Final Project: Romance Book Recommendation System

Medha Karthik
Department of Computer Science, UNC Charlotte

ITSC-3162-002: Introduction to Data Mining

May 8, 2024

I. Introduction

Problem and Overall Goals

As an avid reader and enjoyer of romance books, I often find it difficult to find books I end up enjoying reading while using current, popular recommendation systems. In fact, I often find many of my least favorite books this way. The goal of this project is to create a personalized book recommendation system that helps me find books that are more curated to my taste. Although this project will use a model built on my personal tastes, I plan to create a method that can be replicated by other users to fit their own specific needs. This project will outline the creation of a content based recommendation system.

Why Romance?

Many book recommendation projects focus on a large range of genres. I chose to focus solely on romance books for this project in order to explore the diversity within the romance genre. Additionally, I have a much greater idea of my tastes within romance compared to all other genres. This would allow me to have a much more informed evaluation of the model.

The romance genre is the largest selling book genre in the US and has been for many years. As such, the applications of this project could be beneficial to many people as well as the publishing industry.

II. About the Data

Where the Data is From

For this project, I used data from an existing source. I used the ‘Romance Books Dataset’ found here on kaggle, uploaded by user ‘The Devastator’. This dataset contains 1246 romance books from numerous authors and years. The dataset contains the following 8 features: ‘ID’, ‘title’, ‘author’, ‘release year’, ‘synopsis’, ‘book length’ (number of pages), ‘rating’, and

‘number of ratings’. I also scrapped additional data for each book from Goodreads. I added two columns, ‘genre’ and ‘audience’.

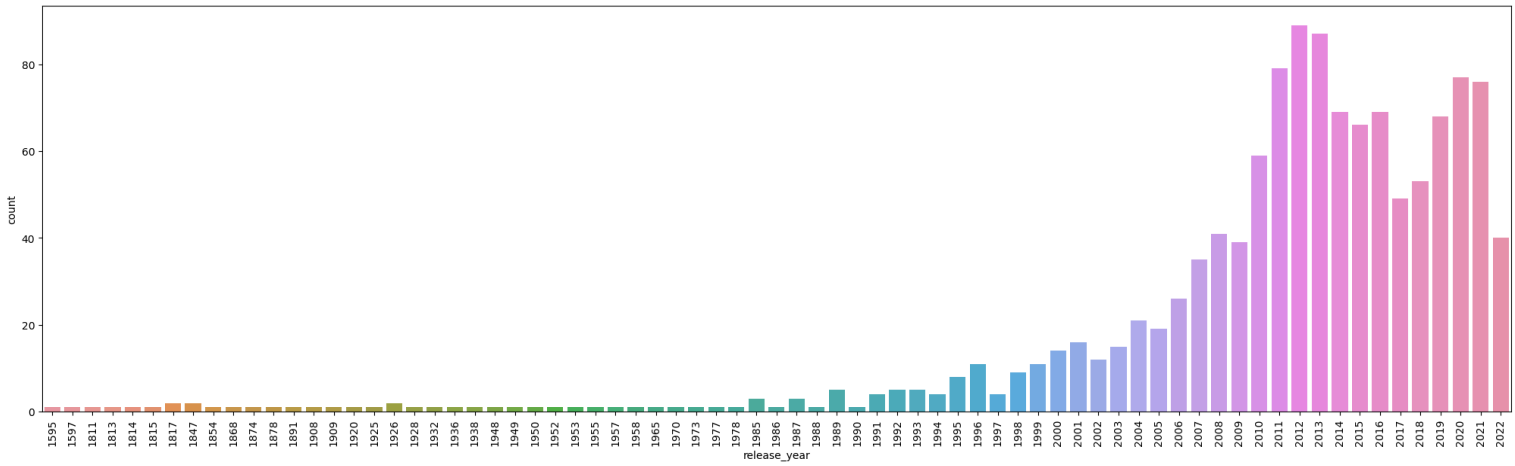
- ‘genre’ - refers to the subgenre each book is part of within romance, such as contemporary, historical, comedy, fantasy, or paranormal
- ‘audience’ - refers to who the book is marketed towards. For this project, the options were either ‘adult’, ‘young adult’, or ‘new adult’.

