Book Recommendation Engine

Arielle Rabinovich and Andy Sun

DS 3500

Link to Repo: https://github.khoury.northeastern.edu/arabinovich/book-recommender.git

Abstract

Finishing a great book is a bittersweet feeling; while it was a great experience, it is hard to part ways with the world the author created. This is why we decided to create this book recommendation engine. Sites such as Goodreads provide recommendations based off user similarities; if user A and user B both like similar books, then a book user B has read and enjoyed will be recommended to user A. However, we wanted to find similarities between books through actual writing style. By delving into Natural Language Processing (NLP), we explored excerpts by calculating aspects such as average syllable count, average word frequency, average sentence length, average word length, the Flesch-Kincaid score, and the Automated Readability Index (ARI).

Using these statistics, we then used the SciKit-Learn library and K-means clustering to cluster together the books, using the elbow method to find the appropriate amount of clusters

We then developed a dashboard using plotly dash where the user can hover over points in the scatter graph and see which books were clustered together as well as choose a book from a dropdown and get the top 5 closest books in one tab. The other tab being some background information about the project to give the user some more understanding.

Background

Currently, book recommendation engines rely on similar users, the same authors, and genres to make their recommendations. While these are sound methods, we wanted to see if analyzing actual writing style could provide better options. To do this, we needed to leverage NLP. By combining our human created statistics, such as average word frequency, average sentence length, Flesch-Kincaid score, etc. and the machine learning aspects of K-Means clustering, we were able to explore the excerpt with Natural Language Processing.

Of course, there are some limitations here. The classic NLP libraries such as TFIDF provided clusters based on term frequency, but they were not defined or clear groups, and we felt it didn't explore enough of the text. We did utilize the NLTK library for its stop words, however.

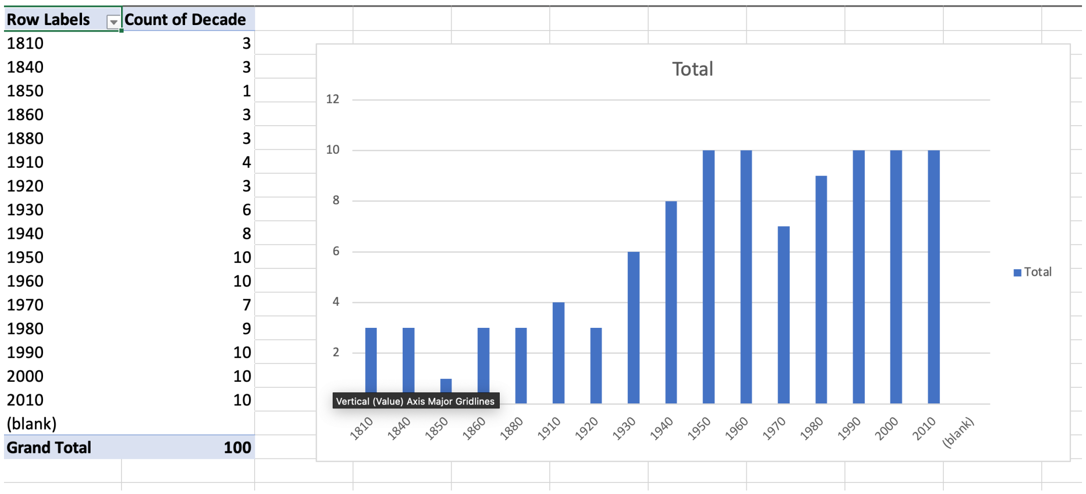
K-Means clustering allowed us to group our excerpts *k* clusters based on the similarity of data points. These datapoints were determined based on the aforementioned statistics. The algorithm aims to minimize the within-cluster sum of squared distances, making data points within the same cluster as similar as possible and those in different clusters as dissimilar as possible.

To choose the *k* value, we employed the elbow method, which plots the within-cluster sum of squared distances (WCSS) against different *k* values. We found 2 clusters to be the elbow, however we felt this was a bit scant given the objective of having different groups to show users. But we also didn't want to interfere with the clustering, which deterred us from choosing our initial desire of 10. So, we chose 5, as we found this provided more unique and customer results as opposed to 2 or 3 which would give you virtually the same 5 books no matter what you chose within that cluster. By making the clusters smaller, you got more specific recommendations

Then, by using the plotly dash library, we were able to turn our findings into a fun, interactive experience, which we felt better modeled an actual book recommendation engine as opposed to typing the preferred book into the console. It also allowed the user to choose from a dropdown, as they had to pick a book already in the dataset.

