The six data sparsity levels picked for this project were 0%, 16.7%, 33.3%, 50.0%, 66.7%, 83.3% and 91.7%. A data sparsity level of 0% is the same as the original *rating* data mentioned above while a 91.6% sparsity level indicates how many data points are missing from the original *rating*

dataset. This is done to simulate a real-life scenario and is mostly the case for many online items that users consume. For example, it is impossible to expect that all users have rated all movies on Netflix.

The coverage values for the collaborative-filtering model were then compared with the hybrid model to measure how well they both sustain these levels of sparse data. This process is repeated for the six levels of data sparsity levels mentioned above. A k value of 4 was used for all KNN-based collaborative-filtering models trained in this process. Additionally, the hybrid model was made to give out 2 recommendations based on the content-based recommender and 8 recommendations using the collaborative-filtering recommender for a total of 10 hybrid recommendations made for a single user. These parameters were kept constant throughout the evaluation process to have a fair comparison of the models at the end. The main steps involved in the process of generating different sparse data and computing coverage for all levels are summarized below.

- Step 1. Generate new sparse rating data for all 400 users for the selected level of sparsity.
- Step 2. Train a new KNN- based collaborative model using the sparse data from step 1 and a k value of 4.
- Step 3. Input the new collaborative model into a new hybrid model.
- Step 4. Extract the highest-rated movie for each user from the new sparse rating data for the pure content-based recommendations.
- Step 4. Make purely content and collaborative recommendations from the new hybrid model.
- Step 5. Make hybrid recommendations.
- Step 6. Compare coverage between hybrid and collaborative models.

The above process was mainly designed to compare the collaborative-filtering model with the hybrid model. To compare the hybrid model with the purely content-based recommender based on *diversity*, the original *rating* data was used with a 0% sparsity level. The result from this comparison could suffice in concluding how a hybrid model can help overcome the filter bubble scenario. Having said that, it can be intuitively understood that sparse data where users have rated only a portion of the movie dataset means that there is little diversity in the users' view history upon which a content-based recommendation can be made. For this reason, the content-based recommendation was also compared to the hybrid model on levels of data sparsity based on *diversity.*

Conclusion

So far in this chapter, the methodologies of building all three models were discussed in detail. The use of TD-IDF and cosine similarity and their role in building a content-based recommender were discussed. Furthermore, the principle behind KNN models was discussed to illustrate how item-based collaborative-model can be built using this approach. Given the objective of this project, *diversity* and *coverage* have been discussed as metrics used to compare the performance of all three models. More specifically, diversity was discussed to compare how the models avoid filter bubbles. On the other hand, coverage was mentioned as a tool to compare how data sparsity impacts the models. The results of all comparisons and their implication within the context of this project will be discussed in chapter 4.

Chapter 4: Findings and results

Introduction

In this chapter, the essential findings from the analysis of the three models, content-based, collaborative, and hybrid models will be presented within the context of the objectives of this project which is to overcome *filter bubbles* and mitigate the *item-based cold start* problem. The shortcomings of each model will be highlighted initially based on the findings of the case study. Following that, the chapter will cover how purely content-based recommender sufferers a filter bubble scenario and how it measures against a hybrid model. On the other hand, the chapter will also show how a purely collaborative-filtering model suffers when applied to sparse data to highlight the cold start problem. The results of this analysis will then be compared with the hybrid model.

To highlight the significance of the findings in a manner that highlights the importance of the hybrid model compared to the two pure models, this chapter will be structured in the following way.

- **Part 1** will cover the comparison of only content-based and collaborative-filtering models based on the *diversity* and *coverage* obtained from all 4000 recommendations made by each model. Popularity plots of the two models and the distribution of similarity scores from the two models will be discussed to highlight the concepts of diversity and coverage.
- **Part 2** will cover the comparison of the hybrid model with the content-based recommender based on the *diversity* of the recommendations made by each model.
- **Part 3**. Comparison of the hybrid model with the collaborative-filtering model based on the *coverage* of the recommendations made by each model for varying data sparsity levels.

- **Part 4** will cover the comparison of the hybrid model with the content-based recommender based on the *diversity* of the recommendations made by each model for varying levels of data sparsity levels.

