Book recommendation engine

DD2477 Search Engines and Information Retrieval Systems

Spring 2022

Group 13

Philipe Granhäll (granhall@kth.se)

Olivia Herber (oherber@kth.se)

Pedro Costa (pmmdc@kth.se)

Wei Shi (wshi@kth.se)

Abstract

Readers today are faced with an enormous amount of books when choosing what to read. This project aims to help readers in the choice of what to read next by creating a content-based book recommendation system. The data of 10,000 books was scraped from the book social media site Goodreads and indexed in Elasticsearch. A user is able to select a number of books that they have read and liked, which will be used to generate recommendations for similar books. The recommendations were based on both the description and genres of the chosen books, with the description being weighted the highest. This approach resulted in good recommendations, which were ranked using TF-IDF-like methods. Notably, the engine recommended books of similar genres to the selected books, and test runs where the user only input childrens’ books or nonfiction resulted in recommendations with only books of that category. In conclusion, the project shows that content-based recommendations for books is a viable option.

Introduction

Today's readers have to choose between millions of books when choosing which to spend their time reading, with large amounts of new books being published every year. In this project, we are looking to build a system that uses content-based methods to generate personalized book recommendations, based on the books that a user has already read and liked.

Background

Recommendation engines are an essential tool on the internet, and are implemented in systems that recommend everything from products and movies to friends and people. It is a system that tries to learn a user's taste and recommend new items that the user might like. The system consists of three parts; a database of items, one or multiple users, and interactions between the users and the items. [1] The interactions are all actions the users can do with the data, for example ratings, purchases, or classifications. After a user has interacted with enough data, the engine will try to find relationships between the interactions and items in order to recommend new items to the user.

The first recommendation systems were implemented in the beginning of the 1990s. An early example was the Tapestry system used at the Palo Alto Research Center (PARC). The system allowed researchers to interact with documents in their database and tag them, essentially letting other researchers know that the document was valuable. [1] Similarly to this, many early recommendation systems relied on users to manually rate items to use for later recommendations, and were mostly implemented in narrow research environments. However, with the rise of the World Wide Web, automatic and scalable recommendation systems started to become the norm. Today recommendation systems are integrated into most web services, and many companies, such as Amazon, Netflix and Facebook, might have their recommendation systems to thank for their success [1]

There are a few main types of recommendation engines, all creating recommendations using different techniques. One of the most common techniques is collaborative filtering, where user's are compared using their rating of items and similarities between different users are found. Each user is then shown recommendations based on what similar users have liked. Content-based recommendation is a technique where each item is represented by a set of features, and recommends items with similar features to other items the user has liked. These techniques can be combined to create even better recommendations. [2]

Recommendation systems can in a way be seen as ranked retrieval, where the relevance feedback, in this case, would be the user's past actions. This analogy was one of the inspirations for the method used in the following work.

Previous/Related Work

There has been some previous work related to book recommendation systems. Pera and Ng created a recommendation system based on social media connections that recommends books that a user’s friends have read. [3] Soyusiawaty and Zakaria used cosine similarity to help library visitors find books with similarities to what they’re looking for. [4] Hwang and Lim used a data mining approach to be able to also recommend unrated and never borrowed books. [5]

Since recommendations are an important part in commercial systems, many implementations of book recommendation engines can be found on commercial websites. The book social media Goodreads, where the data for this project was obtained, also implements book recommendations on its site. When the recommendation feature was announced in 2011, Goodreads stated that the recommendations are based on “how often they appear on the same bookshelves and whether they were enjoyed by the same people” [6] This would suggest that they use collaborative methods for recommendations, as the bookshelves are manually created by individual users. The initial announcement also makes it seem like they hadn’t implemented personal recommendations yet (and instead only showed similar books to individual books), but the current version of Goodreads states that a user can improve their recommendations by e.g. rating more books and dismissing bad recommendations, so personalized recommendations seem to have been implemented.

