2/06/2022
COSC480

PROJECT REPORT

Nicole Dunn
NDU31

Introduction

A recommendation system is an information filter that tries to predict a user's preference to a given item. They are used in multiple systems online and have different filtering algorithms depending on what outcome the programmer is wanting to produce for a user. One of these algorithms is content based, this means the filter tries to find a similar item to one already liked. In the context of this project, the recommendation system will be used to recommend similar books to a reader based on prior interests and the GoodBooks-10k-extended data set. This report focuses on the creation of this book recommendation system using a content-based filter and the steps and challenges faced as design of the program progressed.

Development

The initial concept for this program came up due to a prior need for a fresh recommendation following finish a book series. The first step was finding a source for the data and GoodBooks was seen as the most valuable entity in this process. The dataset used is one premade by other users that offers an extended version of the GoodBooks-10k dataset. This includes titles, authors, descriptions, ratings and genres collected from the sites ten thousand most popular books, with the ratings coming from around 50,000 different users. With all this data it opened a lot of possibilities for how it can be processed into a usable program and many ideas stemmed from this.

It was originally thought that the program could do a massive cross comparison with all different fields, but this soon became outlandish in terms of skill, and this was reduced in view of the different types of filtering sets available. These filters are as follows, content-based, collaborative-based, popularity-based, and hybrid-based. Collaborative depends on the different user data according to the original user's data and any correlation between the two would let the algorithm recommend similar content between the matches. This would be useful if both the online user's data and the original user data was updated every time the program ran, but this process would've been arduous with the given time limit. Popularity has basis in how popular an item of content is, even if this is negative popularity. This would mean a book with the highest number of reviews with no correlation with previous completed books would be recommended every time which isn't useful in the case of this program. Hybrid recommendation solves this problem with popularity as it combines two or more algorithms to return a more connected result to the user which is similar to Google's search function. Finally, content-based analyses the similarities between two pieces of content and allows for a result that has a much stronger connection to the one given.

The content-based filter was chosen as it offered a new way to analyse book similarities compared to those online which favour comparisons between author or genre. A reason author wasn't used is that when it is used a lot of the recommendations will end up becoming the sequels of the given title. This isn't necessarily a bad thing, but it does mean that the user will not be getting new or unseen recommendations when they've already read the entire series. Genre presented its own issues as it would require a second part analysis to provide a usable recommendation, and this became an idea for a future development for the program as it was starting to get very in depth. A lot of consideration went into whether rating should come into consideration when giving the recommendation, but it was thought that it would introduce too much sway and skew the results for whatever outcome.

When it was settled what sort of filter would be used, a lot of effort went into cleaning the data to help prevent any mishaps when the program ran. This ranged from dealing with different languages to removing duplications within the dataset. Alongside this came a lot of research into different modules to use, ones like BeautifulSoup or SKLearn came from skills learnt in Digital Humanities which analyses text data and helps to clean it for data analysis, whereas others came from recommendations to help speed up the program like lxml which speeds up character reading for BeautifulSoup.

Once the data had been cleaned and the filter had been chosen the rest of the program's development went smoothly without too many other changes. Some challenges arose but this didn't affect the outcome of the program.

Use of the Program

When the program initially runs, it prompts the user to enter the title of a book to base the recommendation on. Once a title has been entered, the program compares this to the GoodBooks-10k-extended dataset titles to see if there is a match. If there is no match, a message pops up to tell the user of this error and to please try again. Once a match is found, the title is associated with its associated book ID from the dataset, the book ID helps keep the program associated to the correct book.

In the background since the program first ran, the data has been imported and cleaned. As the book description is the basis for the recommendation the first step of cleaning was removing the HTML from them using BeautifulSoup, next was dropping the erroneous columns that wouldn't be used in the rest of the program. This second step put the dataset at 17 columns compared to the original 29 it started with, and because the columns have been sorted the next step was to clean up the rows. Because this program is based on an English perspective, all books that did not contain an

English language code were removed and all duplicated titles were removed to assist with the programs original search for titles. This reduced the rows to 9645 rows from the original 10000 data points it started with.

Once the data had been cleaned, an extra column was added combining the title and descriptions together, this allows the use of CountVectorizer which converts text to numerical data in a matrix. Taking this numerical data and passing it through cosine similarity produces a similarity matrix to all other books within the dataset and this is the baseline for all the recommendations taking place. This is where the book ID comes back into play as each similarity score has an ID associated with it and the program returns with the highest similarity scores for the chosen book.

Finally, the program then prints the top 5 recommendations with their Authors in association with the given title.

