Ruizi Xu. A Hybrid Recommendation System for Online Car Auction Platform and Product Review
. A Master's Paper substitute for the M.S. in I.S degree. December 2021. 27 pages. Advisor: Robert Capra

Abstract:

Recommender systems are a relatively new field of interest in data science. There are roughly 2 approaches to recommender systems: content-based filtering and collaborative filtering. For this master's paper substitute/project, I designed and implemented a hybrid recommendation system combining the two approaches. This document will detail the methods I used and the process of arriving at what I had. It will also include some of the obstacles I faced during the process, how I addressed them, and the ways the project changed because of it

Subject Headings:

Recommendation system

Machine learning

Data analysis

A HYBRID RECOMMENDATION SYSTEM FOR ONLINE CAR AUCTION PLATFORM, AND THEN PRODUCT REVIEW

by
Ruizi Xu

A Master's paper substitute submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

December 2021

Approved by

_____

Robert Capra

Table of Contents

1. Literature Review

In this piece I will review some relevant literatures discussing different types and implementations of recommendation systems, evaluate the effectives of each instance for my project, as well as discuss what I have learned from each works to apply to my project. For the scope of this literature review, I decide to focus on three types of recommendation systems: content-based filtering, collaborative filtering, and hybrid method. I will review various works that utilize and discuss these approaches, and decide their respective pro and cons when it comes applying them to my project.

Background knowledge and tradeoffs: To build a recommendation system, background knowledge about their inner mechanisms is needed. In this section, I will discuss several papers that I have taken from in order to understand the current state of the field of recommendation systems.

Prem Melville in his article Recommender System starts by briefly talking about the importance of recommendation systems in research and commerce, he then gives detailed explanation of how each type of recommendation system work, some of the popular ways of implementing them, and their respective strength and weaknesses. Then he states some of the challenges and limitations of recommendation systems. Melville states that collaborative filtering (CF) work by obtaining user feedback in the form of ratings for items, and utilizing similarities in rating behavior amongst the users in determining how to recommend items (Melville et al., 2010). In other words, an item will be recommended to a user if other users with similar behavior also favored the item. Melville then divides CF into two sub domains: neighborhood-based or memory-based approaches and model-based approaches. For neighborhood-based CF, the

author stated that most of implementations can be summarized as assigning weight to all users with respect to similarity of the target user, selecting a number of users with highest similarity scores, and computing a prediction from weighted combination of the selected group (Melville et al., 2010). However, model-based CF is rather unique and stand out as a separate category. It tries to estimate parameters of statistical models for user rating. With latent factor and matrix factorization recently becoming the state-of-the-art methods in this field (Melville et al., 2010). According to Melville, model-based CF do not use explicit features to calculate similarities between users, instead, it tries to learn the implicit features that determine similarities by itself, using various machine learning techniques (Melville et al., 2010). This makes model-based CF very useful under circumstances where explicit ratings are not available, and the features that determine the level of user preferences have to be found out by the algorithms itself. For content-based filtering, the author states that it functions by utilizing a more detailed profile of a single user including preferences and personal information, as well as metadata about the items, for example, knowing the genres of movies, and the types of movies a particular user liked, the system could pick movies that fit the profile of the user's preference, and recommend them (Melville et al., 2010). The author further details that research about content-based filtering largely focus on recommending items that come with a lot of metadata, mostly text, such as books and movies, sine more text means more metadata. Two of the popular ways to design a content-based filtering system are treating it as an IR task, or as a classification task. The IR approach treat the user preference data as a query, and the unrated items are sorted in terms of relevancy to the query, whereas the classification approach treats the user preference as a label to content of items, using classification algorithms, the items can be mapped to a scale of rating, for example, 1 to 10. As for the hybrid approach, the author stated that any methods that combine content-based filtering and CF, for example, content-based predictions can be

applied to convert a sparse (sometimes users may not rate all the movies for example) user rating matrix to a full one, then CF is used for recommendation. Lastly, the author discusses some important challenges and limitations of recommender systems, such as sparsity mentioned above, the cold start problem, which is a problem mainly concerns CF systems, in which a newly added item is not rated, and therefore rarely recommended, and frauds that try to game a recommendation system on a website to get ahead. The author also discussed the pros and cons of CF and content-based filtering, with CF more potent in areas where there is little information associated with items, and content-based systems better at avoiding pitfalls such as the sparsity problem and the cold start problem.