Findings

Comparing a Content-based and the collaborative-filtering models

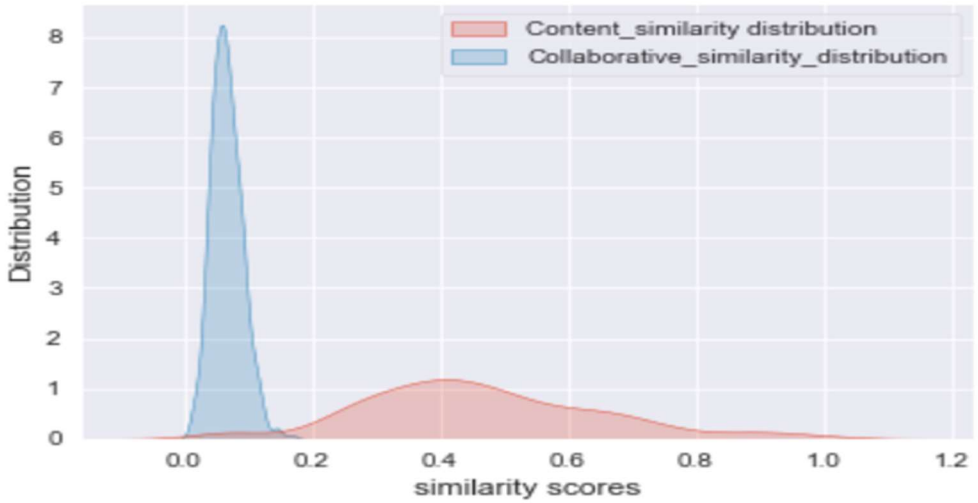
As mentioned in earlier chapters, the two models have strengths and weaknesses that complement each other. In the first comparison made between these models, the similarity for each set of 10 recommendations for every 400 users was considered to assess their diversity. Each set of recommendations has its mean similarity score which then is averaged to obtain the overall mean similarity score of the two models as mentioned in chapter 3. Upon analysis, the content-based recommender had a mean similarity score of 0.46 while the collaborative-filtering model had a mean similarity score of 0.07.

As seen in (6) from chapter 3 and according to Banik (2018), the *diversity* score for the two models was computed by subtracting these values from 1 giving a diversity value of 0.53 and 0.93 for the content-based recommender and collaborative-filtering models respectively. The results from this comparison highlight that the collaborative-filtering model is far better at giving more diverse recommendations to the 400 users in the dataset and its ability to better handle a filter bubble scenario compared to the content-based recommender. This is also illustrated by Isinkaye et al (2015) who highlight the two models in contrast while addressing the filter bubble problem. This is also illustrated in Figure 8 which illustrates the distribution of the mean similarity score for each set of recommendations made by the two models. As shown in the

distribution plot, the collaborative model's mean similarity score distribution, highlighted in blue, is smaller than the content-based recommender highlighted in red.

Figure 8

Mean similarity score distribution plot



Note: This plot made using seaborn in python, shows that the collaborative model gives far more diversified recommendations compared to the content-based recommender for the 400 users in the dataset as shown in its low scores for similarity compared to the content-based recommender. The scores are collected from 10 sets of recommendations made to each user by the two models.

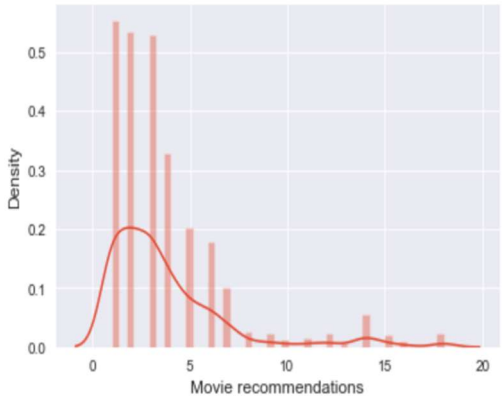
In the analysis made above, the principles of similarity and diversity are used to illustrate how the collaborative-filtering model is better at giving more diversified recommendations for each set of recommendations made to each user. Another approach to assess the two models is to also measure how much redundancy there is in the total recommendations made by the two models to all users. This concept is the same as measuring *coverage* for the two models to assess how many unique movies are recommended by the two models from the total movies available in the original dataset. The coverage values for both models were computed using (9) from chapter 3. Upon analysis, it was found that the content-based recommender had a coverage score of 0.59 while the collaborative-filtering model had a score of 0.68. This means that more movies

were made available to users using the collaborative-filtering model compared to the content-based recommender.

This is further illustrated using a popularity plot of the total recommendations made by the two models as shown in Figure 9 and Figure 10**.** The plots show that results from the content-based recommender have more redundant recommendations compared to the results from the collaborative-filtering model. This is apparent while looking at the trend line of the two plots. Specifically, the content-based recommender has a steady decline in the trend line showing that most recommendations are concentrated on specific movies in the dataset. On the other hand, the trend line for the collaborative-filtering model shows that more movies in the dataset were included in the recommendations to users as shown in the trend line's horizontal arrangement.