Positive Progress

Along the process of developing the program a lot of things went well which made the process very pleasant. The first would be finding an easy-to-use dataset with clear instructions on how to import the dataset to use. This gave a good introduction for how the program would progress and there were plenty of files within the website to provide further clarification over what was contained in each column and row of the dataset. When using the dataset, it also became very clear that it was well labelled and stored for use which made it a very easy learning process on how to manipulate it for the use of this program.

Another situation that went well was searching for modules to use. As mentioned previously some of the modules used came from inspiration while studying Digital Humanities as the focus there is text interpretation and converting it into numerical data for analysis. The other perspective of this is for the modules that came up organically, these were ones that helped smooth the process over or help with data exploration. Fortunately, all modules had relatively well laid out documentations to explain how to use them and to fix any errors that came up.

Exploring the dataset came along with increasing skills and this gave an opportunity to explore for any future development the program could endure. This also helped eliminate any extra steps when not understanding the dataset as it could be easily modelled and analysed to see where any issues may be arising, which was the introduction step to cleaning the data so these issues wouldn’t be faced in the overall program.

The entire process while developing and writing the program was a widely positive one and a lot of skills were gained and then honed to create a program that produces pleasing results.

Challenges

The largest challenge faced was a corruption within pip. It is still unknown how this problem came about but it meant that no modules could be installed, and a significant bit of time went into trying to understand the issue to resolve it. Ultimately a solution was found to install a separate store version of Python and route Wing to that version for installation. Although an incredibly frustrating challenge, this did produce a learning opportunity and it was incredibly satisfying to figure out a solution to the issue.

Along with the previous challenge, figuring out which modules to use sometimes presented issues. This was more because there were just so many to choose from and it could be difficult to narrow it down to ones useful for the program's development. Fortunately, there were a lot of websites and help guides to narrow down the selection and to help with how to use them. A minor challenge in retrospect but it did take up a fair portion of time in order to learn.

The last challenge faced was more of a frustration than anything. When the program is run within Wing it returns with a warning from BeautifulSoup even though it still proceeds as normal. From further research, this warning doesn't mean that something is wrong but that it might be better to import the file straight into the module. The creator of the module mentioned that they included this warning so that they didn't have to fix the module for every different type of file that might be analysed by it. And can give a user an error report if an issue arises, as this isn't an issue in this case it is safe to ignore but it was a big worry when trying to deal with it.

Future Development

As mentioned before, thought has gone into how to develop this program further. The first step taken would be dealing with the BeautifulSoup warning mentioned in the challenges section, this may come with a different sort of import system or even downloading the file so that it can be parsed with the module. In addition to this, creating a way to have the dataset become up to date so there is a constant flow of fresh additional data for continuous recommendations would be a large step forwards for developing this program.

A consideration would also be to give the program a user interface that looks nice and provides auto complete for the titles so that there isn't any user related error regarding title input. This would help to give the program a more complete feel as it would be able to function as a web app rather than through Wing. Along with the user interface, interactive graphs would be a useful aspect to give a user more variety in their choices as it could provide up to date analysis for whatever filters they would like to compare with.

More development regarding the filters would also be a good step as this would allow for hybrid filtering which produces much more reliable results and can allow for larger searches over many different variables rather than just the one. Further development to this point would be so that a user could use their GoodBooks account and read list to provide ample history for a content search while also filtering out books already read.

Conclusion

In conclusion, this has been a pleasant process to create and develop a book recommendation program. There was plenty of learning opportunity and room for growth in terms of the challenges faced. The program focuses on content-based filtering which places more importance on title and the description of the book rather than rating or genre which is a unique choice compared to other book recommendation systems out there. Finally, there is also plenty of opportunity for future development, with a clear path forward as to how that could progress.

References

Agrawal, Raghav. "Book Recommendation System: Build A Book Recommendation System." *Analytics Vidhya*, 27 June 2021, https://www.analyticsvidhya.com/blog/2021/06/build-book-recommendation-system-unsupervised-learning-project/.

"Beautiful Soup Documentation." *Beautiful Soup Documentation - Beautiful Soup 4.9.0 Documentation*, https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

Malcolmosh. "Malcolmosh/Goodbooks-10K-Extended: Extended Version of the 2017 Dataset, with Additional Fields." *GitHub*, https://github.com/malcolmosh/goodbooks-10k-extended.

Zygmuntz. "Zygmuntz/Goodbooks-10K: Ten Thousand Books, Six Million Ratings." *GitHub*, https://github.com/zygmuntz/goodbooks-10k.