In Tewari's Generating Top-N Items Recommendation Set Using Collaborative (Tewari et al., 2018,) Content Based Filtering and Rating Variance, which propose a hybrid recommendation system using collaborative filtering and content-based techniques, it is stated that content-based filtering largely focus on items that are highly similar to items that user has favored earlier. In other words, the recommendation can't reach anything that the user hasn't experienced before, therefore, it has trouble generating serendipitous recommendation. Collaborative filtering solves this problem by recommending items based on other user's activities, which can contain things that the target user has never experienced before, offering more chances of chance discovery of new types of items (Tewari et al., 2018).

In Recommendation System in E-commerce by J. Ben Schafer, the author details many examples of recommendation system on e-commerce websites, including Amazon and eBay in their earlier days. Schafer categorized different recommendation techniques into browsing, similar item, email, text comments, average rating, ordered search results, and top-N (Schafer et al .,) Then the author talks about different recommendation systems and what types of websites they are

suited for, and the author categorized them into types: non-personalized recommendations, a basic system that only takes into account of the average rating for an item across the board for all customers. Attribute-based recommendations is also a basic recommendation system where the system generate recommendation based on the user query. Item to Item correlation as described in the article is similar to content-based filtering techniques, where a user would provide input for their favored types of products, either through implicit feedback such as search history or explicit feedback such as ratings, and the recommendations are generated, based on the similarity of the attributes of the items and the attributes favored by the user. People to people correlation as described is essentially collaborative filtering, where recommendation is based on behaviors of other users that are similar to the target user.

In Hybrid Recommender System: Survey and Experiments, Robin Burke evaluates the pros and cons of various different recommendation techniques (Burke et al. 2001). And suggested several ways of combining different techniques to create hybrid recommendation systems that can get the best of both worlds: A weighed hybrid system is one in which the score of a recommended item is computed from the results of all of the available recommendation techniques present in the system (Burke et al., 2001). For example, the scores of a collaborative filtering system and a content based system can be combined together linearly, with weights attributed depending on each system's importance. A switching hybrid system uses some criterion to switch between recommendation techniques. In this article the author uses the example of Daily Learner's system where a content-based filtering system is applied first, and if the system can't make a confident recommendation, then a collaborative filtering system is applied.

These papers are beneficial to my project because they discuss how the most popular ways of designing recommendation system work, as well as the pros and cons of different approaches.

This gives me a good baseline for evaluating implementations of different approaches, and it helps me to consider what kind of methods I want to adapt to my project. For now, I think the optimal way to implement the system might be a hybrid approach that combine collaborative filtering techniques and content-based techniques, and their respective results can be combined into a final score in a linear way.

About the Methods: The examples in the literatures also helped me to device and shape my methods of implementing the recommendation system.

In Tewari's paper (Tewari et al., 2018), the recommendation system is comprised of 5 building blocks: Profile builder (PB), Similarity Finder (SF), Collaborative classifier (CC), and Item weight and variance calculator(IWV) (Tewari et al., 2018). PB establish profiles of every user and item using a set of keywords. For items, the keywords are provided by sellers on websites, and profiles of users are constructed by extracting information from user's past purchasing behaviors on the websites. SF uses cosine similarity to calculate similarity scores between users. CC generates recommendations using collaborative filtering techniques. It selects a neighborhood of very similar users to predict ratings for items that have not yet been seen by the target user, but has been seen by the other members of the neighborhood, using the equation (Tewari et al., 2018. page3.):

$$P_{mk}=A_m + \frac{\sum_{n=1}^{c}(Rn,k-An)^{*}Sim(m,n)}{\sum_{n=1}^{c}Sim(m,n)} (3)$$

With Pmk being the predicted rating of user m to item k, Rn,k denoting a Rating of user n to item k, and Sim being the similarity equation being used in the paper.