Goodreads lists these features on each book’s page, making the scrapping process routine and precise.

Finally, in order to make the recommendations more personalized to my taste, I adjusted the ‘rating’ feature for the books on the list that I have read. Of these total books, I have read 171 of them. This provides a good balance between books I had never read before and books that I could add my personal rating for. I averaged the included community rating with my personal rating (collected from my own book dataset), with the hopes that it will skew the ratings to more reflect my tastes.

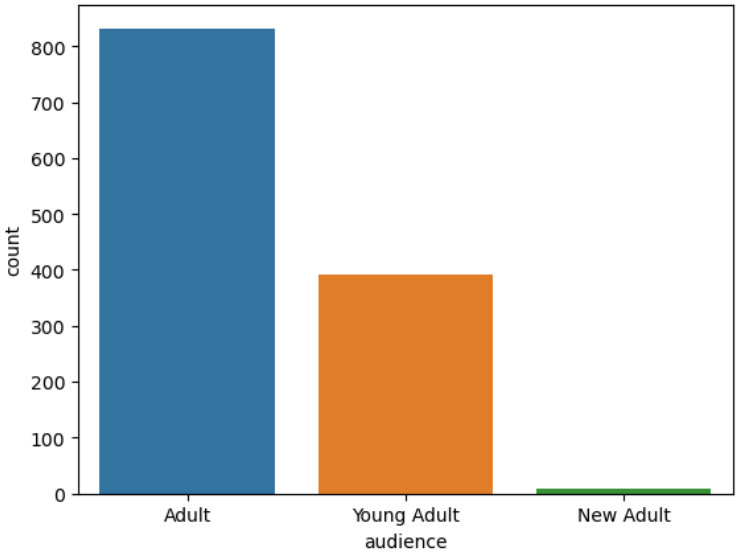
Data Visualization

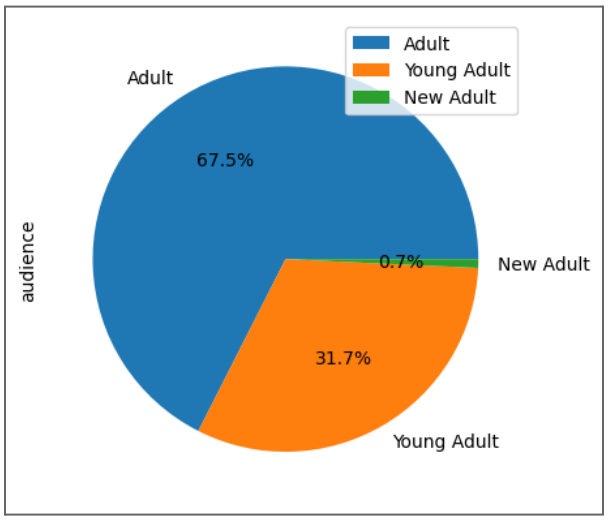
Years:



This image shows the spread of the release year for each book. Most of the books were published fairly recently, with the most books published in 2012. The years range from 1595 to 2022. The distribution is skewed left.

