



# National College of Ireland

BSc (Honours) in Computing

Specialisation: Data Analytics

Academic Year: 2020/2021

Name: Remigija Vindasiute

Student Number: 17432284

Email: [x17432284@student.ncirl.ie](mailto:x17432284@student.ncirl.ie)

## Classical Book Recommendation System

### Technical Report

## Contents

Executive Summary .....	4
1.0 Introduction .....	5
1.1. Background .....	5
1.2. Aims.....	6
1.3. Technology.....	6
1.4. Structure .....	7
2.0 Data .....	8
2.1 Classical Book Data .....	8
2.2 Modern Book Data .....	9
3.0 Methodology.....	10
3.1 Implementation .....	11
3.2 Important R functions Used .....	17
3.3 Important R Packages Used .....	17
3.4 R Shiny.....	18
3.5 Things That Did Not End Up in The Final Version of the Project .....	19
4.0 Analysis .....	20
4.1 Word Cloud .....	20
4.2 Top Words per Topic.....	24
4.3 SOM, LDA, Kmeans models.....	27
5.0 Results.....	32
5.1 Testing.....	34
6.0 Conclusions .....	39
7.0 Further Development or Research .....	40
8.0 References .....	41
9.0 Appendices.....	44
9.1. Project Plan/Proposal .....	44
Objectives.....	44
Background .....	44
Technical Approach.....	44
Technical Details .....	44
Evaluation .....	44
Project Plan .....	45
9.2. Reflective Journals .....	45
October .....	45
November .....	45

December.....	46
February.....	46
March.....	46
April.....	46
May.....	46

## Executive Summary

The purpose of the report is to present the steps taken towards creating a classical book recommendation system and then the results achieved from it. The report includes introduction, data description, methodology, analysis, results, and conclusions sections that cover every step of the project.

The overall goal of the project is to create a book recommendation system – one that takes in input, data about modern books, and produces an output which would be classical book recommendations. The recommendations would be based on the similarity between the topics (and words) within the books. Some of the main models used in the project include LDA, SOM and K-means.

## 1.0 Introduction

### 1.1. Background

I have always been interested in books (especially fiction) and writing, so it seemed like a good idea to try and incorporate that into my final year project. The idea that I came up with in the end was to gather and analyse data relating to books (such as reviews, descriptions, or text of the book itself) and use that to try and predict what kind of books someone would like (based on the books that the person already enjoyed).

In terms of recommender systems, there are two main types – collaborative and content based. In many cases a hybrid of both types is used.

Collaborative system is when the past interactions of any user with a product/item are what determines which products/items will be recommended to them. As an example, if I were to add a movie/show on Netflix to the list of things I plan to watch eventually, that would count as an interaction – I clicked on the movie, read the description, and then added it to the list. That signifies a clear interest. After many of such interactions, what I see as recommendations on Netflix changed significantly – for example, after watching a few Asian tv shows that my friends recommended, I now see a lot more of them at the top of the recommendations on Netflix.

Content based system is when the content of the product/item (such as, the description, genre, specifications, typical user profiles, etc.) is used to determine who is most likely to be interested in that product. For content-based systems, the additional information used can be either user-based (the specifications of the user, such as age group, gender, etc.) or content-based (genre of the product, description etc.). As an example, I often see a video recommended on YouTube where, at the bottom of the video, it says something like “viewers of ‘this other channel you watch’ also watch this channel”. That would count as content-based recommendation which bases its logic on other user’s profiles. The assumption is that if there is a large crossover of viewers from two different channels, then the viewers from channel 1 are very likely to enjoy the content from channel 2 and vice versa.

It is a content-based recommendation system that would be the most suitable for my type of project. The overview of the process that would be taken to reach the result (the recommendation system) is as follows; first will be the data gathering stage, including getting the book data for classical books and extracting summaries of modern books through an API. Then, the data gathered will be pre-processed and analysed, various topic modelling (LDA) and clustering (K-means, SOM) models will be applied. The recommender system itself will make use of LDA model to find topically similar books and then as a further step, a cosine similarity matrix will be used to find the similarity score of the books that fell within the same LDA topics. The final recommendations would then be based on the similarity scores between the books.

## 1.2.Aims

The aim of the project is to create a content-based recommender system that would take in a list of books that a particular person likes, analyse them, and then recommend a few classical books for the person based on the results of the analysis. The recommendations would be based on the similarity of the content – the text of the books in the case of classical books, and the summaries in the case of modern books.

*Aim 1:* gather the required data in terms of classical and modern books, pre-process it and prepare it for the analysis.

*Aim 2:* Apply LDA or similar models to cluster and divide the books into separate groups that would be based on how similar the books are.

*Aim 3:* Do some data visualisations for the data after the LDA model has been applied.

*Aim 4:* Implement the recommender system by making use of the LDA and cosine similarity matrix.

*Aim 5:* Create a user interface (UI) through R Shiny, write the final report about the project.

## 1.3.Technology

For the project, the languages that I will be making use of are R and, where needed, Python. The analysis and pre-processing steps will be carried out with R, while Python will be used in the cases where I would need to work with API's (the Google Books API for getting the modern book data).

R is a programming language used for statistical analysis, data mining and modelling, etc. It is typically used through an IDE (integrated development environment) such as R Studio. Python, in comparison, is more of a general-purpose programming language which can be used for many different things, including but not limited to data analytics.

An API (Application Programming Interface) allows different applications to communicate with each other, it makes it possible to do things such as retrieving data from another application which is what I will be using the Google Books API for. By using the API, I'll be able to source the data for modern books, data that will originate from the Google Books website.

For the creation of the recommender system, it is NLP (Natural Language Processing) and text mining that I will be using. DTM's (document-term matrix) will be irreplaceable for many parts of the project, it is used for indexing of words within documents (or books in this case). A DTM can be used in many ways, and in the case of text analytics and using mostly categorical data, it allows the data to be used with models that would typically only work with numerical data.

NLP is something that allows computers to interpret our language (in the form of text-based data) and then manipulate it as needed. NLP makes it possible for computers to not

only read text but also understand it, interpret it, and even hear it. It can be used for determining the parts of the text that are important, for running sentiment analyses to determine intent, for handling textual data and modelling of textual data in general.

I will also be using Google Colab for execution of code, to by-pass some of the issues I encountered, such as not having enough RAM on my own device for some parts of the code to fully execute.

Google Colab is a web-based IDE that can be used with either Python or R languages. It provides access to resources such as GPU's and RAM, which can be very useful in many different situations. In my case, it allowed me to make use of more RAM than my own device had, therefore executing parts of code (and models) with bigger data samples than I would have been able to use otherwise.

#### 1.4. Structure

The document structure is as follows: section 1 is the introduction to the overall project, section 2 describes and summarises the data that is to be used, section 3 is about the methodology followed to handle the data, section 4 is the analysis of the data that includes some visualisations, section 5 contains the results.

Following that, section 6 is for the conclusions, section 7 for future work, 8 for references and 9 for appendices (including initial project proposal and reflective journals).

## 2.0 Data

### 2.1 Classical Book Data

The data for classical books was sourced from the Project Gutenberg website.

R package called 'gutenbergr' was used for the sourcing of the data as it provides many functions for doing so. Some of the functions that can be used with the 'gutenbergr' package are outlined and summarised in table 1 below.

To summarise the gutenbergr package, it helped me to gather data such as the Gutenberg id (each book has a unique one), authors, subjects of each book, language of the book, text, etc. The data gained was saved into csv files for usage later in the project.

Table 1 Gutenbergr package functions

Function	No of Attributes	Description	Attributes
<code>gutenberg_metadata</code>	8	contains the metadata for all the available project Gutenberg books	<code>gutenberg_id</code> , title, author, <code>gutenberg_author_id</code> , language, <code>gutenberg_bookshelf</code> , rights, <code>has_text</code>
<code>gutenberg_works()</code>	8	Same as above, but with more filtering. No duplicates, only english books and books that are available for downloading	<code>gutenberg_id</code> , title, author, <code>gutenberg_author_id</code> , language, <code>gutenberg_bookshelf</code> , rights, <code>has_text</code>
<code>gutenberg_download(book id, meta_fields etc.)</code>	3	Downloads a book by taking in the <code>gutenberg_id</code> as an input	<code>gutenberg_id</code> , text, title
<code>gutenberg_subjects</code>	3	information about the genres/topics of books based on the <code>gutenberg id</code>	<code>gutenberg_id</code> , <code>subject_type</code> , subject
<code>gutenberg_authors</code>	7	information about the authors available on gutenberg	<code>gutenberg_author_id</code> , author, alias, birthdate, deathdate, wikipedia, aliases

Only the data that was suitable for the project was saved – one of the main requirements being that the book had to have a downloadable text and it had to fall within one of the many 'fiction' genres. At the same time, only books that were marked as being in English were kept.

