Machine Learning-Based Book Recommendation System for Students

Serena Ding, Ankit Gupta, Vikram Mishra and Rong Cao

ABSTRACT

Recommender systems are widely incorporated and used today in many fields. The purpose of this study is to demonstrate the development of book recommendation systems through the usage of Artificial Intelligence (AI). Models of recommender systems displayed in this study are based on popularity, correlation, and content. One system investigated in this study is the Lexile recommender, which suggests books based on Lexile levels, measures of how difficult a text is or how advanced a reader is. For the dataset, the researchers used a sample of books frequently taught and read in high school and user information from the public site Goodreads. Though the target audience for these books in the sample was high school students, the study can be applied to elementary and middle school students as well. As prior research shows that there is no single best way to create reading lists for students, the results of this study would encourage both recreational and educational reading and allow readers to create personalized reading lists of their own. The researchers used Python libraries to implement these recommender structures. The results of this study showed that creating recommendation systems through AI was successful, but there is still much room for improvement in the complexity of these structures. For future studies, researchers can make improvements on the types of recommenders used in this study or use a recommender structure other than those in this study.

KEYWORDS

Python, Pearson correlation, recommendation system, content-based, correlation-based, popularity-based, machine learning, artificial intelligence

1. Introduction

Artificial intelligence (AI) is used today to create machines that have capabilities of human intelligence. This study uses AI to create and implement recommendation systems that are a part of a large system called information filtering, which deals with filtering data to deliver the most relevant and useful information to users. Fayyad, Piatetsky-Shapiro, and Smyth [1] describe this further, explaining how these "systems ask the user to specify a profile of interest and search for related information among a wide variety of… sources" (p. 39). This study explores one type of these systems—a book recommendation system that analyzes data to provide personalized feedback.

Machine Learning (ML) is a subfield within Artificial Intelligence that focuses on training a machine to learn from a pattern of specific data. One type of Machine Learning structure utilized in this research article was the decision tree. The decision tree is based off of the segments between multiple nodes, which form a tree. Decision Tree branches, which contain several segments, can be formed dependent on data given to the machine learning model.

Statement of the Problem

According to Gorman [2], oftentimes students see the "summer reading list as a chore" (p. 52). This could be because they dislike the book options, especially since they have no choice in what to read, because the reading level is not right for them, or for a variety/combination of other reasons. Gorman also suggests that there is no single best way to create school reading lists because there is little data on the topic (p. 54). The more data there is available about any subject, the better one can identify trends and improve results. This issue is key in conducting this study.

Significance of the Study

A personalized recommendation system can be useful in creating reading lists, especially for students. As supported by Gorman, reading lists are extremely helpful in encouraging recreational reading and providing educational benefits (p. 56). This study particularly benefits students by creating reading lists that build around their reading levels. The data collected can also help language arts teachers construct curriculums, as they can easily see common trends in books (such as which books are most popular among readers) and to decide which books and topics to study in the future. The research can also provide more data online for those who may need it and adds to existing book recommendation systems by introducing new ideas such as the Lexile recommender.

2. Methodology

As the researchers had no prior experience or knowledge of recommendation systems, they relied on past projects found online relating to the topic. They built their research off of a project that created a system of recommenders in Python. The project included recommendation systems based on popularity, correlation, and content. The researchers also added a recommender that was based off of Lexile scores of the books. After implementing these recommendation systems, they gathered the results and used the Python library Matplotlib to create graphs of the data.

Sample

The researchers manually gathered information about books for elementary, middle, and high school students. For their data collection, the dataset consisted of fifty books from a Goodreads list titled "Required Reading in High School." The researchers also collected information on users who had rated the books in the dataset. They collected data from twenty three popular users—which books from the list they rated and what rating out of 5 stars they gave the book. They collected 412 ratings overall. The researchers assumed that this was enough to determine trends. The researchers extracted all of this information from Goodreads.

Materials/Instruments

To run code and produce results, the researchers used Jupyter Notebook, a web application that allows users to create and save documents of code. The researchers installed Jupyter through Anaconda, a large and popular distribution of programming languages. Through Anaconda, the researchers were also able to access and use pandas, NumPy, Scikit-learn, and Matplotlib, all Python libraries that are highly useful in data science.