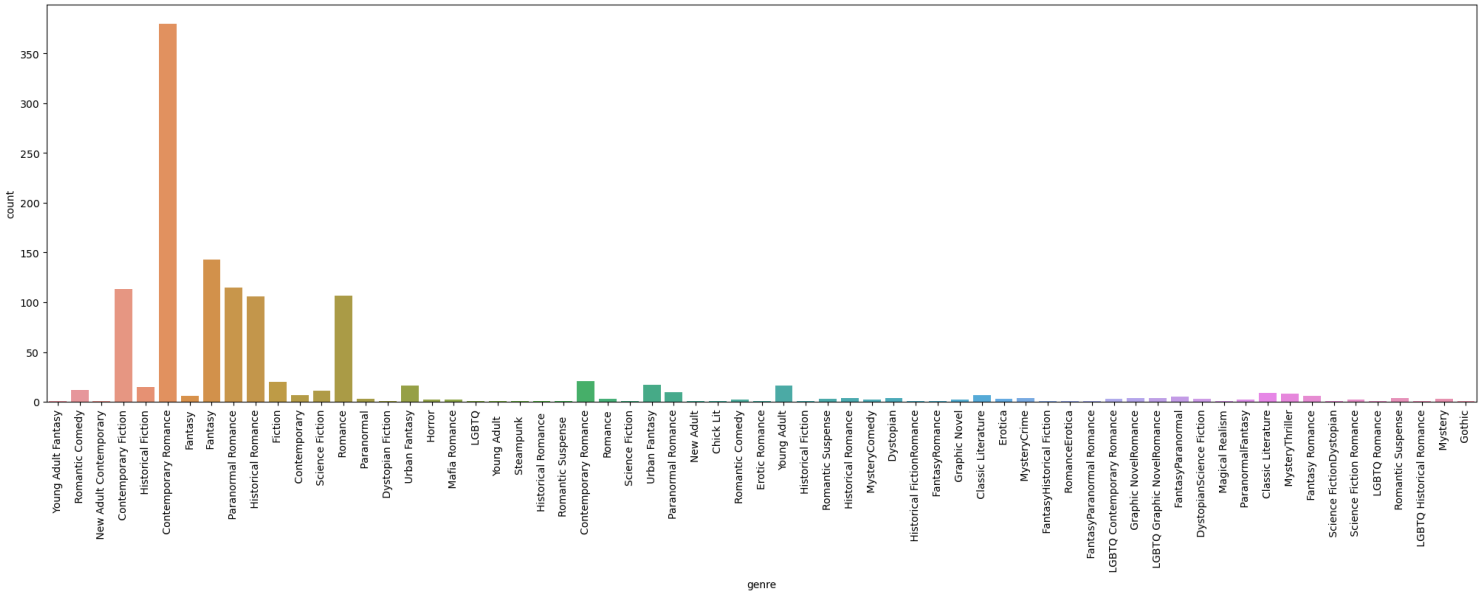
Audience:





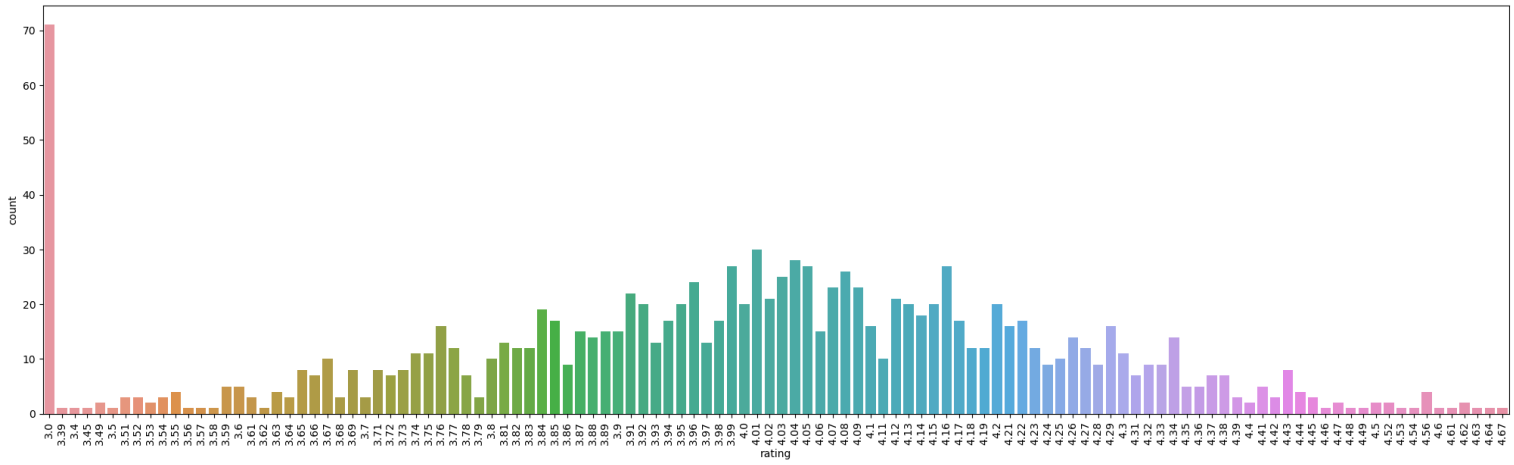
This image shows the counts for the 'audience' column. Most of the books (67.5%) are categorized as 'adult', 31.7% of the books are categorized as 'young adult', and just 0.7% of the books are categorized as 'new adult'.