Lately, commercial websites with the sole purpose of generating book reviews have started to appear, for example, www.whatshouldireadnext.com, www.likewise.com/books, and www.bookarang.com.

Method

All code used for the project is available on GitHub:

https://github.com/WeiSHI7/Project_DD2477

Obtaining data

The data used for this project was obtained by scraping the popular book social media service Goodreads. Since the site stopped issuing new developer keys to its API in December 2020, the data had to be scraped from the website manually. This was done through Python scripts using the “requests” library for accessing the site, and “beautifulsoup” to parse the HTML to extract the data. The data was stored in JSON files for easy upload into Elasticsearch.

Since the scraping was time costly, only a limited amount of data could be obtained. The books that were chosen are the 10.000 top books on a Goodreads list called “Books That Everyone Should Read At Least Once”. All Goodreads users that add to and vote on books on the list, and with the list’s currently over 100.000 voters, the top books should give a good representation of popular books.

First, the list was scraped to receive the URLs to the books (done using the URLScraper.py script), and then the URLs were accessed to scrape the individual page for each book (using the Scraper.py script). This was done in ten chunks of 1000 URLs each since the scraping was slow. Each run also yielded some failed requests, which were collected and scraped in the end. The saved data was each book’s title, author, the series it is part of, the number of ratings and reviews, the average rating, its genres, and the description available for the book.

Implementation

The recommendation engine was implemented using Elasticsearch. Initially, we wanted to host the engine on a server so that the team could all access it easily but we had issues setting up Elasticsearch in a manner that worked for us. The group collaborated in person for this assignment so instead, we simply hosted everything locally.

With the data collected, the approach we took when generating results for book recommendations involved giving different weights to descriptions provided for each read book and the genres the books had. The process for recommending books is the following. The user searches for different books they have read and flag them as *read*, when the user is satisfied, they ask for recommendations based on those. The engine then gathers the books' descriptions and merges them - in a similar way to how an augmented query would be created, when implementing relevance feedback. The merged description is then used to search for new books, which are ranked using cosine similarity and TF-IDF-based methods. More precisely, the BM25 metric. The merged genre is also considered, but with less significance.

The current approach is a content-based recommendation system where we do not use information based on other users' ratings to return results. The benefit of a content-based approach is that we are able to ignore the need for 'external information' and recommend books solely on their content. This allows us to potentially recommend new and unpopular items as well as tailor them to a particular niche the user may have. Of course, the system then depends on the user's previously read books but that is assumed to always be given for this assignment.

Results & Evaluation

The results obtained from the tests run seem to be quite consistent with the aim of the group. We set out to create a content-based recommendation system that recommends similar books and that allows for recommendations that may not be deemed the most popular.

With all the tests that we have run, the results are quite instant. We chose to calculate our metrics by using 10 recommendations instead of 3, to get more accurate metrics. Based on the books selected, we often find ourselves seeing several books of the same series showing up, which can of course be expected given the implementation. When comparing to other solutions found online, e.g. Goodreads, whatshouldireadnext.com, and likewise, we notice that books in the same series are often not recommended next to each other. This is probably the case as we can assume that the reader already knows of the series when recommending books to read so it would not be relevant. The implementation we have does not take this into consideration as this was not something mined when combing through Goodreads.

When looking at Goodreads, the recommendations for books when searching for a title is presented as *“Readers also enjoyed”* indicating that it probably takes the ranking of the book and compares it to those of other users, i.e. a collaborative filtering recommendation system.

The characteristics of the books returned are quite a clear cut. The engine makes do with the information it has. If a user would want to filter after the audience type, like adult or children's book, this would simply have to be a genre that is attached to the book. If the user would pick books from varying audience types there is nothing stopping the engine from recommending books that are not the intended age group.

Other Critique

Something we observed during testing is the existence of books that have duplicate copies. In the data, this is seen as different ‘versions’ of the same book such as a new edition or publisher. This is something we observed on the website whatshouldireadnext.com as well. These books often have the same descriptions and genres. In the evaluation, these are ignored when making computations.