The filtered "gutenberg\_subjects" and "gutenberg\_works" datasets were merged to take advantage of the fact that both had some of the attributes that I would be needing but not

all of them. After merging, the relevant attributes were moved into a new, final table that would become the dataset that I would use in the project the most. Table 2 visualises the final dataset of classical books.

*Table 2 Final Classical Books Dataset*

No of Attributes	Description	Attributes	No of rows
10	the table with the data relating only to the relevant books (english and fictional)	gutenberg_id, title, author, gutenberg_author_id, language, gutenberg_bookshelf, rights, has_text, subject_type, V1	13815

## 2.2 Modern Book Data

Sourcing the data for modern books was a bit more tedious – requiring to first manually compile a list of ISBN numbers that could then be used with the Python code to extract the data (such as title, author, summary) from the Google Books API. The ISBN numbers used were for some of the top books from recent years (from around 2020-2010). Table 3 summarises the final modern books dataset.

*Table 3 modern Books Dataset*

No. of Attributes	Description	Attributes	No. of Rows
3	Modern Books data, gained through Google Books API using ISBN numbers	Title, Summary, Author	505

In most cases, the books had more than one ISBN number but not all of them were recognised by Google Books API. In some cases, the first ISBN number I tried did work, but oftentimes it did not – which meant I had to insert a different version of the ISBN number for the specific book and hope that the second one will work. Although it did not happen with a large portion of the books, in some cases I still did need to go through more than 5 ISBN numbers before I found one that was recognised.

The summaries gained were short – only a couple of sentences, none exceeding even 50 words. In addition, some of them were not useful at all – with summaries that gave no indication on what the book was about. With the dataset of approximately 500 modern books with summaries that I had created however, even if only half proved to be useful that would still be enough for the current scale of the project.

## 3.0 Methodology

The methodology used for the analysis of the data was KDD (knowledge discovery in databases) which has several distinct steps – selection of the data, pre-processing, transformation, data mining, interpretation/evaluation. The main goal of using this methodology is to find useful information or patterns within the data.

- [Dataset Selection and Description](#)

The first step of KDD, the data selection, was described in more detail in section 2 of the report. To summarise it, the data used was the book data from the Project Gutenberg website and modern book data sourced from the Google Books API.

- [Pre-processing and Transformation](#)

After gathering all the necessary data (on both classical and modern books), further pre-processing and transformation steps followed. Besides the steps done earlier (such as filtering books by subject and combining datasets), more needed to be done for the datasets to be ready for modelling.

For classical books, downloading the book text was necessary. After that, the pre-processing and transformation steps done for both datasets (classical and modern books) were relatively the same.

For the textual data to be more accurate during modelling stages, clean-up of the text was needed. Stopwords were removed and TF-IDF (Term Frequency Inverse Document Frequency) was used to drop the least relevant words.

It was only after removal of stopwords and TF-IDF that a DTM (document-term matrix) was made with the remaining words, which was to be used for each of the models further in the project.

- [Data Mining and Evaluation](#)

The very first and main model used in my project was the LDA (Latent Dirichlet allocation) which is a topic modelling technique that assigns words to different topics. Through this method, the books themselves can also be assigned to different topics – in this way, the books that fall within the same topic are bound to be somewhat similar. For the sake of comparison K-means and SOM (self-organising maps) clustering algorithms were also used and the comparison itself was done using the Jacquard Distance method.

The last part of the project was to make a recommendation engine. The process of making the content-based recommendation system was simple (in terms of approach) but relatively

effective as it made use of the LDA model (the assigned topics) and recommended books based on how similar the classical books were to the 'favourite' books of the user.

For creation and testing of the recommendation system, a dummy user was made by taking a randomised sample of  $n$  books from the list of modern books. The  $n$  value can be replaced with any number and the code would still work as the essential parts of the code used *for loops* instead of having a fixed value for the number of favourite books.

### 3.1 Implementation

Due to RAM limitations (even after moving the code to Google Colab), the classical book dataset was split into 5 parts and all the models were done with all 5 of the splits.

The main part of the pre-processing was the removal of stopwords. In addition to removing the already existing stopwords, common names and a customised stopwords list (including the less common names found in the classical books) were also removed.

The TF-IDF (Term Frequency Inverse Document Frequency) method was used to determine the relevance of the words that remained, only words that had a TF-IDF score of more than what was the average were kept.

TF-IDF is a method that evaluates the importance of each word within a document in relation to a set of documents. For this, it uses two measures; first one is the number of times a word appears within a document and then the inverse document frequency of the same word across the full set of the documents. The TF-IDF scoring is useful for ruling out the words that appear in a document often but are still not relevant for the analysis – if a specific word appears often not within one document but within many (or all) of them, then the TF-IDF score would be lower.

In the case of my project, the documents are instead books (book text), but the process is still the same. By using TF-IDF before a document term matrix (DTM) is made, it will become less likely that unimportant words (ones that would be useless for further analysis) will remain. After each of the words was given a TF-IDF score, the average score was taken and only the words that had a score above the average were kept.

Although the initial list of classical books was split into 5 samples, those samples still proved to be too big for some of the models to finish with the available RAM. To bypass the issue, instead of using a full sample, I instead randomly sampled a thousand books from each of the 5 datasets and used those for the modelling. Which means that in total, there were 5 sets of one thousand classical book titles that were then used for all the models.

After the words below the average TF-IDF score were removed, a DTM (document term matrix) was made. A DTM is a sparse matrix that describes the frequency at which words occur within the set of books. Sparse matrix means that within the DTM, there would be many values of 0.

Table 4 below visualises a small part of a matrix, where the columns (numbers 1,2,3) represent the different books and the rows (a unique word on each of them) represent the words that exist within the books, all sorted in alphabetical order.

The intersections between the rows and columns represent how many times the word (from the row chosen) appeared within the text of a book represented by a specific column. For example, from table 4, the word ‘abandon’ can be found 8 times in text 1, and 0 times in texts 2 and 3.

*Table 4 Small part of a word matrix*

	<b>1</b>	<b>2</b>	<b>3</b>
<b>abandon</b>	8	0	0
<b>abandoned</b>	12	0	1
<b>abandoning</b>	2	0	1
<b>abandonment</b>	4	0	0
<b>abase</b>	1	0	0
<b>abacement</b>	3	0	0
<b>abashed</b>	1	0	1

Using DTM, three different models were implemented – LDA, K-means, SOM.

LDA (Latent Dirichlet Allocation) is a topic modelling technique that discovers topics within a collection of documents (books) and then assigns each of them towards a particular topic. The underlying concept is that LDA treats documents as mixture of topics and each topic as a mixture of words. That allows the documents to overlap with each other in terms of a topic instead of each one being treated as a separate topic. Each document, or book in this case, can have more than one topic within its content, however it was the most prevalent of the topics that was in the end assigned for each of the books to represent them.

For LDA modelling, there was a value of ‘k’ that could be assigned, which would correspond to the number of topics the LDA model would produce at the end. After trying out a few different values for the number of topics for the LDA model, I settled on 9 as that seemed to bring back the most consistent results.

The same number of clusters was to be used for the K-means and SOM (self-organising map) models as well, just to have consistency between the three models.

The final csv file containing the titles of the classical books and the LDA topics assigned for each of them was saved for later use. A sample of such can be seen in table 5 below.

Table 5 LDA topics sample

title	LDAtopics
A Certain Rich Man	6
A Connecticut Yankee in King Arthur's Court, Part 4.	8
A Daughter of the Land	1
A Doctor of the Old School — Volume 5	1
Alice of Old Vincennes	5

Clustering is a very common data analysis technique that is used for the exploration of data and to get a grasp on the structure of the data. It identifies groupings of data based on how similar the different data points (or in the case of this project, books) are.

Kmeans is a clustering algorithm that takes in a value of 'k', corresponding to the number of clusters it should identify (the value of k, as mentioned above, was kept at 9 throughout the project). Each data point belongs to only one cluster and, the data points within the same cluster, tend to be more related/similar to each other than data points in other clusters.

Table 6 below visualises the final csv document output, containing the titles and the K-means cluster numbers.

Table 6 Sample of Kmeans Clusters

title	KmeansClusters
A Certain Rich Man	3
A Connecticut Yankee in King Arthur's Court, Part 4.	3
A Daughter of the Land	3
A Doctor of the Old School — Volume 5	3
Alice of Old Vincennes	3

The SOM (Self-organizing map) model is also used for clustering and exploration of data. It can be considered as a type of artificial neural network which is trained through unsupervised learning. For SOM model, first a grid must be defined and then the SOM model plots and sorts (clusters) the data points on the grid.