All predicted items are then classified into liked and disliked categories, taking the top of the liked list as items fit for recommendations. Lastly, IWV finds the popularity of different items among all users in the forms of weights, as well as calculating the rating variance if different items across different users. The item weights are calculated using the equation (Tewari et al., 2018. page4.):

$$IW_i = \frac{AIR_i}{Rating_{max}} \times \frac{Count_i}{Count_{max}} (5)$$

With AIRi being the average item rating of item i, Ratingmax being the largest rating in the given rating scale, Counti being the number of users who have rated item I, and Countmax being the highest value of any Counti.

FR is the block that generates final recommendations to the target user. It collects inputs from CC and IWV blocks, and generate recommendation value (FRV) using the following equation (Tewari et al., 2018. page5.):

$$FRV_i = (CC_i - SD_i) + IW_i (7)$$

The system was implemented using Java7 and the data are stored in MySql, and the effectiveness of the system is tested on live datasets.

In this article, the final recommendation is generated by combining the results of two rating predictors using different approaches, which seem to be beneficial, in the sense that it gets the best of both worlds. I think this is worthy of consideration if I decide on a hybrid approach for my project.

In Science Concierge: A fast content-based recommendation system for scientific publications by Titipat Achakulvisut, the author proposes a content-based recommendation system for academic literature and scientific publications. On to the materials and the methods of the project, the author decides to use licensed conference data rather than journey data, because conferences require short and quick presentations, so it is crucial for recommendation systems to let the scientists to discover materials fast. The design of the system itself follow 3 main design principles. First, to prevent Mathew's effect, which ties into the cold start problem from MVLL, by utilizing content-based filtering (Achakulvisut et al.,2016). Second, it aims to be especially fast due to the specific needs of researchers and scientists. And lastly, it will be validated using external input. For the representation of the documents, science literature is converted into 2-dimensional vector representation through a workflow, and the documents are processed to make the task easier, such as removing stop words and terms that do not appear frequently enough. The remaining terms are weighed using tf-idf (a type of term frequency), and latent semantic analysis is performed to reduce the noise. Popular keywords are also highlighted during the preprocessing. The recommendation is produced by the Rocchio algorithm, based on relevant and non-relevant documents voted by the user in the past. The Rocchio algorithm finds a document vector that combine the 2 sets of documents. Then, a nearest neighbor search will be performed to find suggested elements. In terms of results, the Achakulvisut finds the best

combinations of parameters for the algorithm, and found that documents that are rated non-relevant should not be used to modify the user preference vector at all. And, in comparison with other common alternatives, SC had favorable performance. The author then concludes the paper by recounting important points of the process, and talking about how the system could be expanded in various ways. This article provides an example of a content-based filtering system. The items are users are more specialized than regular movie audience, being scholars and scientific literature, and the metadata about users and items are both available, and well utilized. I think this is quite similar to my project, which specializes dealing with cars and dealers, the user profile would contain a lot of information, and the metadata about cars are abundant, such as facets used by the website itself, and user click stream data. The content-based system described in this paper seem to avoid pitfalls such as the cold-star problem, and user rating sparsity, which is another advantage of including content-based system in my project.

In Algorithm for movie recommendation system using collaborative filtering by Nisha Bhalse, the author discusses the characteristics and effectiveness of each type of recommendation systems, and proposed a model based collaborative filtering system for movie recommendation. The article details the construction of a user-item utility matrix, matrix factorization via SVD algorithm, user similarity computation process, constructing the group of similar users, and how the ratings are predicted. This article provides an example of a collaborative filtering recommendation system. This system tries to solve the matrix sparsity problem by normalizing and transforming the user rating matrix in to a dense matrix, and does not require clearly defined metadata for items or users, since it is a model-based CF system. However, in the case of my project, the user base(dealers) mostly consists of returning users, and smart auction

website uses facet search system, so there is plenty of metadata available for both items and users. Moreover, a CF movie recommendation usually require a great number of users, in this case it is due to the vast collection of movies available on a platform, and the need to construct neighborhoods of similar users. The dealer user base is much smaller than regular buyers, thus it might make sense to design a system to work for a potentially small user base.