Figure 9

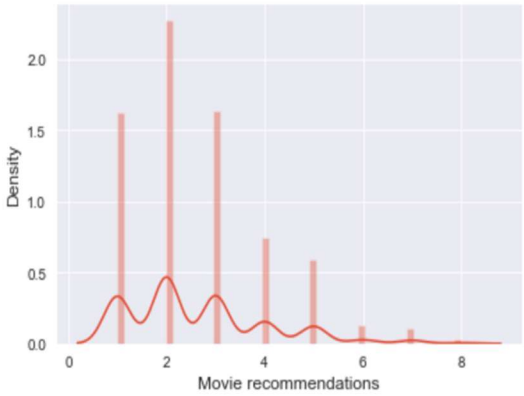
Popularity density plot for content-based



Note: Made using seaborn in python, plot shows the number of unique numbers of movie recommendations made using a content-based recommender**.**

Figure 10

Popularity density plot for collaborative-filtering



*Note***:** Made using seaborn in python, plot shows the number of unique numbers of movie recommendations made using a collaborative-filtering model**.**

So far the shortcomings of a content-based recommender are highlighted by comparing it to a collaborative-filtering model, it is still important to remember that the content-based recommender unlike the collaborative-filtering model does not suffer from the item-based cold start problem. This is the reason why the two models were picked to be hybridized and enhance

each other's performance. The shortcoming of a collaborative-filtering model will be discussed within the context of data sparsity to highlight its inability to mitigate the item-based cold start problem. But first, a comparison between the hybrid model and the content-based recommender will follow.

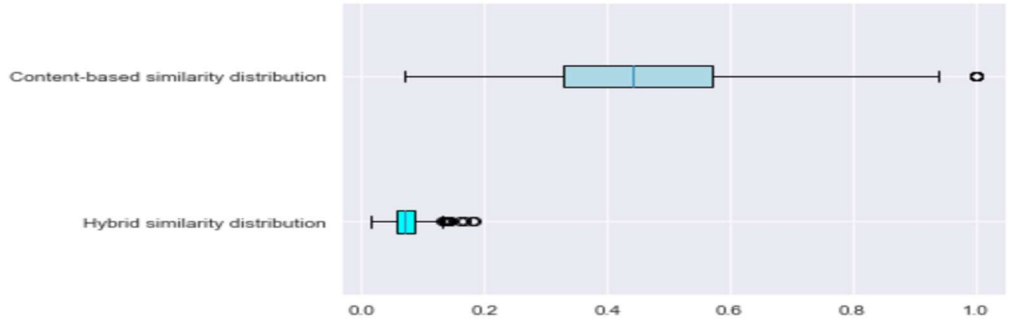
Comparing the hybrid model Vs content-based recommender

As mentioned in previous chapters, one of the main reasons why content-based recommendations suffer a filter bubble scenario is because it solely depends on historical data from users' consumption history to make future recommendations. This can eventually lead to disengagement from users as discussed in previous chapters. Due to the hybridized model's ability to incorporate collaborative filtering, this problem is mitigated in a significant way. This was apparent for this project whereupon analysis of the similarity scores obtained by the two models, the hybridized model gave a mean similarity score of 0.07 while pure content-based recommendations gave a mean similarity score of 0.46. This is also shown in Figure 11 showing the overall mean similarity score distribution of the two models.

To assess the significance of the difference between the two mean values, a Mann-Whitney U test was conducted in python which as its null hypothesis assumes that the two distributions come from the same distribution or that the difference in mean of the two samples is not significant. Upon conducting the test, the null hypothesis was rejected and the difference in mean of the two populations was found to be significant with a P-value = 0.00 at a 0.05 significance level.

Figure 11

Mean similarity score distribution for Hybrid and content-based recommenders



Note: Box plot shows that the overall similarity in recommendations of a content-based recommender is significantly higher than the hybrid model. The plot shows how the hybrid model can diversify recommendations and thus mitigate filter bubble scenarios while making recommendations for users.

Comparing the hybrid model with the collaborative-filtering model

The shortcoming of a collaborative-filtering model was discussed in previous chapters regarding its inability to withstand the item-cold start problem. The hybrid model's content-based recommendations enable the mitigation of this problem by incorporating recommendations that solely rely on items' descriptions or in this case the overview of each movie in the dataset using content-based recommendations. Furthermore, it is already discussed that a purely collaborative-filtering model cannot give recommendations of movies that lack any rating data from users and how coverage is used to highlight this limitation.