Genres:



This image shows the various genres represented. The most common genre is ‘contemporary romance’, followed by ‘fantasy’ then ‘paranormal romance’. There are 64 total genres.

Ratings:



This image shows the various ratings for all books. The ratings range from ‘3.’0 to ‘4.67’. Most of the books are labeled as ‘3.0’.

III. Methods

Preprocessing

The first step in preprocessing the data was to remove any null values. Then, duplicated entries were deleted. This reduced the number of books down to 1232. Third, unnecessary features were removed. This included the existing ‘read’ column, ‘synopsis’, and an unnamed column that was added unintentionally when transferring the data. The ‘read’ column was not originally part of the dataset, it was added as a method to track which books I had already read and so were removed before modeling.

Then, the values in ‘book_length, ‘title’, and ‘genre’ were adjusted. The ‘book_length’ column contained entries that included the word ‘pages’, an example being ‘362 pages’. The word ‘pages’ was removed from all entries using the re (Regular expression operations) python module so that only digits were included. Additionally during this step, the column was renamed

as ‘pages’ for clarity. The ‘title’ column contained unimportant information after the title within parentheses, such as the edition or format (hardcover, kindle, paperback) of the book. This information was removed also using the re python module and added to a new ‘adjusted_title’ column. Finally, a few of the entries in the ‘genre’ column incorrectly contained numbers. These were removed again using the re python module.

The last step before modeling was to trim the ending whitespace from relevant columns to prevent errors.

Creating the Model

For this project, a content based recommendation system was created. Here is a step by step guide on how the model was created:

1. A new dataframe named ‘content_df’ was made with the chosen features: 'adjusted_title', 'author', ‘release_year', 'audience', 'genre', 'rating', 'number_of_ratings', and 'pages'.
2. All of the information from the columns were joined together in a new column, ‘Content’, as string types and separated by spaces.
3. A TF-IDF (term frequency-inverse document frequency) vectorizer was used to convert the ‘Content’ into a matrix of TF-IDF features. This process transforms and finds the most important words in the column and saves it in a new variable, ‘content_matrix’.
4. A cosine similarity matrix was created using a linear kernel with the ‘content_matrix’ column as both inputs. This resulted in a measure of the similarity between all books arranged in a new matrix, ‘content_similarity’.
5. The function ‘get_recommendations()’ was created:
    a. The function first asks for the user to input a book title and how many recommendations they would like.
    b. The index of the inputted book is found.
    c. The similarity scores are calculated by looking up the index in the ‘content_similarity’ matrix. This step simply finds all of the books similarity in relation to the inputted book.
    d. The top however many the user requests books are sorted and stored into a ‘top_similarities’ variable by the book’s index.

e. The title, author, and similarity score are returned to the user. Recommendations are found by searching the dataframe for the ‘top_similarities’ index values.

6. The user can call the function using ‘get_recommendations()’. They will then be asked to input a title and how many recommendations they would like.

Here is an example of the output from calling ‘get_recommendations()’ with the title ‘Pride and Prejudice’ and requesting 5 recommended books:

```
#call recommendation function!
get_recommendations()
```

```
Enter a book title: Pride and Prejudice
Enter how many recommendations you would like: 5
1 . Pride, Prejudice, and Other Flavors by Sonali Dev , similarity =  0.005458154040981423
2 . Sense and Sensibility by Jane Austen , similarity =  0.0026976664412964924
3 . Emma by Jane Austen , similarity =  0.015657883109071413
4 . The Proposal by Jane Austen , similarity =  0.01274693783884338
5 . Eligible: A Modern Retelling of Pride & Prejudice by Curtis Sittenfeld , similarity =  0.04390543653542314
```

Iterations

It took many iterations to get a working model. The initial plan was to use a neural network and Word2Vec instead of TF-IDF. Word2Vec contains many improvements on TF-IDF because it takes into account any semantic relationships between words, such as context and hidden meaning. This differs from TF-IDF, which only takes into account the frequency of words. Word2Vec is much more powerful and was used in many similar recommendation systems to this one, making it the first choice for this project. However, the actual implementation was not very successful for this project. Due to the relatively small size of the database, this implementation caused all the recommendations for books to be the exact same. As such, I chose to use a simpler model incorporating TF-IDF to hopefully resolve this issue.