As a post-SOM-processing step, k-means was executed by using SOM results, producing a dataset of book titles with a SOM cluster topic assigned for each of them. Table 7 is an example of how the final .csv file with SOM clusters would look like.

Table 7 SOM clusters example

title	SOMclusters
A Certain Rich Man	7
A Connecticut Yankee in King Arthur's Court, Part 4.	7
A Daughter of the Land	3
A Doctor of the Old School — Volume 5	3
Alice of Old Vincennes	7

In a separate file, the same type of pre-processing and modelling was done with the list of modern books and their summaries.

It was the data extraction of the modern books instead that was very different from the processes that were done to the classical books dataset. The extraction of the summaries, using the Google Books API, was done using Python and referencing multiple tutorials and google books API documentations. The data extracted in the end was the titles, authors, and summaries of the books. There was more data that could have been extracted, however I deemed it unnecessary (things such as page count, language, etc).

The recommendation system, as a final step of the project and the one that would contribute the most towards achieving the aims of the project, was done last. The main steps of how the recommendation system was implemented can be seen in image 1. Below the image, each of the steps will be explained in more detail.

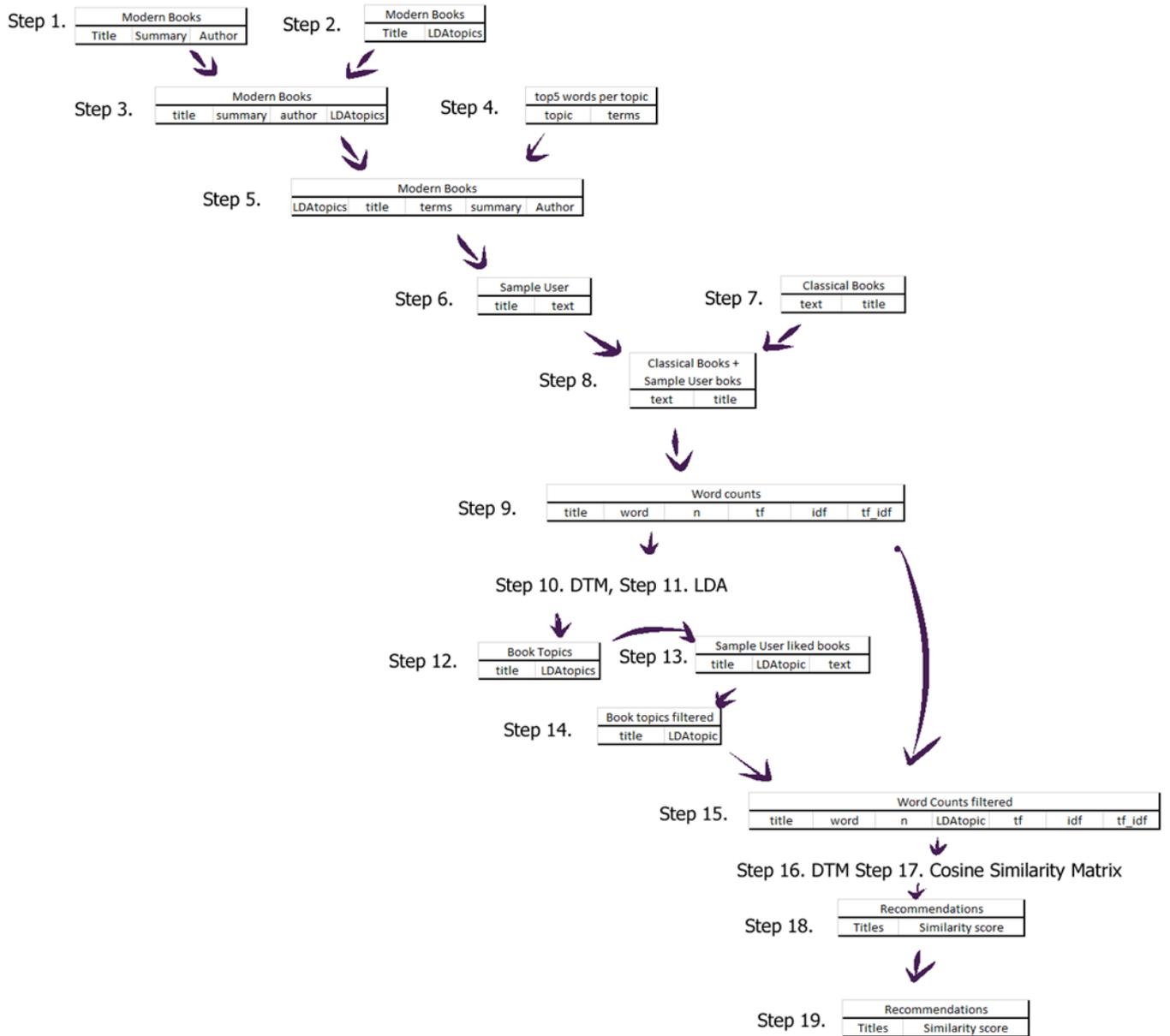


Image 1 Step-by-step of Recommendation System's process

**Step 1.** The dataset of the modern books was imported, containing attributes such as title, summary, author.

**Step 2.** The dataset of the LDA topics assigned for the modern books was imported, containing two attributes: title and LDA topic number.

**Step 3.** The two datasets were merged into one, now having 4 attributes (title, summary, author, LDA topic number).

**Step 4.** The dataset with top 5 words for each of the LDA topics (gained from modern books) was imported.

**Step 5.** The top words for each of the topic were merged into a single cell (for example, top 5 words of topic 1 went into one cell, top 5 words from topic 2 into another cell, etc.). Then, those top words were merged with the modern book dataset, using LDA topic number as the common value. The result was a dataset where, each modern book had an extra attribute assigned to them – that attribute contained the top words corresponding to the LDA topic assigned for the book.

Typically, that would not be an essential step, however in many cases the summaries that the modern books had were very short and insignificant which made me think that it may be a good idea to add a few of the top words that related to the LDA topic they were assigned. In this way I was trying to make sure that the modern book summaries would not become completely 'insignificant' within the whole process of getting the recommendations.

**Step 6.** A sample user was created on this step. An n number of modern books from the dataset gained after step 5 were added into a separate dataset. The books contained in this sample user dataset are replacing what would typically be a user's liked book list.

**Step 7.** The classical book dataset (a sample of one thousand books) was imported, their text downloaded, and pre-processing started. Only the text and title attributes are kept.

**Step 8.** The classical books dataset and the books contained within the sample user dataset were merged into one dataset. Only two attributes remain in this dataset: title and text.

**Step 9.** Further pre-processing is done with the merged dataset, such as splitting the text into single words and applying the TF-IDF method to remove the words with the TF-IDF score below average.

**Step 10.** A document term matrix(dtm) is done with the fully pre-processed and cleaned dataset.

**Step 11.** The dtm is used to create an LDA model.

**Step 12.** LDA topics are extracted from the LDA model, creating a dataset with two attributes: title and LDA topic assigned to the book.

**Step 13.** The sample user's liked books (the ones that were merged with classical books in step 8) are extracted into a separate dataset to check what LDA topics were assigned to them.

**Step 14.** Based on what LDA topics were assigned to the user's liked books, the classical books with matching LDA topics are extracted into a new dataset for further use. For example, if there were 3 books in the sample user's liked books, one with an LDA topic of 3 and the other two with the topic 6, then all the classical books containing topics 3 and 6 would be put into the new dataset.

**Step 15.** The words correlating to each of the books are added back in, essentially merging the dataset from step 9 and the dataset from step 14, while keeping only the book titles present in the dataset from step 14.

**Step 16.** A dtm (document-term matrix) is done again, this time with the dataset from step 15 which is smaller than before (because of the fact it contains only the classical books that had the same topic as one of the user's favourite modern books).

**Step 17.** A cosine similarity matrix is done with the second dtm, it gives similarity scores for each of the books, and it is the main thing that will decide what books manage to get into the final recommendations.

**Step 18.** Each of the books within the user's favourites list go through a for loop and the 5 classical books with the highest similarity score are put into a 'recommendations' dataset. The for loop continues until there are no more books in the user's favourites list.

For example, if there were 3 books in the favourites list, the for loop would go through the code 3 times and by the end the 'recommendations dataset would have 15 (3x5) books in it.

**Step 19.** The books in the recommendations list are sorted based on their similarity score and the top 5 with the highest similarity score are kept for the final recommendations.