When measured against the collaborative-filtering model using the original *rating* data set with 0 % data sparsity level or what Kula (2015) calls a warm-start as suppose to a cold-start, the hybrid model showed only a 1.33% increase in coverage and 1.13% increase in coverage for a 16.70% data sparsity level as shown in Table 3. But as the sparsity level was increased, the percentage improvement of the hybrid model compared to the purely collaborative-filtering

model was significantly higher with a maximum value of 17.43% increase in coverage for a 91.67% data sparsity level.

Table 3

Coverage values for hybrid and collaborative-filtering models

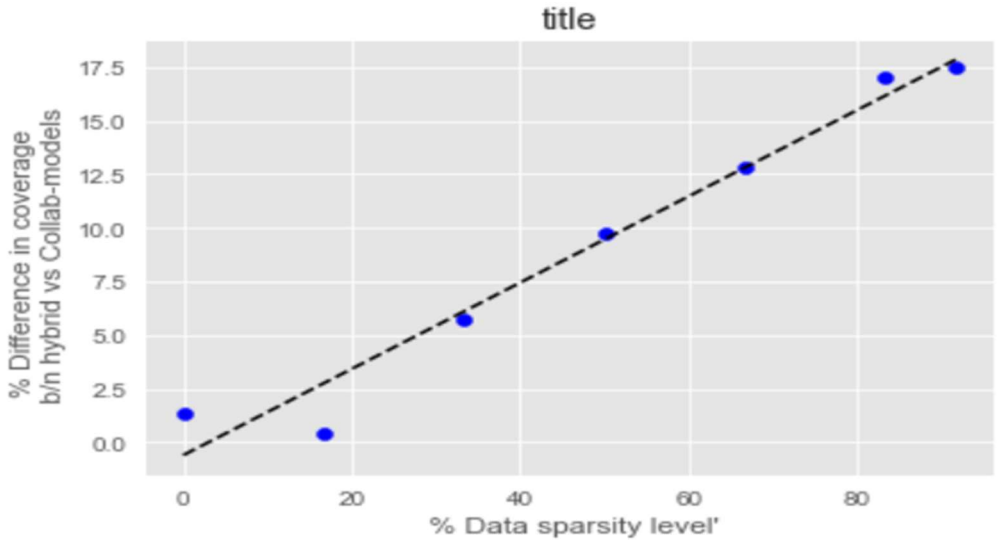
% Data sparsity level	Collaborative coverage	Hybrid coverage	% Difference in coverage b/n hybrid model vs Collab-model
0.00	0.681333	0.694667	1.333333
16.70	0.620000	0.623667	0.366667
33.30	0.541000	0.598667	5.766667
50.00	0.442000	0.539333	9.733333
66.67	0.318333	0.446333	12.800000
83.33	0.166667	0.337000	17.033333
91.67	0.083333	0.257667	17.433333

Note: The table summarizes the increase in the percentage difference of total movies explored in the movie dataset using a hybrid model compared to a collaborative model.

The continued increase in the difference in coverage between the hybrid model and the collaborative-filtering model is better illustrated in Figure 12. The plot illustrates how using a hybrid model instead of a pure collaborative-filtering model in high levels of data sparsity can help mitigate the item-based cold start problem. It must be noted that any value above 0% for a difference in coverage is evidence that the hybrid model is outperforming the purely collaborative-filtering model. The most significant finding is the increase in percentage difference of coverage for increasing levels of data sparsity between the two models as shown using a trend line in Figure 12.

Figure 12

Percentage difference in coverage between hybrid and collaborative models



Note: The increase in the difference in coverage of the hybrid model compared to the collaborative model illustrates how using a purely collaborative-filtering model in high levels of data sparsity can lead to poor performance as measured by coverage in its inability to mitigate the cold start problem.

Assessing diversity in varying levels of data sparsity

Another way to compare the hybrid model and the content-based recommender is to assess the diversity of recommendations made to each user at different data sparsity levels. The reason for this comparison is that, even if the hybrid model can mitigate the item-based cold start problem, its performance is still hindered at high data sparsity levels even though it is to a lesser extent than a purely collaborative-filtering model. Given that 80% of recommendations from the hybrid recommender are based on collaborative filtering, it is evident that at high data sparsity levels, these recommendations are made on only a few subsets of the total movie dataset. Therefore, the number of diverse recommendations made by the hybrid recommender for a specific user can also be said to be limited.

To test this hypothesis, the diversity of recommendations made by the hybrid model and the content-based recommender were compared at the same data sparsity levels mentioned above. The results from this comparison show that, despite the assumption above, there remains still a significant increase in the diversity of recommendations made by the hybrid model irrespective of the data sparsity level. As shown in Table 4, the difference in the diversity scores for movies recommended by the hybrid model were all above 38%.