Methods

We began by curating out dataset. We wanted a strong emphasis on books from the past 50 years, so in our Excel (before it was a CSV), we wrote the year of publication to gather the decade. To ensure we had a wide spread, trying to avoid bias to more modern books, we created the following pivot table:



Row Labels	Count of Decade
1810	3
1840	3
1850	1
1860	3
1880	3
1910	4
1920	3
1930	6
1940	8
1950	10
1960	10
1970	7
1980	9
1990	10
2000	10
2010	10
(blank)	
Grand Total	**100**

We chose our titles partly through personal bias, and through researching the most influential books of a given decade. Here, we had to be selective, as many works were graphic novels or plays, which did not fit our objectives. We sourced our excerpts from their respective publisher site, such as Penguin Publishers, or other sites, like Alma Books. After the creation of our Excel sheet, which included book title, the author, year published, decade published, and the excerpt, we transformed it into a CSV to be analyzed. All columns excluding Title and Excerpt were dropped, but they were necessary in ensuring we didn't have a strong bias towards a particular decade or author.

We then needed to look at the features. As aforementioned, TFIDF did not achieve what we wanted it to, so we looked at what makes a piece of writing distinct, how does an author share their voice. We knew we needed to look at averages, as the excerpts were different lengths, which is why punctuation count was created, but then not implemented as obviously longer excerpts have more punctuation. To begin our feature extraction, we first needed to parse the excerpts. Depending on the feature, stop words were removed or kept in. For the readability scores, they needed to be kept in as this was part of the calculation. The calculations with stop words kept were the Flesch-Kincaid index, the ARI, and average sentence length. The calculations where stop words were removed were average word frequency (on average how many times is an individual word repeated), average syllable count and average word length. The indexes mentioned were decided on after doing research on how humans digest writing. Simply put the Flesch-Kincaid score and the ARI communicate how easily read a text is. To ensure accuracy, we ran some excerpts through online calculators and compared with what our program found.

Then, the meat of the project began. We implemented k-means clustering by attributing each book with their respective list of features. We then turned this into a NumPy array and fit it with two PCAs for clustering. As aforementioned, we chose 5 as our number of clusters.

We used the scatterplot and developed a plotly dashboard, where the only user input was which book they chose, which was selected from a dropdown. Taking this input, the next 5 books in the cluster, removing the one submitted were recommended. We also created a second tab for fun, to model the background of our project, mimicking that of an actual book blog.

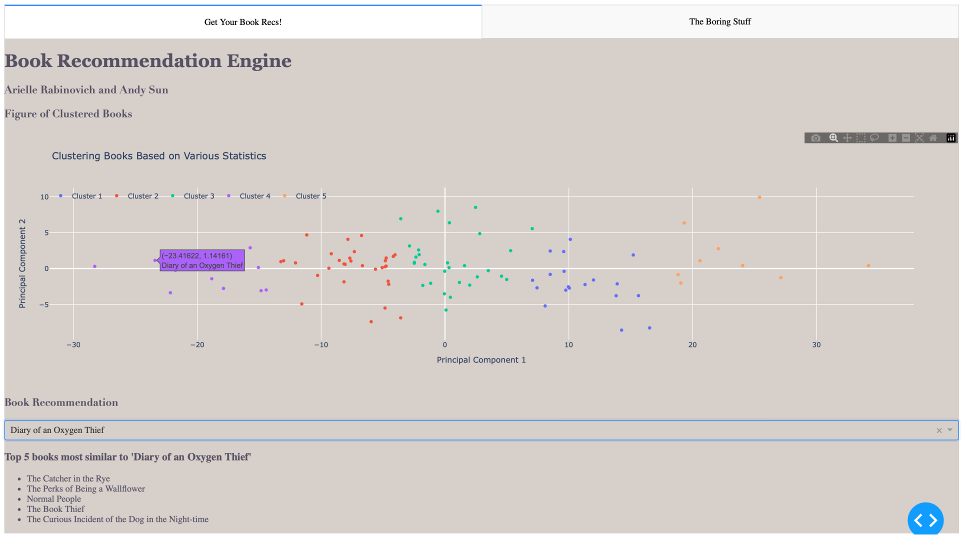
Analysis

There were a few bumps along the road when creating this application. As mentioned, we started with TFIDF, as it works very well alongside K-means clustering. However, the clusters were overlapping and seemed to be made arbitrarily, so we could tell this was not a good measure for our excerpts.