### 3.2 Important R functions Used

- ◆ Read.csv()
- ◆ Write.csv()
- ◆ Subset()
- ◆ Merge()
- ◆ As.data.table()
- ◆ Sample\_n()
- ◆ Set.seed()
- ◆ Rbind()
- ◆ Group\_by()
- ◆ Filter()
- ◆ Count()
- ◆ Cast\_dtm()
- ◆ LDA()
- ◆ Tidy()
- ◆ Arrange()
- ◆ Top\_n()
- ◆ Ungroup()
- ◆ As.matrix()
- ◆ Order()
- ◆ Slice()
- ◆ Kmeans()
- ◆ Mutate()
- ◆ Somgrid()
- ◆ Som()

### 3.3 Important R Packages Used

- ◆ Gutenbergr()
- ◆ Curl()
- ◆ Dplyr()
- ◆ Ggplot2()
- ◆ Data.table()
- ◆ Tm()
- ◆ Topicmodels()
- ◆ Tidytext()
- ◆ Tidyrr()
- ◆ Stringr()
- ◆ Scales()
- ◆ Reshape2()
- ◆ Rmarkdown()
- ◆ Tidyverse()
- ◆ Factoextra()
- ◆ Kohnen()
- ◆ Shiny()

## 3.4 R Shiny

At the very end, R Shiny was used for creating a simple user interface (UI) through which the recommendation system could be explored. The UI can be seen in image 2 below and the features of it are explained below the image.

### Recommendation System



Number of books in Favourites list:

Number of books in Classical book list:

Number of LDA model topics:

### Summary of the sample dataset

```
gutenberg_id      title          author          gutenberg_author_id
Min.   : 840      Length:20      Length:20      Min.   : 45
1st Qu.:15028    Class :character Class :character 1st Qu.: 1298
Median :25417    Mode  :character Mode  :character Median : 4970
Mean   :21926
3rd Qu.:31764
Max.   :35493
language          gutenberg_bookshelf rights          has_text
Length:20         Length:20         Length:20         Mode:logical
Class :character  Class :character  Class :character  TRUE:20
Mode  :character  Mode  :character  Mode  :character

subject_type      V1
Length:20         Length:20
Class :character  Class :character
Mode  :character  Mode  :character
```

### Sample User's Favourites

title	Author
Last Sacrifice	Richelle Mead
Difficult Women	Roxane Gay

### Recommended classical books

titles	Similarity
Lewie Or, The Bended Twig	0.05
The Governess; Or, The Little Female Academy	0.04
The Governess; Or, The Little Female Academy	0.03
Waste Not, Want	0.02
Lewie Or, The Bended Twig	0.02

Image 2 UI of the recommendation system

The sidebar in the UI contains 3 fields with values that can be changed by the user: the number of modern books within the favourite's list; the number of classical books to perform the analysis on; the number of LDA topics (the value of k for the LDA model). The default values were kept low to reduce the load time as, with bigger values (especially in terms of classical books), it would take longer for the results to load.

With the default values it takes a couple of minutes, but the larger the sample of classical books that is to be used, the longer it would take for the results to be displayed.

The main section of the UI contains the 3 sets of the results: at the very top a summary of the classical books dataset that is to be used (including only the number of books specified by the

user in the sidebar); Below that are the modern books contained in the favourites list – the number of books in the list correspond to the number chosen by the user in the sidebar.

The very last section of the results is the recommendations themselves with the similarity score displayed alongside them. With smaller classical book samples used the similarity score would, more often than not, be small as well. The larger the classical book sample the more likely it is for the similarity score to be higher, however the computation time would be longer then as well.

### 3.5 Things That Did Not End Up in The Final Version of the Project

At one point during the process of working on the project, Goodreads was considered as an option for extracting book summaries and reviews. Goodreads is a website where people can rate and review the books they have read and it has an extensive database of all kinds of books – fictional and not, recent, and old. There was not a specific reason for dropping Goodreads from the project, besides the couple restrictions they had with their API and the fact that Google Books API was more straightforward to work with.

In the same way, Amazon book reviews were once under consideration for analysing the modern books. Just like with Goodreads, the idea was dropped for the same reason – the Google Books API was more straightforward, which was better for the project overall as I did not want to spend weeks of time just gathering the data when the end goal is analysing book data and not just sourcing it.

SVD (singular value decomposition) is yet another thing I have looked into. SVD is a technique for decomposing a matrix into many component matrices which can help discover some useful or interesting features of the initial matrix. Although I have researched and even coded up some SVD methods, in the end those did not contribute towards the final results of the project (the recommendations).



Since the size of the image is quite big, the smaller words can be hard to see so I have cropped a few parts out and enlarged them so the words can be seen more clearly. (Images 4, 5, 6).