In Hybrid collaborative filtering methods for recommending search terms to clinicians by Zhiyun Ren, the author proposes a set of methods of hybrid collaborative filtering for recommending search terms to clinicians. Ren assumed that useful search terms are related to the past search terms the clinicians have used for this patient, and the diagnosis of the patient, and used them as the base for the system. Ren also only used recent search terms as appropriate data. The author goes into the details of some terms and definitions, then he moves on to talk about the methods, which involves constructing co-occurrence matrix of ICD Code-Search Term to calculate a co-occurrence frequency, which is used to predict scores. Matrix factorization is then used since actual co-occurrences are rare and the matrix is sparse (Ren et al., 2021). Ren also introduces hybrid collaborative filtering model for healthcare (HCFMH), which involves combing two scores (one based on previous search terms, another based on previous encounters) linearly to produce a final score for recommendation. The structure of HCFMH again involves combing multiple ratings generated by different systems, which I'm interested in including, if I decide on a hybrid approach.

2. Overview:

The subject of discussion in this paper is a hybrid recommendation system of my own design, consisting of a content-based filtering subsystem and a collaborative filtering subsystem. In the following sections, I will go through the overview of the system, including the preliminary work done beforehand, data selection, exploration and processing, the methods and other details about each subsystem, and lastly some reflections on the work I have done on this system.

The original purpose of this project is to design and implement a recommendation system for a smart auction system or website for car dealerships. The system will utilize user data such explicit user feedback/profiles, and user activity and interactions with the smart auction system or website to generate recommendations of cars. The scope of this project would include the process of deciding on the techniques through researching, explorations, and experimentations, creating simulated data for the project, the implementation of the system, and some testing and evaluation if circumstances permit. The specific techniques will likely be within the area of IR and machine learning. The types of recommendation techniques I choose to focus on will be content-based filtering, collaborative filtering, and hybrid techniques of the two.

2.1 Preliminary work:

This project was inspired by some of the previous work designing and prototyping faceted search system for an online car auctioning platform during an internship. It was discussed that data relating to each facet of the car, such as make, type, color etc. as well as the website interaction data such as clicks and purchasing, can be collected. Therefore, a dataset consisting of all of the car facets and the interactions related to them can be extracted from the data. The idea was to transform the raw data into a format where each column represents a specific interaction on a car facet (ie. Clicked on a ford car). I prototyped the data by creating a sample

dataset, and performed research on recommender technics and methods, much of which is documented in the literature review section above. Due to access/security reasons, and unavailability of some of the data at the time however, I could not get the data I needed from the company that I interned with, therefore, significant changes to the data, and therefore the project, had to be made. Nonetheless, the aim and general direction of the project itself remained the same, in the sense that it would still be a hybrid recommender system.

3. Research Questions/Project Goals

1. Decide on the proper IR and machine learning techniques to use, and what type of recommendation system will this be (content-based, collaborative, or hybrid), when designing the system, through researching, explorations and experimentations.
2. Finish designing a simulated dataset based that satisfies the requirements purpose of this project.
3. Implementing the project using programming and machine learning tools.
4. Evaluating and testing the system.

4. Data selection:

During the initial stage of the project, I attempted to design a mock dataset, so the data could still be of the structure I originally planned for the project, and I could use the previous prototyping work. This would have been a huge task, for the dataset to produce good realistic results that could be analyzed and documented, the data would have to have realistic distributions, and any biases that I program into the dataset would affect the results. To address this issue to an extent, I tried various random generation methods, with some tweaking and controls. However, after a process of exploration and consulting with experts, I realized that it was a near impossible undertaking for me to construct mock data that could realistically represent different interests of millions of users in the real world. Moreover, I could not find enough literature that could help me to verify that the mock data is indeed realistic under the peculiar circumstance of the project, without available ground truth or control data.