Once I began to experiment with TF-IDF, I again ran into many issues, most notably with indexing. The model appeared to be working, however, it was continually outputting the recommendations for the very first index. I then realized that the original titles included many whitespaces and unnecessary information, which made the model difficult for the user to use. This led to the creation of the adjusted_title column, as described in the preprocessing section above. After this change, the model drastically improved.

Another issue due to indexing resulted from dropping repeated titles. The index values remained the same for each title, resulting in missing index values during the modeling step. Once I realized this, I was able to reset the index, solving this problem.

IV. Evaluation

In order to evaluate the performance of the model, I chose to read a book recommended to me. Several of my favorite romance books are in the dataset, so I simply entered the title into the model to find similar books. The book I chose to enter was *10 Blind Dates* by Ashley Elston. This is one of my favorite romance books and so would be a good way to judge the success of the model. This is the output of the model after inputting my chosen book:

```
#call recommendation function!
get_recommendations()
```

```
Enter a book title: 10 Blind Dates
Enter how many recommendations you would like: 10
1 . The Proposition by Katie Ashley , similarity =  0.9999999999999999
2 . Breathe by Kristen Ashley , similarity =  0.045069772009528816
3 . Ravished by Amanda Quick , similarity =  0.011284379959761889
4 . The Wedding Party by Jasmine Guillory , similarity =  0.005630212209520558
5 . Ruin and Rising by Leigh Bardugo , similarity =  0.004999049059181171
6 . Love Unscripted by Tina Reber , similarity =  0.005367843491549479
7 . Royal Holiday by Jasmine Guillory , similarity =  0.04168419336124113
8 . Things You Save in a Fire by Katherine Center , similarity =  0.03765149471784258
9 . Never Never by Colleen Hoover , similarity =  0.039209688522996486
10 . Own the Wind by Kristen Ashley , similarity =  0.028308460862558036
```

The Proposition by Katie Ashley was the top recommended book, with an extremely large similarity score of 0.999. I had never heard of this book or author before and I read this book without looking at any further details, such as the genre, synopsis, or number of pages.

Unfortunately, this book was not to my preference. I found both the writing and the plot to be very weak and rated it 1.0 out of 5.0. I also found it to be vastly different in content to '10 Blind Dates'. This was especially surprising as the similarity score was so high. I initially planned on reading three recommended books using this model, however, my disappointment after reading this book revealed that the model required additional fine tuning in order to be effective. As such, I was not able to answer the question from the introduction, namely create a personalized book recommendation system that helps me find books that are more curated to my taste.

After reading this book, I chose to look through popular book recommendation systems to see if this book was recommended to me on other platforms. Currently, I use the recommendation systems found on Google Play Books, Goodreads, and hoopla to find books. I have been using these systems for multiple years and find them to be very hit or miss on books I like. *The Proposition* was not recommended to me on any of these sites. Although this is not an additional indicator on the failure of this model, it is an interesting point of note.

V. Storytelling and Conclusion

Insights Gained

Through this project, I was able to gain insight into the complications behind creating recommendation systems. I ran into many problems while creating the model and the end product was still not successful. I was not able to solve the problem of finding more books I actually liked instead of despising. I was not able to obtain my initial goal of creating a personalized book recommendation system that helps me find books that are more curated to my taste. I was able to create a content based recommendation system that other readers can adjust to fit their needs simply by adjusting the 'ratings' feature.

```
In [42]: content_df[content_df['adjusted_title'] == '10 Blind Dates'].Content

Out[42]: 0    10 Blind Dates Ashley Elston 2019 Adult  Young Adult Fantasy 4.02 20195 336
         Name: Content, dtype: object
```

```
In [43]: content_df[content_df['adjusted_title'] == 'The Proposition'].Content

Out[43]: 1014    The Proposition Katie Ashley 2012 Young Adult  Fantasy 4.02 50566 308
         Name: Content, dtype: object
```