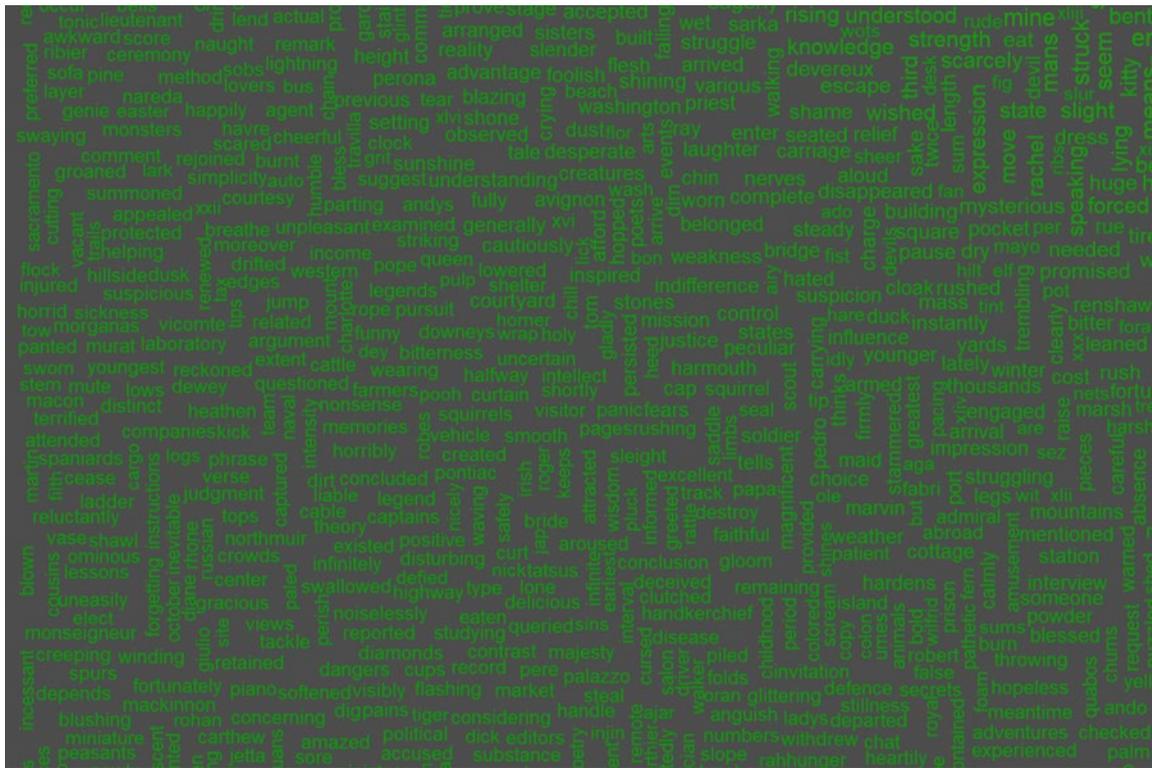


Image 4

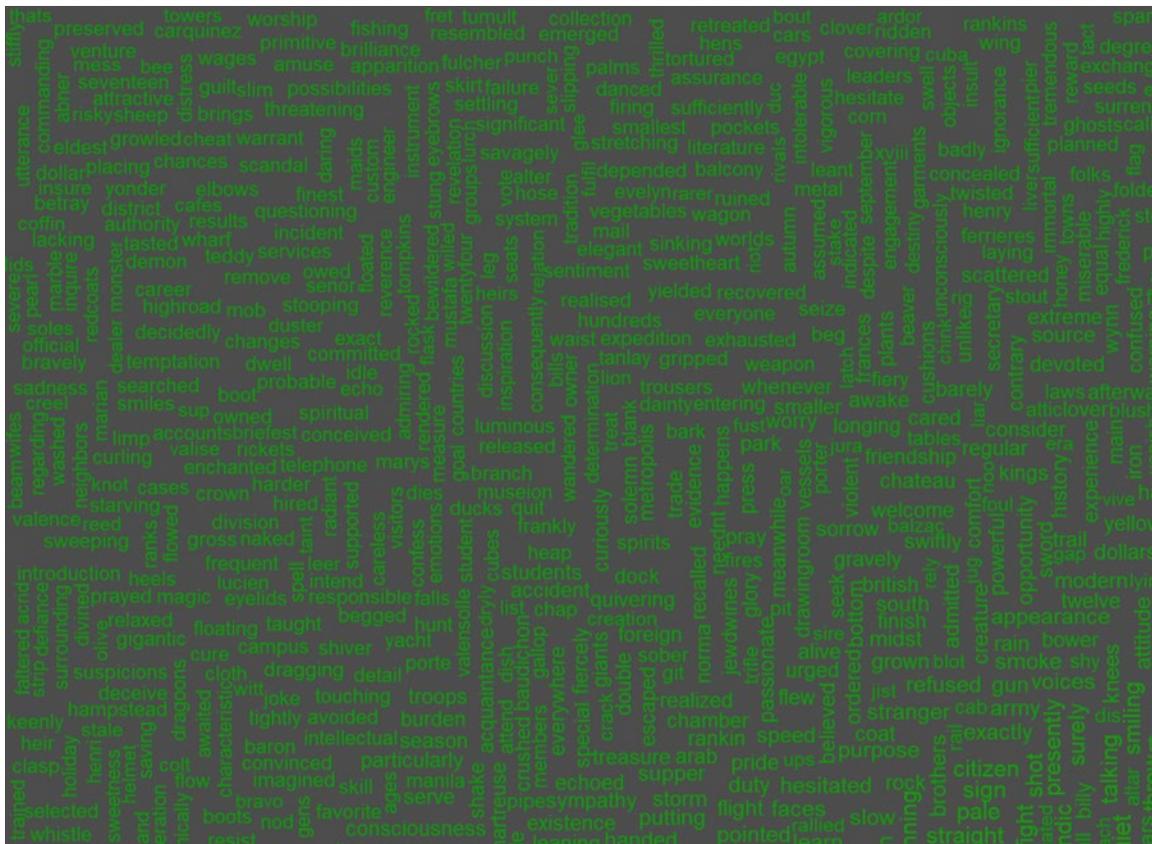


Image 5





## 4.2 Top Words per Topic

Using the LDA model, top 5 words per topic (with 9 topics in total) were extracted. Once again, for comparison, first are the results with the default stopwords removed (images 9, 10, 11) and then with further custom stopwords removed (images 12, 13, 14).

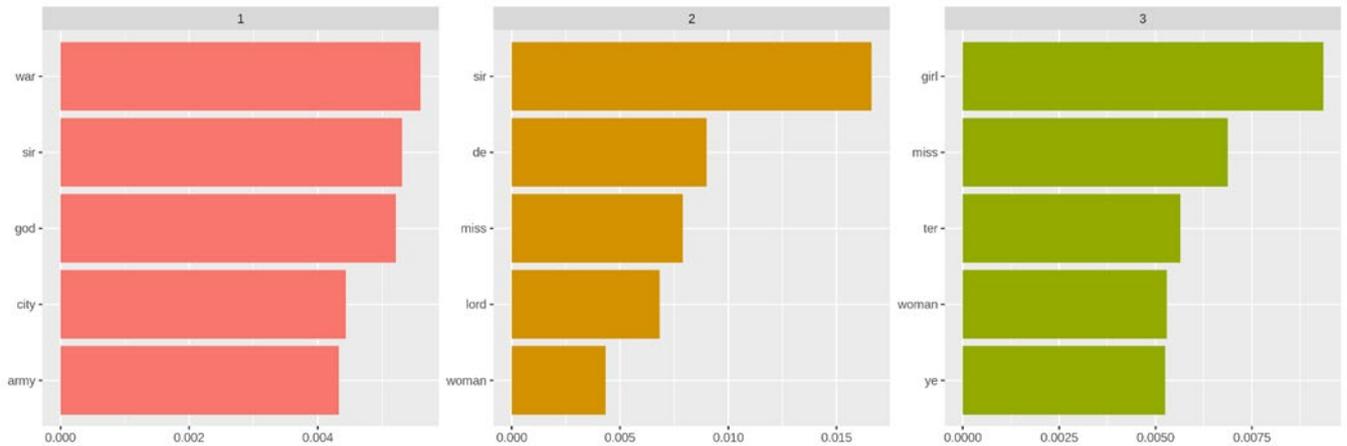


Image 9

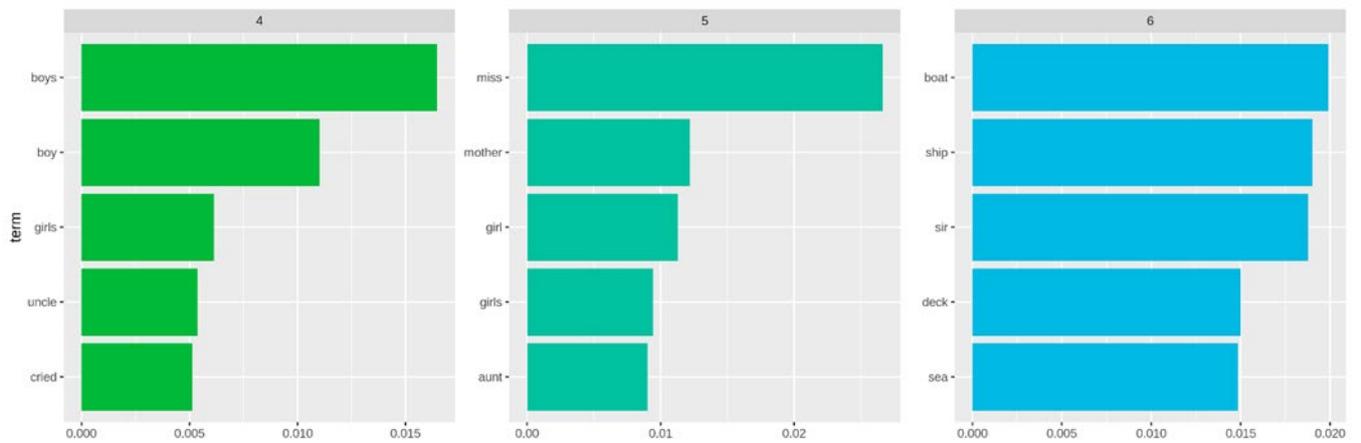


Image 10

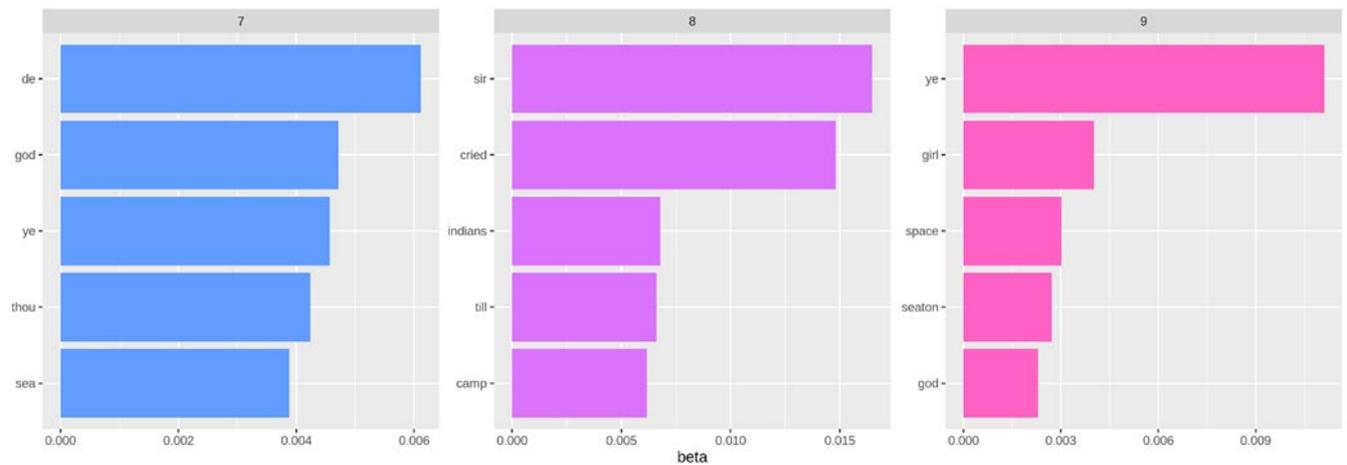


Image 11

To interpret the images: Each block contains top 5 words relating to a specific topic (numbered 1 to 9 and labelled at the top of the block). On the y-axis the top 5 words are displayed, while the x-axis represents how often the word appeared within the topic.