Subsequent selecting the suitable datasets for this project took some effort, as neither the desired data and the option of creating simulated data is available. Therefore, additional steps had to be taken to look for new data that fit the purpose of this project. I had to start off by searching on the different websites for datasets that possess similar qualities with what I wanted. Some of the criteria I used to search for suitable data: must be product reviews, must include product meta data/ description, preferably include numerical ratings, preferably include behavioral data, such as mouse clicks.

Eventually the Amazon product datasets on https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews was decided to be used for this project. The repository consisted of 233.1 million reviews for products in 30 different categories in total, and metadata of those 15.5 million products. Due to the large size of the

data and my limited computing resources (one workstation), I had to choose a subset of total reviews and restrict myself to a particular sub-category of products out of the 30. I decided to use the Appliance datasets because it is relatively isolated, which can avoid some of the inter dataset correlations.

5. Data exploration and preprocessing:

I found the data to be suitable for my product, due to it satisfying most of my criteria for the dataset (contains product reviews, meta data). Additionally, it also consists of item-to-item relations such as the feature: “also buy” and “also view” in the meta dataset for products. Which could potentially factor into the collaborative filtering review, as either a criteria to rank the results, or as a supplementary method for validation.

The review data consist of the image of the product, the overall rating of the user for a product, ranging from 1 to 5. The number of upvote the review received, whether the review is verified, the review time, ID of the reviewer, the product ID, the style of the product, reviewer’s name, review text, the summary of the review.

The product meta data consist of the product ID, the title of the product store page, the listed features of the products, the images of the product, other products people who bought this product also bought/viewed, sales rank of the product. The brand of the product, and the category of the product.

To preprocess the data, to start off, I removed features that will clearly not provide much benefit such as review time and reviewer names of the dataset. The raw data also consisted a lot of embedded html code and unnecessary text, I had to remove the junk data and transformed the useful data so they could be worked on more easily. Another peculiarity of the data is that, more than 60% of the users had only one or two reviews recorded in the dataset. This can potentially negatively impact the collaborative filtering system, as it can be hard to find patterns in the data if there are only one rating per user. However, the number of users who had multiple ratings are still significant, thus, I decided to leave the data as it is for now.

6. Content-based filtering

6.1 Overall: The current way of implementing this sub system is through a custom designed method that measure the cosine similarity/distance of every product against a user profile constructed through past user ratings. For the purpose of this system, I only needed the user numerical rating from the review dataset. For the meta-dataset I removed more unnecessary features from the dataset itself.

A content-based filtering approach for me involves constructing user profiles for target users based on past behavior, and using it to match the characteristics of the products. For example, a user who lives in a desert environment and who bought many SUVs in the past, will get a lot of SUVs and other cars that operate well in difficult terrains. One of the reasons behind this temporary choice is that the cars on the auction website should have a good number of metadata, which is good for content-based filtering. Also, unlike regular users, dealers belong to a smaller base, but they tend to make purchases on the website multiple times, and a user profile will benefit them greatly, which means there should be a lot of data about a single user, but the number of users is not too great, therefore, content-based filtering is a suitable choice, as it only relies on the behavior of a single user, and sufficient metadata on the product itself. Moreover, content-based filtering is better at avoiding various pit-falls such as the cold-star problem and the sparsity of user ratings.

The cosine distance function in scipy package measures the inverse of cosine similarity of two arrays or vectors, with an additional weight vector as optional parameter. I constructed the user profile for a fictional target user who have multiple review histories. The user profile is essentially a list of scores for all of the categories based on the user's history. I then extract appearance/count vector for every product, in the form binary lists, the product can either be of

or not of a particular category (represented as an index in the list). Lastly, I measure the inverse of cosine similarity between the count vector and a constant, control vector, using the user profile vector as weight. Therefore, the system will detect whether a particular product belongs to a category, and add the corresponding "weight value" to the overall cosine distance score. As final results of the content-based recommendation subsystem, the products are sorted by the cosine similarity score, and the ranking was adjusted according to the product ranking in their respective category according to information in the dataset.