Table 4

Diversity values for the hybrid model and content-based recommender

% Data sparsity level	Content model diversity	Hybrid model diversity	% Difference in diversity
0.00	0.535078	0.927037	39.195864
16.70	0.428490	0.912906	48.441589
33.30	0.526261	0.925958	39.969780
50.00	0.518247	0.924863	40.661618
66.67	0.538587	0.922692	38.410487
83.33	0.526765	0.923430	39.666460
91.67	0.523503	0.917408	39.390465

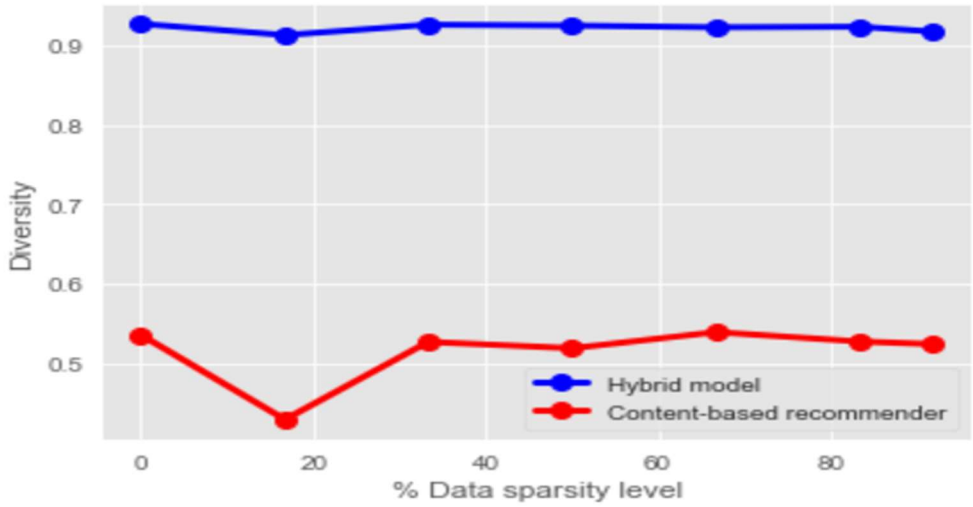
Note: The table shows that despite the sparsity level in the *rating* dataset, the hybrid model consistently gives a highly diversified recommendation to users compared to a purely content-based recommender.

The findings discussed above are presented in Figure 13 for more clarity. As shown in the figure, there remains a significant difference in the diversity of recommendations made by the hybrid model compared to the content-based recommender irrespective of the data sparsity level. The diversity of recommendations made by the hybrid model all have values above 0.90 showing 90% or more diversity in the set of recommendations made to each user. On the other hand, all

recommendations made by the content-based recommender have a diversity score below 0.55 showing 55% or less diversity in the set of recommendations made to each user.

Figure 13

Hybrid model Vs content-based recommender diversity scores



Note: The plot shows that there remains a significant difference in diversity score between recommendations made by the hybrid model compared to recommendations made by the content-based recommender.

This finding is further evidence as to how the hybrid model helps mitigate the filter bubble problem, unlike a purely content-based recommender. This again is due to the collaborative-filtering approach integrated into the hybrid model. The comparison made between the purely collaborative-filtering and content-based recommender has already been highlighted earlier in Figure 8.

Conclusion

So far in this chapter, the findings of each model have been presented in a manner that highlights the appropriateness of hybridization to overcome the filter bubble and item-cold start

problem. Findings from each analysis show the weakness and strengths of each approach. Furthermore, coverage and diversity have been used in conjunction with similarity scores to demonstrate how each model performs in different data sparsity levels. The hybridized model has shown that it can significantly improve diversity compared to a pure content-based recommender. Similarly, the hybrid recommender has shown that it can withstand higher data sparsity levels much better than a purely collaborative approach. A summary of these findings and suggestions for future research will be given in chapter 5.

Chapter 5: Summary, Recommendations, Conclusion

Introduction

So far in this case study, several assumptions about the recommendation systems used for the project have been explained and put to test using various applications. This chapter will continue to illustrate the findings of these applications and their relevance in achieving the project objective. Furthermore, suggestions for future research with related objectives will be discussed. The overall structure of this chapter will look as follows.

- **Part 1** will discuss the summary of the main findings discussed in chapter 4. The discussion will include a summary of the comparisons between different models and the relevance of the findings in the real world.
- **Part 2** will discuss the suggestions for future research and how shortcomings that were not tackled using the applications for this project can be tackled using other approaches.
- **Part 3** will conclude this section by giving a quick rundown related to how the project objectives were reached for this project by revising relevant points from previous chapters.