2.1. Implementation

The whole project is implemented as shown in Fig. 1, beginning from data collection, and then going on to analyzing and evaluating the population-based, correlation-based, content-based, and Lexile methods accordingly.

Popularity-based recommender

Using the pandas and NumPy, the researchers ordered the books by descending number of ratings and created a table for the top five most rated books. The researchers then created a table for the top highest ranked books. Through

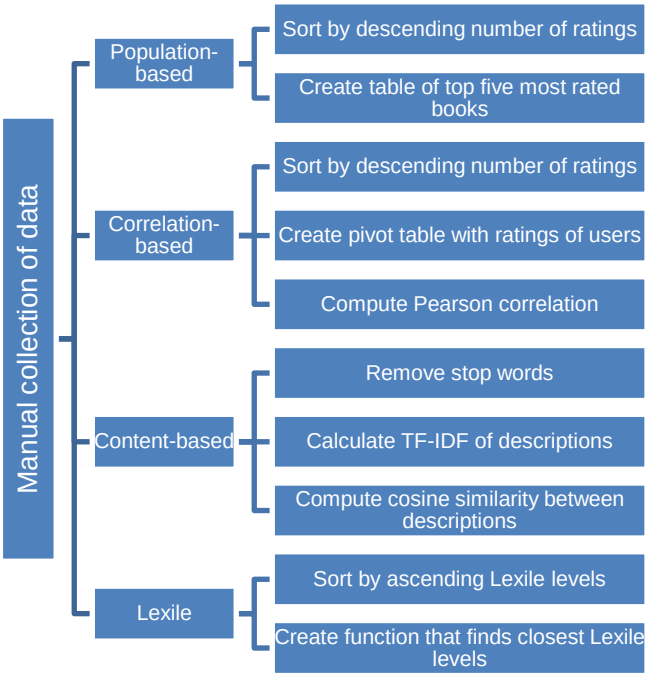

Figure 1: *Implementation of the whole project*

pandas, they could also see the statistics of the dataset, such as the max, min, mean, etc. This was useful to see which books had the highest number and value of rankings, as well as which author came up the most in the list. This type of data is ordinal because the order was based on popularity.

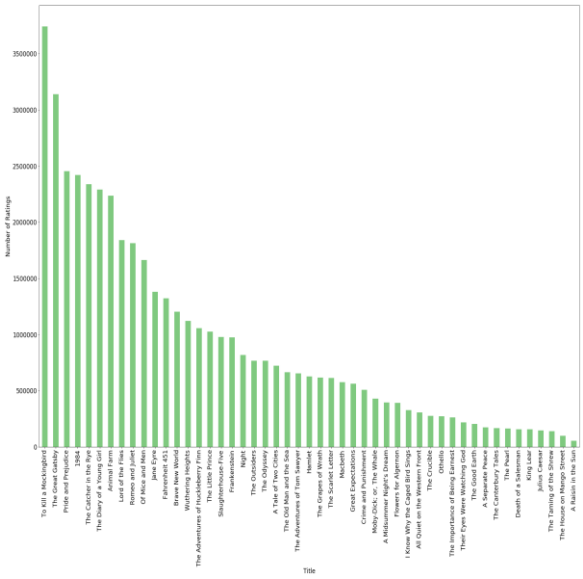

Figure 2: *Number of ratings by title*

The figure shows the popularity of each book based on the number of ratings it received.

Correlation-based recommender

Similar to the popularity-based recommender, the researchers used Python libraries pandas and NumPy. The researchers sorted the order of books by descending number of ratings and created a pivot table that displayed the ratings each user gave to at least one of the books in the dataset. With this data, the researcher found a correlation between the books based on ratings by computing utilizing Pearson correlation coefficients. The researchers used Pearson correlation because it could measure the linear correlation between the books' ratings. The researchers then selected one book to compare with the other books in the list by again using Pearson correlation and drew up a list of the most similar books based on user ratings. This type of data is ratio because it has a "true zero"—when two books compared have no relation at all, causing for no correlation coefficient.