Future Steps/Things to be Improved

After the failure of the model, I looked deeper into the factors that made up the similarity scores. This image shows the 'Content' column for both *10 Blind Dates* and *The Proposition*:

From these columns, it is clear why these books were considered so similar. Both books are categorized as young adult and fantasy with the exact same rating and similar number of pages. An interesting point is that the first name of the author of *10 Blind Dates* is the same as the last name of the author of *The Proposition*. This factor would have increased the similarity score while not directly relating to the content behind the book. For the future of this model, the author names could be listed together instead of as separate words. This would allow the similarity score to be influenced by the same author instead of just the author's first or last name.

Additionally, the small size and spread of the dataset diminished the usefulness of the model. As there were only slightly more than 1200 books present, similarities between the books could be calculated as very high while not being all that similar. As most of the books were published from 2010 until 2021, the books not published in this range were not numerous or diverse. As such, recommendations for older books were unlikely to be very accurate due to the small sample size. Another limitation of a feature came with the 'genre' column. An overwhelming majority of books were categorized as 'contemporary romance', which again diminished the likelihood of finding an accurate recommendation. A third column that was limiting was the 'rating' column. Since this column only ranged from 3.0 to just under 5.0, there was not a large difference between books. Using a larger dataset, with books published in many different years, with many different genres, and with a larger range of ratings (perhaps from 1.0 to 5.0) would have improved the performance of this model.

The failure of this model revealed how the features I chose to analyze for this project were ultimately very poor at recommending a book I would like. This model may be improved by removing some of the features entirely. After I finished *The Proposition*, I went back to read the synopsis. I realized that if I had read the synopsis before reading the book, I would not have read the book. This revealed that the synopsis of a book could be a good indicator of a book I would like. For the future of this model, I would add a 'synopsis' column to the features analyzed. Additionally, I would place more weight on certain features instead of all of them having equal importance. For example, two books having the same genre should be more likely to be recommended together over two books having the same rating. This would hopefully help the model make better recommendations.

Another way to improve this model would be to replace the 'rating' column entirely with my own ratings for the books I had read. I choose to use the average of my rating with the initial

community rating for this project, with the fear that if I replaced the ratings entirely with my own, the recommended books would only be those I rated very highly. However, after creating this model, I have since realized that the ratings are only one small part in the overall calculation of similarity, so completely overriding the 'ratings' column with my own ratings would not negatively impact the performance of the model. Making this change would help the model make recommendations better suited to my own preferences.

An additional way to improve this model would be to use a training and testing dataset. I currently have a dataset with all of the books I have read in the last four years, complete with titles, authors, pages, year published, genre, and my own rating. For the future, I would like to use my personal dataset as a training dataset for the model, then use the 'Romance Books Dataset' used for this project as a testing dataset. Doing this would also help the model make recommendations more in line with my own tastes.

A final way to improve this model would be to use a hybrid recommendation system, made from both content filtering and collaborative filtering. A collaborative recommendation system would bring in user specific data, making recommendations based specifically on the user's previous ratings. This would help the model make more informed decisions for users and would additionally make the model easier for users to customize. As most current book recommendation sites use this approach, it may be a more accurate recommendation system.

What I learned

Through this project, I have learned the difficulty of creating recommendation systems and the limitations of using only content based filtering. However, the failure of this project has only made me more excited to experiment with new techniques and implement the changes I described above to hopefully make a better recommendation system. I have also learned a lot about the potential negative impacts of recommendation systems, making me a more informed citizen.

Through this class, I have learned about many different data mining techniques, including classification, regression, and clustering. I have greatly improved my python programming, specifically in data visualizations. I have also improved my written communication skills. Writing reports at the end of each project has been very fulfilling and I have learned how to be

concise and descriptive during technical writing. I will continue to use the skills I learned in this class in my future.