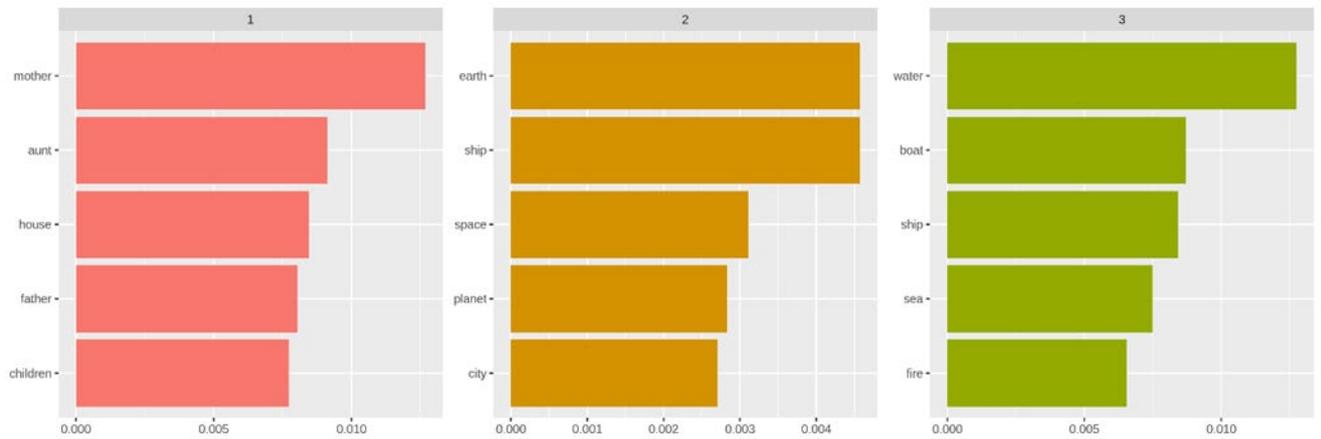


Image 12

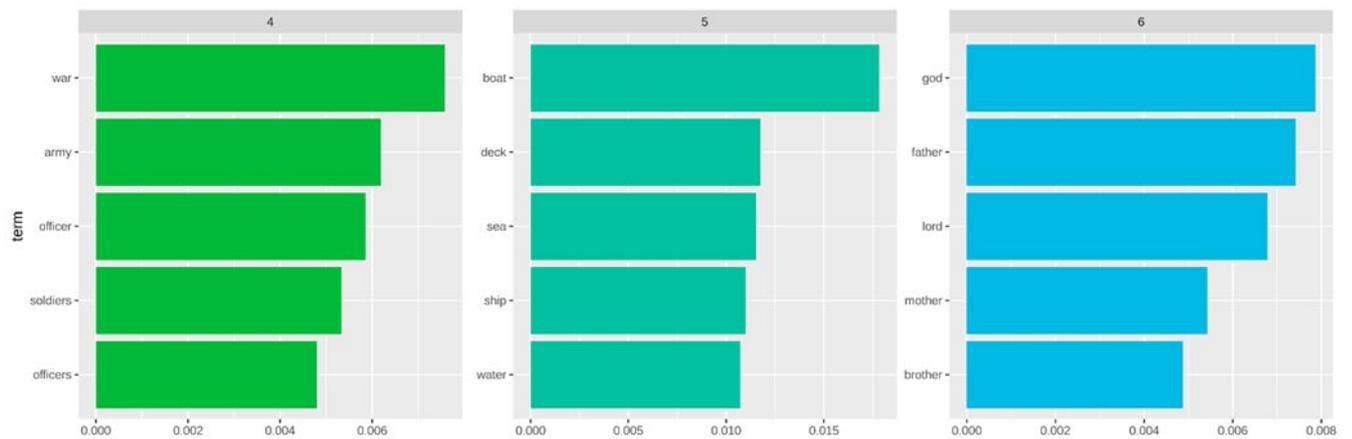


Image 13

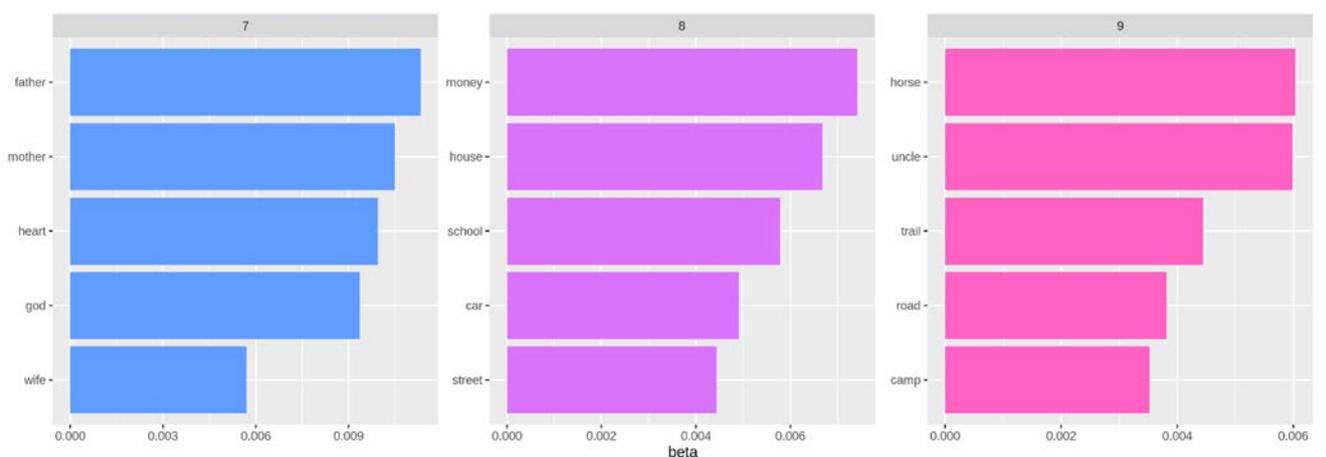


Image 14

In the first 3 images, there are still words that do not make the topics any clearer – such as “de”, “ter”, “ye”, etc. while the latter three images make the topics a lot clearer to see with

words such as “sea”, “officer”, “war”, “boat”, “camp” etc. Table \* below visualises the key/top words of the 9 topics and provides an interpretation for each.

*Table 8 interpretations of topics*

<b>Topic</b>	<b>Keywords</b>	<b>Interpretation</b>
1	mother, aunt, house, father, children	Family oriented
2	earth, ship, space, planet, city	Sci-fi
3	water, boat, ship, sea, fire	Maritime
4	war, army, officer, soldiers, officers	War/military
5	Boat, deck, sea, ship, water	Maritime
6	God, father, lord, mother, brother	God/religion
7	father, mother, heart, god, wife	Family
8	money, house, school, car, street	School oriented
9	horse, uncle, trail, road, camp	Adventure

Although some of the interpretations for the topics ended up the same, the more nuanced topics within the books would most likely not be – for example topic 1 would be more about the whole family such as parents, kids, relatives, etc. while topic 7 is more likely to be oriented around a newer family – a wife and a husband, their relationship, possibly parenthood at some stage.

Topics 3 and 5 had similar keywords (and interpretations) as well and although it is hard to know specifically how those would differ from just looking at the keywords, some guesses can still be made. One of them could be related to sea travel, while the other one about fishing which would, in the end, result in a lot of sea-related words.

### 4.3 SOM, LDA, Kmeans models

After running the three models (SOM, LDA, Kmeans), some visualisations were done to better see the topic distribution.