Content-based recommender

For this recommender, the researchers used the libraries pandas and Scikit-learn. The researchers removed the stop words (common words that have no significant meaning, such as "the", "is", "which", etc.) from the descriptions of the books, filled the empty space left by this with blanks, and calculated the tf-idf (term frequency-inverse document frequency) for the descriptions, which formed a matrix with each word in a column so as to easily compare significant words between books. Then the researchers computed the cosine similarity between the words using the linear kernel function in Scikit-learn. This tool allows for the user to see which words are most similar, which would allow the researchers to see the books with the most similar content. The researchers used cosine similarity because this computation would give each of the words a score that they could then easily compare with words of another book. The researchers chose one book to compare the other books to and drew up a list of the top five most similar books based on cosine similarity. This data is ordinal, as the order of the data points was important in telling which books were most similar.

Lexile recommender

For the Lexile recommender, the researchers collected the Lexile levels of the books in the sample. The researchers sorted the books by ascending Lexile levels. They then converted the column into a list, as this was the format needed for a later code, and removed the empty values (the books that had no Lexile scores—mainly plays). The researchers then chose one of the rows, which consisted of the ISBN number and the Lexile level, and converted the Lexile column into an integer to be able to compare it to the other numbers in the list. The researchers then created a function that found the Lexile levels closest to the chosen Lexile level. This type of data is also a ratio because it has a "true zero"—there were books in the dataset that did not have Lexile levels at all.

3. Results

Popularity-based recommender

Through this recommender, the researchers found that the top five most popular books based on number of rankings were *To Kill a Mockingbird*, *The Great Gatsby*, *Pride and Prejudice*, *1984*, and *The Catcher in the Rye*, respectively. The researchers also found that the top five most popular books based on highest ranking were *Night*, *The Little Prince*, *To Kill a Mockingbird*, *Pride and Prejudice*, and *I Know Why the Caged Bird Sings*. The author that showed up the most was William Shakespeare, who had eight works in the list of fifty.

Correlation-based recommender

This recommender showed that the five books that produced the closest correlation coefficient to the most popular book based on rankings- *To Kill a Mockingbird*- were *A Raisin in the Sun*, *A Midsummer Night's Dream*, *Frankenstein*, *The Canterbury Tales*, and *Beowulf*, with coefficients of 1.000000, 0.838525, 0.632644, 0.622543, and 0.562500, respectively. This information could have been found for other books in the dataset as well. There were many books that did not have any relationship at all, as expected.

Content-based recommender

This recommender showed that the five most similar books in content to *To Kill a Mockingbird* were *The House on Mango Street*, *Their Eyes Were Watching God*, *Romeo and Juliet*, *Of Mice and Men*, and *Wuthering Heights*. This information could have been found for each of the books in the dataset.

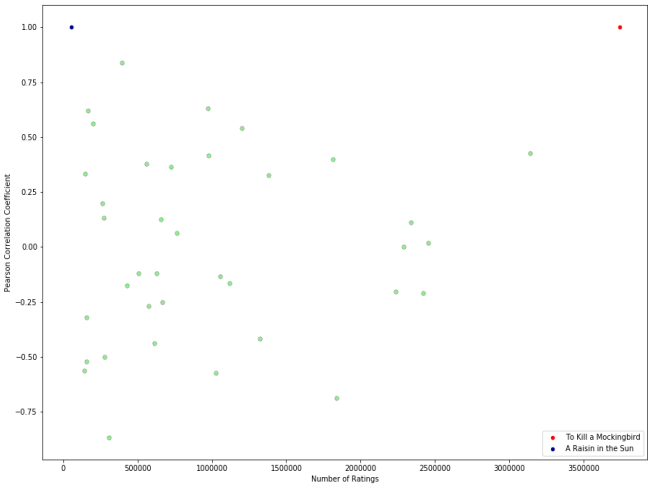

Figure 3: *Most similar books based on Pearson correlation This figure displays the Pearson correlation coefficient of each book in the dataset, sorted by the number of ratings.*

3.1. Evaluation of the Findings

The results of the study supported the hypothesis. The researchers used Python libraries to detect trends among the books. The study showed that as expected, Python is an extremely valuable tool in data science and that its numerous libraries are a key part of its widespread usage. The researchers was able to detect trends among the books through the different types of recommendation systems, which each used their own set of libraries.