Table 1 illustrates some tests run using the recommendation engine created by the group. The scores are computed by assigning relevance to each search result and an ideal relevance ordering based on those. Ideally, there would be an averaged relevance rating for the searches but that was not feasible.

Search	NDCG
(Mystery) Inferno (Dan Brown) Murder on the Orient Express (Agatha Christie) Gone Girl (Gillian Flynn)	0.96
(Non-fiction) A Short History of Nearly Everything (Bill Bryson) Sapiens: A Brief History of Humankind (Yuval Noah Harari) A Brief History of Time (Stephen Hawking)	0.90
(Children's) Charlotte's Web (E. B. White) The Little Prince (Antoine de Saint-Exupéry) Charlie and the Chocolate Factory (Roald Dahl)	0.97
(Fantasy) Circe (Madeline Miller) Hobbit (J.R.R Tolkien) American Gods (Neil Gaiman)	0.94
(Romance) Outlander (Diana Gabaldon) Pride and Prejudice (Jane Austen) Dear John (Nicholas Sparks)	0.92

Table 1: NDCG Results

Conclusions / Discussions

The content-based recommendation system revolves around suggesting books with similar descriptions and genres to users previously read books. The tests we have run definitely yield results that point at a functioning system with books almost always relevant to the query input. The computed NDCG scores are all high indicating high relevance. Whether the books are the ‘best’ books that could have been recommended to a user is highly subjective and is perhaps not an accurate way of measuring the recommended books. It is also difficult to comment on the quality of the books recommended as ratings were not taken into consideration. The NDCG metric is perhaps not one that can accurately portray the quality of search results by itself.

The driving forces behind the engine are the tools available to us by elasticsearch which allow for rapid computations that we could utilize for our approach. The design of the algorithm was done based on assumptions made of the books in the database, that their descriptions would yield quality information that could be used to measure the similarity of other books. The same goes for genres. The weight we chose for the data points during computation was done holistically. As mentioned, the quality of books is difficult to judge so we had to intuitively decide on a balance that worked.

Future work

If we were to improve on this system's ability to recommend books, the incorporation of ratings would be an area to focus on. This would perhaps shift the type of recommender system and have a focus on trying to ensure quality books instead of books that are simply similar to those a user has read. Another approach one can incorporate is having a user rank books they have read as either good or bad and incorporate that into the algorithm.

References

[1] M. Schrage, “Recommendation Engines,” *MIT Press Essential Knowledge Series*, Massachusetts Institute of Technology, 2020, Cambridge, Massachusetts.

[2] R. Burke, “Hybrid Recommender Systems: Survey and Experiments,” *User Model User-Adap Inter* 12**,** 331–370, 2002, https://doi.org/10.1023/A:1021240730564.

[3] M. S. Pera and Y. Ng, "With a Little Help from My Friends: Generating Personalized Book Recommendations Using Data Extracted from a Social Website," *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 2011, pp. 96-99, doi: 10.1109/WI-IAT.2011.9.

[4] D. Soyusiawaty and Y. Zakaria, "Book Data Content Similarity Detector With Cosine Similarity (Case study on digilib.uad.ac.id)," *2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, 2018, pp. 1-6, doi: 10.1109/TSSA.2018.8708758.

[5] S.Y. Hwang and E.P. Lim, “A Data Mining Approach to New Library Book Recommendations” In: , *et al.* Digital Libraries: People, Knowledge, and Technology. ICADL 2002. Lecture Notes in Computer Science, vol 2555, 2002, Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36227-4_23.

[6] C. Kyusik “Announcing Goodreads Personalized Recommendations,” *Goodreads News and Interviews,* September 15th 2011, Accessed April 27th 2022, https://www.goodreads.com/blog/show/303-announcing-goodreads-personalized-recommendations.