VI. Impact

Although this project is not very successful in its current form, a perfected model would have many impacts, both positive and negative. A positive impact of this project would be the potential it has to improve the publishing industry's economy. As mentioned in the introduction, the romance book genre is very large and is a cornerstone of the publishing industry. A project like this one would give readers better recommendations for books, making them more likely to purchase those books, thus boosting the publishing industry. Additionally, this project would have a great impact socially on all romance readers. Having a curated recommendation system solely focused on romance would help to boost the prominence of the genre and aid in a reader's search for books. The method used in this project can also be replicated easily for other genres, making it customizable and applicable to all readers.

A possible negative impact of this project is related to the cold start problem, which occurs when recommendation systems do not have sufficient information to make inferences for specific books. For example, if a new book was published this year, it would likely not have as many ratings as most other books in the database. As such, it would mostly not be recommended to users even if other factors were similar. This problem is lessened due to the content based system for recommendations and the presence of other features for comparison, but still exists as a hurdle for newly published books. It would also negatively impact debut authors who do not have a large fanbase that can boost their ratings, making it more difficult for their books to be promoted.

Another possible negative impact is the potential for the most popular and similar books to continually be recommended, while books that have fewer number of ratings, are in unpopular genres, or have an uncharacteristically large or small number of pages would all face difficulties being recommended due to their relative spareness. In the context of book recommendations, this would result in books that are different from the majority of books not being recommended at all, which is a large problem as it promotes a lack of artistic creativity. Authors may then be more

inclined to write books similar to what is popular as a means of being promoted through recommendation systems, resulting in less diversity within the romance genre.

VII. Code

Notebook code can be found here. I have also attached my notebook file in the submission post. Data used can be found here.

II. References

Devastator, T. (2022, October 16). *Romance books dataset*. Kaggle. https://www.kaggle.com/datasets/thedevastator/romance-books-a-must-read-for-book-lovers

Gaurav, P. (2023, March 16). *Step by step content-based recommendation system*. Medium. https://medium.com/@prateekgaurav/step-by-step-content-based-recommendation-system-823bbfd0541c

Goodreads. (n.d.-a). *Meet your next favorite book*. Goodreads. https://www.goodreads.com/?ref=nav_hom

Goodreads. (n.d.-b). *The proposition (the Proposition, #1)*. Goodreads. https://www.goodreads.com/book/show/55819363-the-proposition?from_search=true&from_srp=true&qid=bv2bv1YG8c&rank=1

hariom081. (2024, April 5). *Book recommendation*. Kaggle. https://www.kaggle.com/code/hariom081/book-recommendation

hoshi7. (2019, August 5). *Goodreads: Analysis and recommending books*. Kaggle. https://www.kaggle.com/code/hoshi7/goodreads-analysis-and-recommending-books/notebook

How do you remove parentheses and the text within them from a string?. Quora. (n.d.). https://www.quora.com/How-do-you-remove-parentheses-and-the-text-within-them-from-a-string

Kharwal, A. (2023, June 5). *Hybrid recommendation system using python: Aman Kharwal*. thecleverprogrammer. https://thecleverprogrammer.com/2023/06/05/hybrid-recommendation-system-using-python/

Klein, P. (2022, June 11). *The cold start problem: How to start and scale network effects by Andrew Chen*. Medium. https://medium.com/twosapp/the-cold-start-problem-how-to-start-and-scale-network-effects-by-andrew-chen-813f0668c70f

Midouazerty. (2021, April 3). *Book recommendation system with machine learning*. Kaggle. https://www.kaggle.com/code/midouazerty/book-recommendation-system-with-machine-learning/notebook

Pandas.dataframe.drop_duplicates#. pandas.DataFrame.drop_duplicates - pandas 2.2.2 documentation. (n.d.). https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html

Pandas.DataFrame.index#. pandas.DataFrame.index - pandas 2.2.2 documentation. (n.d.). https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.index.html

Pandas.dataframe.rename#. pandas.DataFrame.rename - pandas 2.2.2 documentation. (n.d.). https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rename.html

RE - regular expression operations. Python documentation. (n.d.). https://docs.python.org/3/library/re.html

sankha1998. (2020, September 10). *Collaborative Book Recommendation System*. Kaggle. https://www.kaggle.com/code/sankha1998/collaborative-book-recommendation-system