Since the system does not involve train and test data split, or any machine learning techniques, I did not perform traditional validations on the results. Thus, I only verified that the system does perform the function of assigning scores based on the similarities between user history and products. If it were possible, user testing would be the best way to truly validate the system, however, this was not possible at the time.

6.2 LDA topic modeling:

In this stage of the project, I began to look for additional features to strengthen the existing product meta data. However, the dataset does not exist more explicit meta data on the products themselves that can be considered as categories. Therefore, I attempted at finding implicit features that can be used as additional categories for content-based filtering.

Topic modeling techniques treat the documents as a bag of words, and extract various topics from the documents in the form of prominent word lists. The dataset contained relatively long text descriptions for the products as columns, and if these columns contain consistent patterns, topic modeling could potentially extract "topics" from the texts that can be used as additional

features to strengthen the existing categories. After some research I chose LDA topic modeling package for NLTK in python as the tool.

I fed the transformed description text into the LDA model, and adjusted the parameters, particularly the number of words in each topic, and the number of topics according to the results of each iteration. However, the results never really approached what I was looking for.

LDA topic modeling failed to produce topic lists that can consistently apply to the dataset overall, and therefore was unable to be used to extract categories or features to add to the filtering. This is partly due to the fact that the dataset itself did not have consistent enough format of description for the products. Also, the LDA modeling returned too many words that are entirely unrelated to the product categories but consistent among products with the same format of description, furthermore, when it did return words that are common amongst all types of descriptions across the board, they often serve similar functions as stop words in the description, therefore, significant amount of filtering is needed for it to possibly work for this project, in the end I had to abandon the method.

7. Collaborative- filtering

7.1 Overall: Likely an algorithm that performs latent analysis, (matrix factorization)

Collaborative filtering (CF) involves matching the target user with other users who have similar behaviors, and recommending to the target user something that is liked by the other users. For this project, I will be using a model based collaborative filtering algorithm. Mostly, the features that model based CF algorithms use are latent, as in learnt by the algorithm from patterns buried in the data. This can resolve the potential lack of serendipitous discoveries that come with a content-based approach. A well-designed hybrid approach can potentially get the best of both worlds, hence why I am including both content-based and collaborative filtering methods in my system.

Theoretically, CF techniques are more complicated to implement if it is done from the ground up since it involves more complicated algorithms such as matrix factorization, thus I will be using a prepackaged implementation of a CF algorithm. CF based systems work by grouping users that have similar behaviors and profiles together, and recommend items that have not yet been browsed by the target user, but was favored by other similar users. In other words, a user will be recommended items that are liked by people with similar interest and behaviors. There are various types of collaborative filtering methods. For example, if user A behaves similar to user B, but did not browse item X, which was liked by user B in the past, item X is likely to be recommended to user A. This will eliminate the need to artificially define how each feature contribute to a prediction, since it will be learned by the algorithm, therefore would probably result in a more accurate and generalizable model. For example, I may overlook user zip-code as a feature, and assign it a low weight of contribution to a positive “rating”, when in reality, it may

be a significant contributor for whether a user like a particular type of car, and this will be captured by a CF system, as it only cares about user similarity, and past user behaviors.

7.2 Methods selection:

In the context of recommendation systems, matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. The user-item interaction matrices are usually sparse due to users not being able to rate every product, the algorithm fill in the missing values in the interaction matrix by learning the values of 2 lower dimension matrices, so they correspond well enough with the original interaction matrices. The 2 lower dimension matrices are then re-composed together to obtain a "full" matrix. The latent features selection, or how the algorithms learn the values of the lower dimension matrices in general varies from iteration to iteration. For example, the famous Netflix challenge used gradient descent to achieve this task. For the validation, I deployed a confusion matrix on the test data, and chose to not use the also buy and also view columns due the large amount of none values within them.