Summary of findings

As mentioned throughout this project, the objectives of hybridization are to limit filter bubbles and mitigate item-based cold start problems in recommendation systems. Due to the complementary nature of a content-based recommender and collaborative-filtering model, the two were hybridized to overcome the obstacles mentioned above. The hybrid model shown in chapter 3 was able to achieve these objectives and outperformed both the content-based recommender and collaborative filtering model as shown in chapter 4. The most significant findings from the analysis in chapter 4 are as follows.

Improving diversity

When compared with a purely content-based recommender, the hybrid model has shown a statistically significant increase in recommendation diversity. The comparison was made using varying levels of data sparsity values to assess the increase or decrease of the diversity in recommendations. On all levels of data sparsity, the hybrid model was able to give significantly diverse recommendations. This finding is in line with the first project objective which is to overcome filter bubbles.

Combating item-based cold start problem

Given that items with no rating data are not able to be recommended using a purely collaborative-filtering model, coverage was used as a metric for measuring the performance of this model. A comparison between purely collaborative filtering and a hybridized model reveals that more unique items were included in the set of recommendations to users when using a hybrid model. More importantly, an increase in data sparsity values severely altered the performance of the purely collaborative-filtering model. On the other hand, the difference in coverage between the hybrid and collaborative-filtering model showed a steady increase as the data sparsity level was increased. This finding shows how the hybridized model was able to mitigate the item-based cold start problem which is stated as the second objective of this case study.

Recommendations

Studies in the future can use numerous ways to enhance the findings of this study. Some of the main approaches that can help future studies are discussed below.

Textual analysis from multiple sources

The content-based recommender applied to this project makes use of the *overview* of each movie to draw similarities. Even though this approach was kept constant throughout the project for a final fair comparison between models, expanding the amount of textual data can be beneficiary. This can be achieved using textual data such as the name of directors, production companies, and taglines from each movie as part of the analysis to compute similarity among users as shown by Paialunga (2022) who uses three attributes instead of one for a content-based recommender. Even if the difference computed between the hybrid and content-based recommenders was found to be significant for this project, this approach can help highlight the shortcoming of a pure content-based recommender to a higher degree than what was presented in chapter 4.

Simulate the ratio of content-based and collaborative filtering recommendations

The hybrid model used for this project bases 20% of the recommendations on the content-based model while 80% of recommendations are made using a collaborative -filtering approach. The results from the hybrid model using this ratio enabled the project objectives to be reached. Having said that, a more dynamic comparison of the models can be made using by simulating different values for the ratio of the 10 recommendations made to each user. As explained by Shin (2021), among the different benefits of simulation are visuals for varying levels of parameters, forecasting outcomes, and identifying what percent change in paraments is associated with a specific value for the outcome of the study.

When translated to this project, simulation can give insight as to what ratio is appropriate for a specific data sparsity value for a dataset using both numerical and visual analysis for differing values of parameters. By allowing such analysis, it is easy to also translate the models

built to other datasets. For example, two datasets with the same data sparsity level mustn't be treated identically given the objective of the recommendations made using the two datasets might be different.

Resolving user-based cold start along with item-based cold start

As mentioned previously in this project, this case study is limited due to its inability to deal with user-based cold start problems even if it allows to significantly mitigate the item-based cold start problems. User-based cold start problems are prevalent in the real world due to the continued increase of internet users with very limited information about what users prefer.

One of the ways by which new users' preferences can be incorporated into the recommendation process for this project is to give users a chance to briefly identify their preferences by entering items they might have already consumed (Wu, 2022). When applied to the movie dataset, this approach can look like asking users to rate a movie they have already watched sometime in the past. From the input they provide, some assumptions can be incorporated into the recommendation-making process. This approach is also called a knowledge-based recommendation approach as it relies on what users already know and can attest to regarding their preferences. Therefore, integrating a knowledge-based recommendation into the hybridized model built for this project can partially enable the mitigation of the user-based cold start problem.

Another approach to mitigate a user-based cold start problem is to utilize cross-domain recommendation techniques (Mirbakhsh & Ling, 2015). This is especially useful to eCommerce sites that find it difficult to directly predict users' preferences. For example, recommendations to new users can be made not based on their consumption history but by the types of ads they click on or which items they spent time viewing the most. Data regarding online activity may not

directly be linked to what users would like to purchase, nevertheless, it can be utilized to acquire information about users that would otherwise be impossible. This is shown in a study conducted by Jin et al. (2020) using a dataset from amazon. To solve the user-based cold start problem, the study looks at users' engagement across all domains of products on Amazon. Reviews left by users on other domains such as *sports* are then used to make predictions on what the user would prefer in other domains such as *cell.* This approach can easily be applied to sites with multiple domains for the products they provide. A similar approach can be implemented on the movie dataset if there is data regarding users' book preferences which then might be used to make predictions on what movies each user likes in cases where there is high data sparsity in the original movie dataset.