When we moved on to the combination of statistics we have now, we quickly realized syllables are a difficult thing to calculate in python. However, it was needed for indexes which were crucial in quantifying writing style as they are established numerical representations of readability. We tried our best, and we never had a word off by more than 1 syllable. These errors were infrequent and with them being both above and below, they cancelled each other out. We don't believe there is a way for a program to perfectly calculate syllables, given all the exceptions within the English language.

Furthermore, we had one outlier in our dataset, The Scarlet Letter, which was consistently assigned its own cluster, disrupting the other clusters, so we thought it was best to remove it from the dataset. We are not entirely sure what caused it to be such an outlier. Perhaps with more writing from this era, it would have company in its cluster, or perhaps it was an issue with the excerpt we sourced.

But, after all of the tweaking we got this visualization:



Book Recommendation Engine

Arielle Rabinovich and Andy Sun

Figure of Clustered Books

Book Recommendation

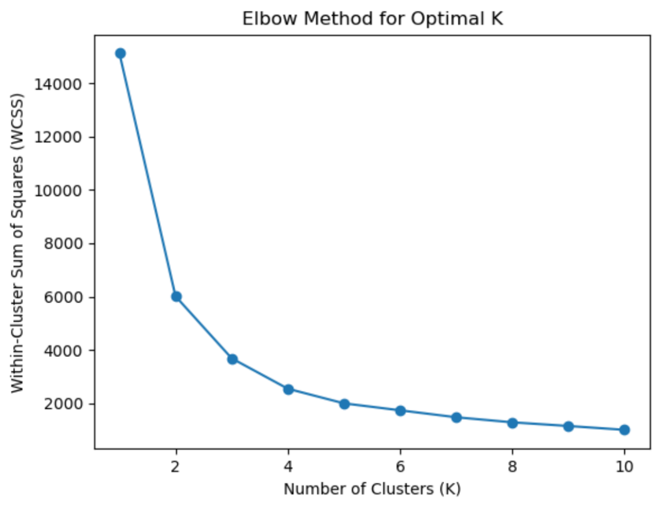
Diary of an Oxygen Thief

Top 5 books most similar to 'Diary of an Oxygen Thief'

- The Catcher in the Rye
- The Perks of Being a Wallflower
- Normal People
- The Book Thief
- The Curious Incident of the Dog in the Night-time

We feel it provides pretty accurate and interesting recommendations. In the example shown above, having read both *Diary of an Oxygen Thief* and *The Catcher in the Rye,* these are similar reads. But it is interesting that this was found by the application as they were published over 50 years apart, which would make one think their writing styles would differ too much for them to be claimed so similar. What's the most intriguing is that the author states on the inside flap that the narrator/main character of *The Diary of an Oxygen Thief* is supposed to be a modern-day Holden Caulfield. Meaning, both books are written in first person and narrated by similar characters, so it absolutely is reasonable to assume that the writing styles of the books would be similar as well. This captures the purpose of this engine. Despite different authors, and the fact these books were written over 50 years apart in different centuries, they can still be detected to be similar solely based on the written content. (But if you're looking for a personal opinion, *The Catcher in the Rye* is the better novel).

Here is the elbow method graph:



As illustrated, 2 and 3 are more optimal clusters, but for the purpose of giving more specialized recommendations, we went with a larger k value.

Conclusions

We feel we were able to provide quite reasonable recommendations, but we found difficulty in the fact that how can one *really* test how closely related two books are. In the future, we would love to combine user similarity and NLP to generate book recommendations. Also, it would be beneficial to allow the user to choose how many recommendations they would like instead of only providing 5. And, of course, expanding our dataset would allow for stronger recommendations and better analyses.

References

Alma Books, 23 June 2020, almabooks.com/

"The Automated Readability Index." *Readable*, 9 July 2021, readable.com/readability/automated-readability-index/.

"Flesch-Kincaid." *WebFX*, 12 July 2023, www.webfx.com/tools/read-able/flesch-kincaid/.

Project Gutenberg, www.gutenberg.org/. Accessed 7 Dec. 2023.

"Welcome to Penguin: We Are What You Read." *Penguin Books*, 28 Nov. 2023, www.penguin.com/.