Content-based Recommendations for Books

For my mini-project, I used content-based filtering to build three different recommendation systems for a Goodreads book dataset. My recommendation systems were based on book genre, plot synopsis and author. In this report, I will be discussing the results of the systems built as well as discussing which of them was the most appropriate to use for recommending books. Content-based filtering involves using the explicit features of media content to compare what media the user liked with all the other items in a dataset and outputs suggestions of other media based on the user's preferences. These sort of recommendation systems are already used by a lot of streaming platforms such as Netflix to recommend new shows to their users. Chiny et al (2022) stated that these sort of recommendation algorithms, especially with the rise of Big Data, allows users to quickly extract the desired information from excessive detail.

I used a Goodreads dataset from Kaggle to perform content-based filtering – Goodreads can best be described as a social networking site for books, where users are able to leave reviews, ratings and create reading lists. The dataset contained information about book titles, authors, descriptions and ratings, which was suitable for the recommendation algorithms I wanted to build.

The first recommender built was ***recommend_by_genre*** (i.e. the Genre Recommender), a function that, when given a book title as input, will give the user the top ten similar books that are in the same genre. To do this, I first had to clean the genre column of the dataframe to turn each book's genres into a list format; these genre lists were then converted into binary and turned into one-hot encoding so that each book was represented as a feature vector for the Genre Recommender. With this, cosine similarity and K-Nearest Neighbour was used within the recommendation function to find the top ten books for each book put in the function. When the book title is put into the function, the function iterates through the dataset; if the input book's ID does not match the ID of the current book in the dataset, the cosine similarity is found and appended to an empty array. Then, the function uses K-NN to find the 10 nearest neighbours of the input book and outputs the results (along with the recommended books' genres and average ratings). In testing this system, the function was able to give recommendations based on genre very well as there was a connecting genre(s) between all the results. However, when using books that were part of an established series, the recommended books were mostly of the same series and/or written by the same author: this had happened when using examples like *The Lord of the Rings* or *Matilda*. So this recommender will give similar results, but it may not be ideal in the case of providing users with diverse recommendations. For the system to do that, the book must be relatively independent; for example, *Wuthering Heights* gave relatively more diverse book recommendations, but this could have been helped by the fact that this was Emily Bronte's only book. However, the recommender can also distinguish between books in the same genre that have different tones. For instance, *The Hitchhiker's Guide to the Galaxy* is a more light-hearted and comedic science fiction book than *Dune* and the recommendations for each followed suit.

The second recommender built was ***recommend_by_plot*** (i.e. the Plot Recommender), a function that also outputs the top ten recommendations based on natural language processing. Unlike the genre lists, the book descriptions are large blocks of text, as such, I created TF-IDF vectors of the plot descriptions and used cosine similarity to find books with similar plots. The Plot Recommender was giving more diverse options for the user but the recommendations were not very accurate. Some synopses having the same wording and tone may not be enough to indicate which books are the same (an author may attempt to trick the readers) – *The Princess Bride* is more of an adventure story than at first glance, the only books that were complete matches with the novel would be *The Return of the King (Lord of the Rings series)* and *The School for Good and Evil*: both books share the input book's themes and aesthetic. Another issue with this method, is that books can be

recognised as recommendations solely based on similar words appearing. For example, *Alice's Adventures in Wonderland/ Through the Looking-Glass* had half of the recommendations containing the word 'Alice'. Generally, this recommendation system would be suited for users if they want to find new books that may or may not be similar.

The last recommender that I built was ***recommend_by_author*** (i.e. the Author Recommender). While using the other two features made basic recommendation systems, I wanted to attempt using an author of a particular book to recommend new authors from the dataset. In making the system, I had adapted and improved the code for ***recommend_by_genre*** to give recommendations that did not involve other works by the input book's author. To do this easily, in the function code, I would take out all the rows in the dataset that contained the base book's author and assign it to a new dataframe that would only be used inside the code. However, the input book's index was extracted from the original dataset and appended to the function one. After that, the recommender uses the same method as the Genre Recommender to find the top ten books for the user. Although this recommender is essentially an extension of the first recommendation system, having some emphasis on the author improved the results; the recommender did not give out books that were written by the same author (which means there is reduced author bias in this recommender than the first one). Whilst there were some recommendations listed that were written by the same person, they don't take up as much space on the recommended list as the Genre Recommender. This recommender is more suited for users who want to find books from different authors that would be like the books of their favourite author. While this method works well, the recommender does not consider authors that write in different genres (realistically, this may not be a major concern for users).

The Author Recommender ended up being the best one for this dataset as there was a balance of author diversity and the recommendations being similar to the input book; this recommender showed that more information given to the system produces better results. However, most entertainment systems use different content-based filtering techniques (even combining theme with collaborative) to help users search for different purposes. The recommendation systems all seem to have a particular goal in what they recommend to the user. Nonetheless, the content-based recommendation systems were successful in producing recommendation lists from the Goodreads dataset.

Bibliography

Chiny, M., Chihab, M., Bencharef, O. and Chihab, Y. (2022) 'Netflix Recommendation System based on TF-IDF and Cosine Similarity Algorithms' (from the International Conference on Big Data, Modelling and Machine Learning, 2021), pp.15-20. Available at: https://tinyurl.com/p9n2vz36 (Accessed: 15 June 2023)

Johari, I. (2023) 'Best Books (10k) Multi-Genre Data'. Available at: https://tinyurl.com/mrynbucf (Accessed: 05 June 2023)