Using deep learning instead of a content-based recommender

Another approach that can be used to enhance the hybrid model built for this project is to use a model built using deep learning instead of a content-based recommender along with collaborative filtering. Deep learning can use item-feature just as a content-based recommender to extract similarities between items and users' consumption history. Having said that, the benefit of using deep learning instead of a content-based recommender is to enhance the scalability of the hybrid model Singhal et al. (2017). The hybrid model built for this project uses a linear transformation of data where each word in the *overview* column is numerically translated using TD-IDF. Such a computation can be extremely expensive when applied to a dataset with a significant amount of data. On the other hand, deep learning can do non-linear complex assessments on item-user interactions that are not evident when doing linear computations while allowing more room for scalability (Sciforce, 2022).

Using metrics such as serendipity for comparison

The performance of each model built for this project was measured using diversity and coverage along with similarity score distributions. Having said that, incorporating metrics such as serendipity can also be beneficial to assess the diversity of recommendations. For example, Ge (2010), illustrates the idea behind using serendipity as a measure of relevance and how surprising or attractive a given recommendation is. Although diversity is closely associated with serendipity, the latter can not only assess the diversity of recommendations but how likely will users find the recommendations enjoyable. This concept is especially important when it is impossible to gather direct information from users, as is the case for this project, regarding the quality of the recommendations given to them.

Devise a method to gather the users' experience

The performances of each model for this case study were compared using important metrics, coverage, and diversity, which are directly linked to the objectives of the project. As an additional procedure, future studies can incorporate an experiment that allows users to input their level of satisfaction based on different sets of recommendations. Although a study with this kind of approach can be difficult to conduct for an individual, it can be easily accomplished at a company level. For example, when applied to eCommerce sites and streaming services such as Netflix, a quick question directed to users regarding how appropriate the recommendations made were can help the companies rate and fine-tune their recommendation approaches.

Conclusion

So far in this chapter, all relevant findings of the case study were summarized within the context of the project objectives. The hybridized model built to overcome item-based cold start problem and filter bubbles was discussed in comparison with the two pure models to highlight

how each shortcoming was mitigated. Furthermore, relevant concepts that can help studies in the future with similar or related objectives were covered in detail. The topic of recommendation systems will continue to evolve with the development of new algorithms and online services that require a specific approach to product recommendation. This case study has covered how that can be accomplished by tailoring hybridized models to achieve specific results.

References

Banik, R. (2018). *Hands-On Recommendation Systems with Python: Start building powerful and personalized, recommendation engines with Python*. Packt Publishing.

Berkay, B. (2022, January 6). *Hybrid Recommender System - Berkay*. Medium. https://iambideniz.medium.com/hybrid-recommender-system-315e86377f50

Chen, S. (2021, December 13). *Introduction to Recommendation Systems - Becoming Human: Artificial Intelligence Magazine*. Medium. https://becominghuman.ai/introduction-to-recommendation-systems-587f644b0ab6

Claypool, M. (1999). *Combining Content-Based and Collaborative Filters in an Online Newspaper | Semantic Scholar*. SemanticScholar. https://www.semanticscholar.org/paper/Combining-Content-Based-and-Collaborative-Filters-Claypool-Gokhale/21cefe0b5fbd4a7bd258d25255f6bfce4dca3306

Flank, K. (2019, January). Practical Recommender Systems [Figure]. In *Practical Recommender Systems* (1st ed.).

Ge, M. (2010). *Beyond accuracy: evaluating recommender systems by coverage and serendipity | Semantic Scholar*. Semantic Scholar. https://www.semanticscholar.org/paper/Beyond-accuracy%3A-evaluating-recommender-systems-by-Ge-Delgado-Battenfeld/16c029e8f4bc3b662b0ad89d15dd57ff567f3726

Grčar, M., Mladenič, D., Fortuna, B., & Grobelnik, M. (2006). *Data Sparsity Issues in the Collaborative Filtering Framework*. SpringerLink. https://link.springer.com/chapter/10.1007/11891321_4?error=cookies_not_supported&code=7deb5b1c-9d63-4ee9-a710-a1f0bddfd7d4