As for the results of the recommenders, there was nothing strange or surprising about the results of the popularity-based and Lexile recommenders. The correlation-based recommender produced a surprising result in that the book in the dataset with the most ratings, *To Kill a Mockingbird*, and the book with the fewest ratings, *A Raisin in the Sun*, had the same Pearson correlation coefficient of 1.000000. This could be the influence of the small sample of the dataset, as in the sample, *To Kill a Mockingbird* had 18 ratings and A Raisin in the Sun had 3 ratings. However, *A Raisin in the Sun* did not have the fewest number of ratings in the sample—there were 2 ratings for *The Good Earth* and 3 ratings for another book. Therefore the researchers assumed that there was no relation between the number of ratings of a book and its correlation with other books.

The content-based recommender also found that the most similar book to *Brave New World* in content was *1984*. The descriptions captured the content perfectly—these are two science fiction novels about a futuristic dystopian world in which the characters stand out, or attempt to stand out, from the general populace because of their individuality. This result in particular helped to show that the content-based recommender was working.

The researchers also created a machine learning model. A decision tree regressor trained and tested on thousands of book reviews got an accuracy of approximately 85%. The machine learning model could successfully predict the rating that would be provided to a certain book based on the author, title and current rating from other users. This makes the model suitable to recommend books that could be read by a user who has reviewed other books in the past. The accuracy of the model will increase as the users rate more books and the accuracy of the model goes higher than 90% when the data set is limited to users with over 500 ratings for books.

3.2. Real World Connections

This study helps readers understand how personalized book recommendation systems operate. Though the researchers could not create a user interface in the timeline of this project, the results of this study would ideally lead to user input into the systems and yield more results. Through the usage of these recommendation systems, users could compile personalized reading lists with books recommended to them. For example, the researchers used a sample of books meant for high school students. Specifying the target audience helps to further personalize the recommendations made to the users. In addition, such vital research can be applied by partnering with a book supplying company, which may have specific information pertaining to a customer and would thus be able to input this customer data into the machine learning model engineered to deliver better book recommendations. Such a model can be further extended to numerous different applications, such as health-related applications that input medical data.

3.3. Future Studies and Recommendations

For future studies, as Adomavicius, Mobasher, Ricci, and Tuzhilin [3] suggest, it could be useful to take into account "the fact that users interact with the system within a

particular 'context'" (p. 67) and how preferences differ when comparing between contexts. A different improvement along these lines could also be useful. For example, for the content-based recommender, comparing the whole texts of the books themselves, not just the short descriptions, could yield much more accurate results. Implementing sentiment analysis research from Waumans et al. [4] in correlation-based recommenders could also be helpful. Instead of merely using ratings, one could compare sentiment between reviews.

In addition, the algorithm can be greatly improved through use of neural networks, specifically by using Recurrent Neural Networks with Deep Learning. Computer Vision could also be crucial as it would allow for a higher corroborated accuracy. As the researchers collect more data, this data can be applied towards developing new machine learning models that have higher accuracies. Such use of Neural Networks to improve accuracy is supported by Shin, Kim, Mohaisen, Park, and Lee [5], who suggest that a neural network exchange framework could be beneficial for syntax analysis. Furthermore, combining syntax and lexical analysis could lead to profound and groundbreaking research in this expansive field.

REFERENCES

[1] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine, 17*(3), 37-54. https://doi.org/10.1609/aimag.v17i3.1230

[2] Gorman, L. (2010). Purposes behind summer reading lists. *Teacher Librarian, 37*(5), 52-56. Retrieved from Student Resources in Context database.

[3] Adomavicius, G., Mobasher, B., Ricci, F., & Tuzhilin, A. (2011). Context-aware recommender systems. *AI Magazine, 32*(3), 67-80. Retrieved from Student Resources in Context database.

[4] Waumans, M. C., Nicodeme, T., & Bersini, H. (2015). Topology analysis of social networks extracted from literature. *PLoS ONE, 10*(6), 1-30. http://dx.doi.org/10.1371/journal.pone.0126470

[5] Shin, M., Kim, J., Mohaisen, A., Park, J., Lee, K. (2018). Neural Network Syntax Analyzer for Embedded Standardized Deep Learning. *EMDL'18: 2nd International Workshop on Embedded and Mobile Deep Learning*, 1(1), 37-41. http://dx.doi.org/10.1145/3212725.3212727