For this subsystem, I used the Knime Spark big data package which consists of collaborative filtering learner nodes. They contain repackaged matrix factorization algorithms that predict user ratings, which are suitable for the purpose of the project. I preprocessed the data by getting rid of users that only have a single rating, and changed the string ids into new, matching integer ids. I used 80-20 split between training and testing data, and devised a work flow consisting of 2 main pipelines, one for training and validating the model, and another that

create a target user profile by using a column editor (create user input ratings), and deploy the system for the target user.

8. Reflection/discussion:

As touched upon in previous sections, the final system is significantly different from what I originally planned, due to a variety of reasons. Chiefly, the unavailability of the car auction data had the greatest impact on the project, necessitating a whole new process of exploring new options for replacement. The failure of attempting to generate simulated data meant more time was lost, and a new replacement dataset that might not satisfy all my expectations is required. Indeed, the final dataset I used for this project did not contain the behavioral data (for example, mouse clicks and purchases) I wanted, and was in a different format in general. This complication resulted in a reimagining of the content-based filtering system, as new features was explored and extracted for usage, and the selection of the final method used as impacted. Strictly speaking, it could be argued that the final project was an entirely different object than what was planned in the beginning, in terms of details.

During the process of designing and implementing the hybrid recommendation system, several important lessons have been learnt by me. To start off, the direction of the project was impacted dramatically by the unavailability of the desired data, thus necessitating additional steps of searching for replacement and rendering a lot of the preliminary work useless. From this experience, I understand that projects can change drastically due to external factors, and things will not always go as planned, therefore I should always be ready for changes that can happen anytime and prepare to adapt for that. Another lesson I learned from this experience is that more often than not, it is better to give up on a method/approach that is failing instead of keep trying to make it work. For example, a significant investment of time and effort was put into the failed attempt at exploring the possibility of constructing simulated data because I did not want to give up my preliminary works, which in the end hurt the project. Later into the

project, I encountered another situation where the approach I was taking wasn't working when I experimented with extracting features from the results of LDA topic modeling. I had to abandon the method entirely and move on after the results of my experiments showed that it was unlikely to work for this project. This notion also relates to dealing with new information/ideas that have emerged half way through a project, balancing between shifting to superior approaches and carrying out the original plan can be quite important.

As for possible future works, some of the directions I can take this project further are: Cross dataset recommendation, since purchasing in one department might imply preferences in another, or deploying my system on the original datasets if they become available in the future.

In conclusion. For this project, I designed and implemented a hybrid recommender system originally geared towards car auctioning data. Due to various circumstances, the amazon product review and metadata were used. The system is consisted of a content-based filtering sub system and a collaborative filtering subsystem. Several failed approaches of transforming the data and extracting additional features were documented. As, for the final system, for the content-based filtering subsystem, an algorithm that measures the cosine similarity between product vectors and the target user profile vector was designed, not involving any machine learning. For the collaborative filtering subsystem, I used a prepackaged matrix factorization algorithm in Knime that works by predicting user ratings in the sparse interaction matrix.

Link to source code: https://github.com/R-Xu123/Recommender-project

Reference List

Ren, Z et al., (2021). Hybrid collaborative filtering methods for recommending search terms to clinicians. *Journal of Biomedical Informatics*, *Volume113.* https://doi.org/10.1016/j.jbi.2020.103635.

Tewari, A et al., (2018). Generating Top-N Items Recommendation Set Using Collaborative, Content Based Filtering and Rating Variance. *Procedia Computer Science. 132 (2018) 1678-1684.*

Bhalse, N et al., (2021). Algorithm for movie recommendation system using collaborative filtering. *Materials Today: proceedings.*

Achakulvisut, T et al., (2016). Science Concierge: A fast content-based recommendation system for scientific publications. *PLOS ONE*. https://doi.org/10.1371/journal.pone.0158423.

Schafer, B et al., Recommendation system in E-Commerce. *GroupLens Research Project.* http://files.grouplens.org/papers/ec-99.pdf.

Burke, R et al., (2001). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction 12: 331-370, 2002.*

Melville, P et al., (2010). Recommender Systems. *Encyclopedia of Machine Learning*.