Grimaldi, E. (2019, June 25). *How to build a content-based movie recommender system with Natural Language Processing*. Medium. https://towardsdatascience.com/how-to-build-

from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243

Gupta, S., & Dave, M. (2019). A Recommendation System: Trends and Future. *International Journal of Engineering and Advanced Technology*, *8*(6S3), 1361–1364. https://doi.org/10.35940/ijeat.f1240.0986s319

Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods, and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

Jin, Y., Dong, S., Cai, Y., & Hu, J. (2020). RACRec: Review Aware Cross-Domain Recommendation for Fully-Cold-Start User. *IEEE Access*, *8*, 55032–55041. https://doi.org/10.1109/access.2020.2982037

Kaminskas, M., & Bridge, D. (2017). Diversity, Serendipity, Novelty, and Coverage. *ACM Transactions on Interactive Intelligent Systems*, *7*(1), 1–42. https://doi.org/10.1145/2926720

Kula, M. (2015, July 30). *Metadata Embeddings for User and Item Cold-start Recommendations*. arXiv.Org. https://arxiv.org/abs/1507.08439

Le, J. (2018, June 20). *The 4 Recommendation Engines That Can Predict Your Movie Tastes*. Medium. https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52

Lee, K. J. D. P. J. (2004, September 23). *Hybrid Collaborative Filtering and Content-Based Filtering for Improved Recommender System | ScienceGate*. ScienceGate. https://www.sciencegate.app/document/10.1007/978-3-540-24685-5_37

Mirbakhsh, N., & Ling, C. X. (2015). Improving Top-N Recommendation for Cold-Start Users via Cross-Domain Information. *ACM Transactions on Knowledge Discovery from Data*, *9*(4), 1–19. https://doi.org/10.1145/2724720

Paialunga, P. (2022, January 22). *Hands-on Content Based Recommender System using Python*. Medium. https://towardsdatascience.com/hands-on-content-based-recommender-system-using-python-1d643bf314e4

Pandey, A. (2021, December 15). *How to improve recommendations for highly sparse datasets using Hybrid Recommender Systems?* Medium. https://ai.plainenglish.io/how-to-improve-recommendations-for-highly-sparse-datasets-using-hybrid-recommender-systems-1a4366e65cff

Pinela, C. (2018, March 31). *Recommender Systems — User-Based and Item-Based Collaborative Filtering*. Medium. https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f

Rocca, B. (2021, December 10). *Introduction to recommender systems - Towards Data Science*. Medium. https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

Saluja, C. (2018, July 8). *Collaborative Filtering based Recommendation Systems exemplified.*. Medium. https://towardsdatascience.com/collaborative-filtering-based-recommendation-systems-exemplified-ecbffe1c20b1

Sciforce. (2022, February 2). *Deep Learning Based Recommender Systems - Sciforce*. Medium. https://medium.com/sciforce/deep-learning-based-recommender-systems-b61a5ddd5456

Sciforce, S. (2021, December 13). *Inside recommendations: how a recommender system recommends*. Medium. https://medium.com/sciforce/inside-recommendations-how-a-recommender-system-recommends-9afc0458bd8f

Scott, W. (2021, December 7). *TF-IDF from scratch in python on a real-world dataset.* Medium. https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Shin, T. (2021, December 16). *Building Simulations in Python — A Step by Step Walkthrough*. Medium. https://towardsdatascience.com/building-simulations-in-python-a-complete-walkthrough-3965b2d3ede0

Singhal, A., Sinha, P., & Pant, R. (2017). Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works. *International Journal of Computer Applications*, *180*(7), 17–22. https://doi.org/10.5120/ijca2017916055

Wu, J. (2022, March 30). *Knowledge-Based Recommender Systems: An Overview - Jackson Wu*. Medium. https://medium.com/@jwu2/knowledge-based-recommender-systems-an-overview-536b63721dba

Yasar, K. (2018, July 16). *Recommender Systems: What Long-Tail tells? - Kadir Yasar*. Medium. https://medium.com/@kyasar.mail/recommender-systems-what-long-tail-tells-91680f10a5b2

Yin, H., Cui, B., Li, J., Yao, J., & Chen, C. (2012). Challenging the long tail recommendation. *Proceedings of the VLDB Endowment*, *5*(9), 896–907. https://doi.org/10.14778/2311906.2311916

Zhang, Y. C., Séaghdha, D., Quercia, D., & Jambor, T. (2012). Auralist. *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining - WSDM '12*. https://doi.org/10.1145/2124295.2124300

Appendix

All codes for this project are compiled into one PDF file which can be found using the link below.

https://github.com/DawitNerea/DS-785-Capstone_code/blob/main/Capstone_final_code%20-%20Jupyter%20Notebook_PDF.pdf