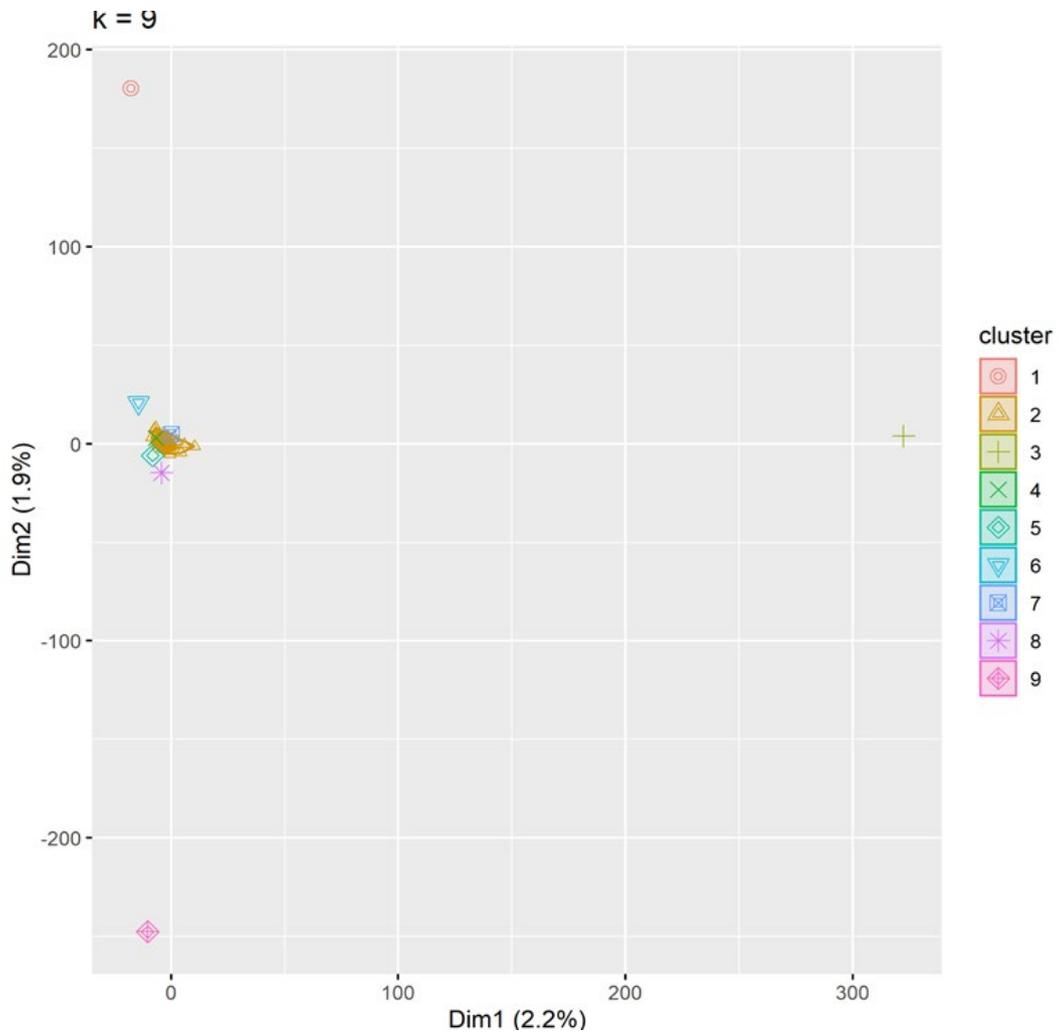


Image 15 Kmeans cluster Distribution

Image 15 shows the Kmeans model cluster distribution, with 9 existing clusters. Each different symbol (shown in the legend on the right side) represents the different topic and each of the books from the sample ran have one symbol attached to it – one that corresponds to the topic that was assigned to it. Many of the books were assigned to topic 2 in the case of this particular book sample, which is why there is a bigger orange cluster on the left side of the graph.

Image 16 is the training progress of the SOM model, it shows how, with each new iteration, the books within a cluster became more and more similar to each other. The model was run through 100 iterations, however as seen in the image, by 60th iteration around 80% of the reduction to the mean distance between units (books) had happened. That means that even if the model were to be reduced to 60 iterations, the resulting clusters would be approximately 80% more accurate than if there was only 1 iteration.

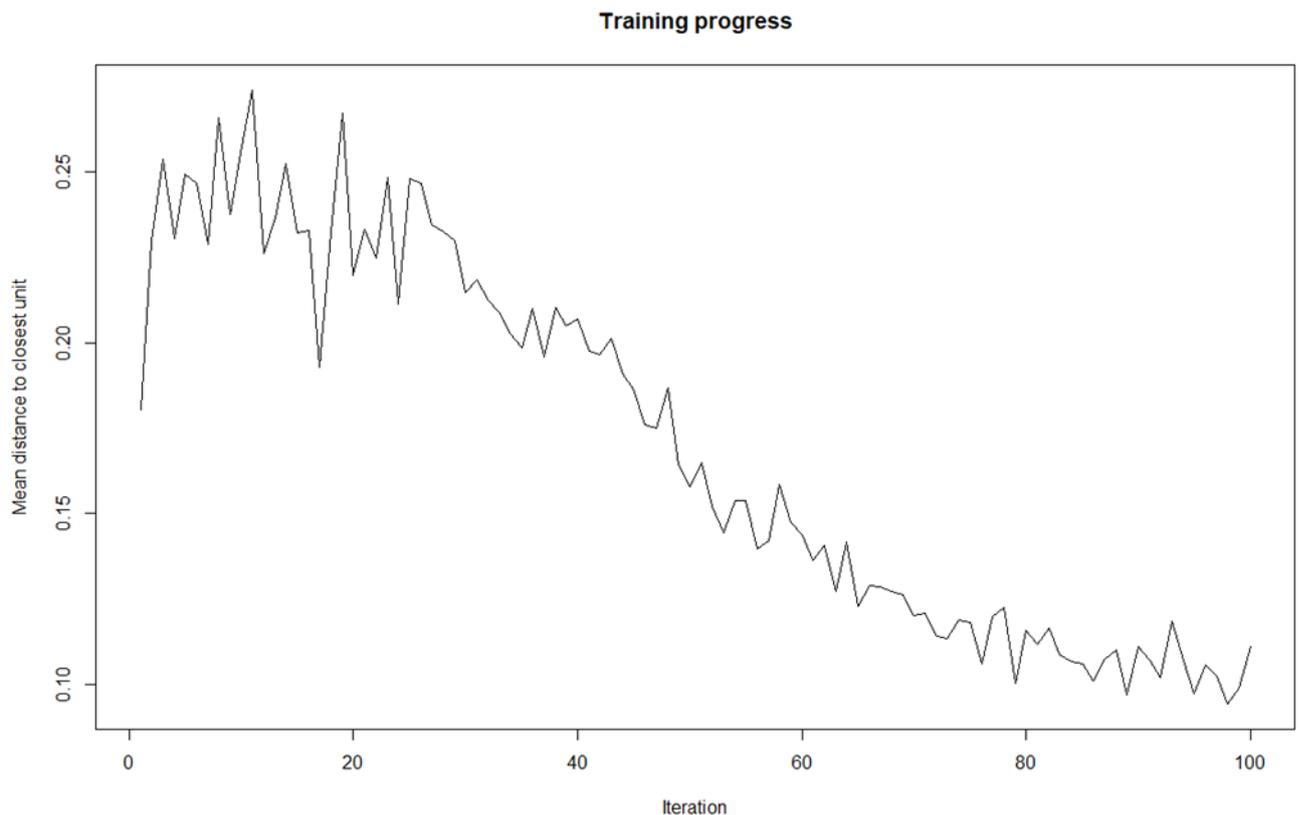


Image 16 SOM model Training Progress

To compare the three models, I have made bar charts for the results gained (cluster distribution) after running the three models on the same sample of 100 books. The bar charts can be seen in images 17, 18, 19 below.

The x-axes on the images represent the cluster (numbered 1 to 9 for the total of nine clusters), while the x-axis represents the number of books that fell within/were assigned that topic.

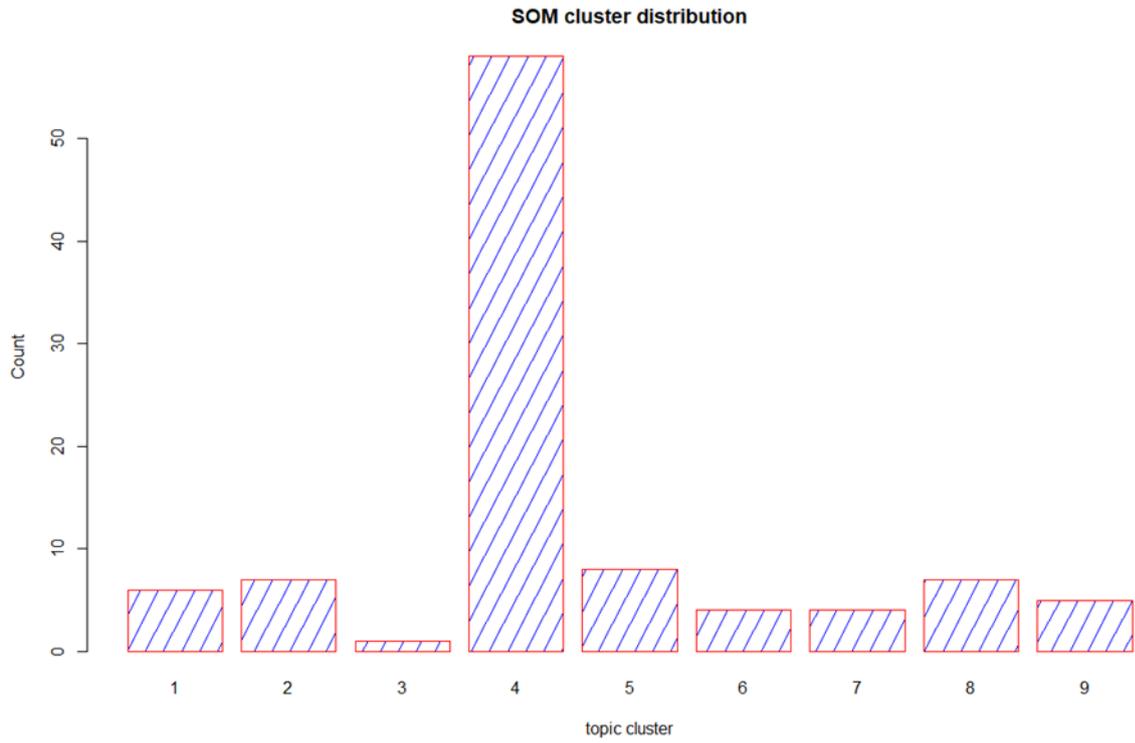


Image 17 SOM cluster distribution

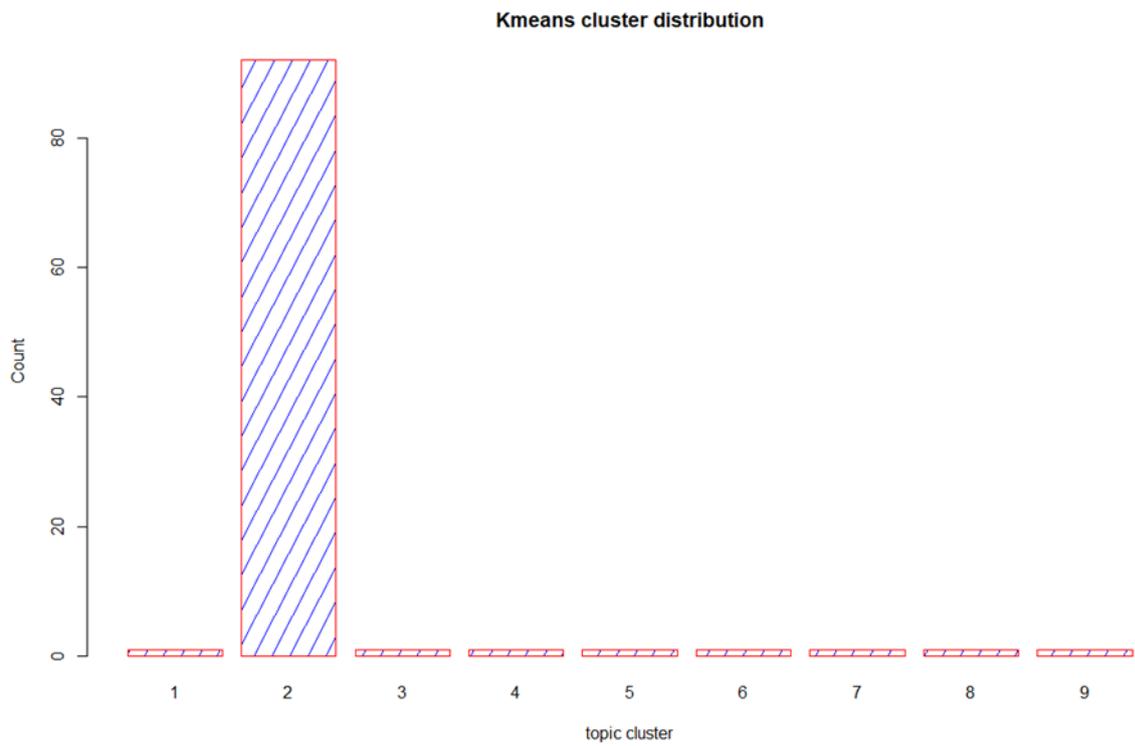


Image 18 Kmeans cluster distribution

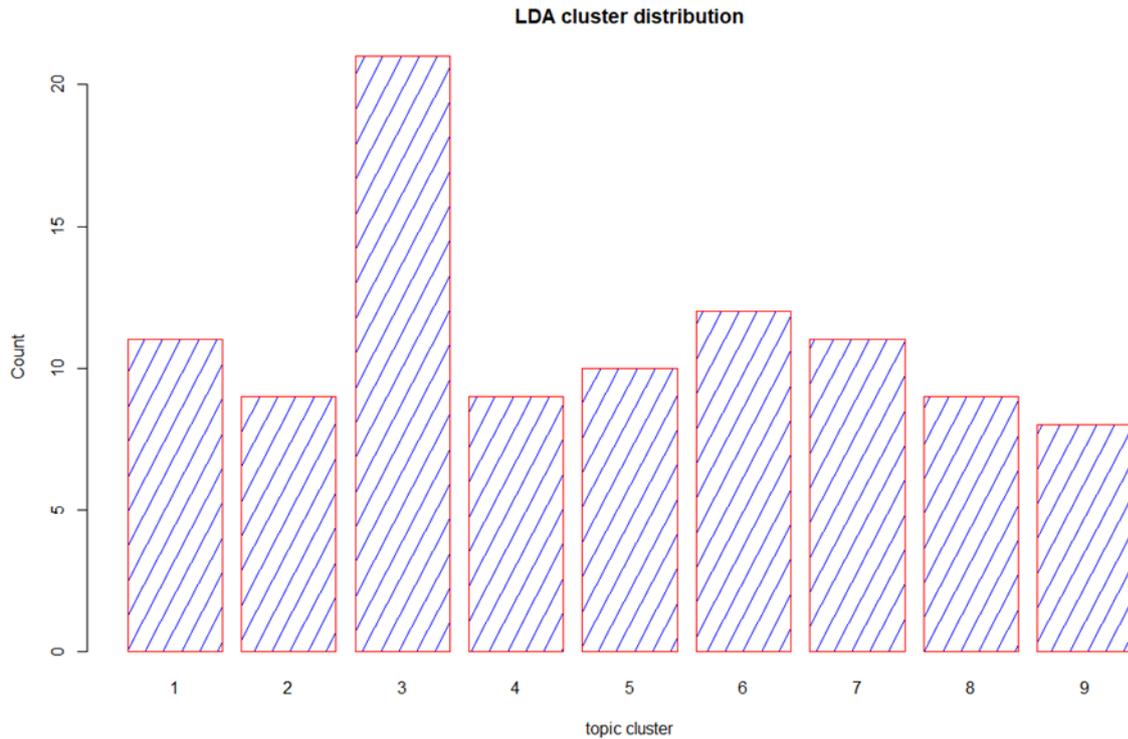


Image 19 LDA cluster distribution

By simply looking at the three images, it can be determined that the Kmeans model would be the least accurate while the LDA would be the most accurate.

That conclusion is based on the overall distribution of books between the topics and, although in some cases there may be a lot of books that fall within the same topic, it is unlikely that with an accurate model a bar chart such as that of the Kmeans model would appear.

After checking the results in the table format, it became clear that the Kmeans model assigned 1 book to each of the clusters besides topic 2 which gained all the rest of the books.

The SOM model's cluster distribution looks a lot better when compared to Kmeans, however overall, the LDA model has the most even distribution and therefore I would be more likely to trust the accuracy of its results.

Table 9 below is a side-by-side comparison of 5 classical books and what topics they were assigned to within each model. The results were taken from the tables that were produced after the three models were run on a classical book sample of one thousand books.

Overall, the Kmeans model assigned all five to one cluster, SOM to two and the LDA model had a total of 4 clusters for the 5 topics.

Table 9 Cluster Assignment comparison

title	LDAtopics	SOMclusters	KmeansClusters
A Certain Rich Man	6	7	3
A Connecticut Yankee in King Arthur's Court, Part 4.	8	7	3
A Daughter of the Land	1	3	3
A Doctor of the Old School — Volume 5	1	3	3
Alice of Old Vincennes	5	7	3

## 5.0 Results

For the recommendation system, a sample user was created with a list of 5 favourite modern books which would then be used to get the classical book recommendations. Table 10 presents the list of the favourite books.

Table 10 Sample User's Favourite Books

Sample User's favourites	
Title	Author
City of Fallen Angels	Cassandra Clare
A Monster Calls	Patrick Ness, Siobhan Dowd
I Let You Go	Clare Mackintosh
The Fault In Our Stars	John Green
The Storyteller	Jodi Picoult

The code was run 5 times, each time with a new sample of 1 thousand books, producing 5 sets of recommendations (Tables 11, 12, 13, 14, 15). The results are in the format of a title followed by a similarity score – the similarity score is based on how similar the content within the classical book is to the content (summary) of the modern books within the sample user's favourites list.

Table 11 Sample 1

Sample 1 Results		Sample 2 Results	
Title	Similarity score	Title	Similarity score
Greyfriars Bobby	0.2020951	Tom Tufon's Travels	0.2256962
The Picture of Dorian Gray	0.1908218	His Grace of Osmonde...	0.2076221
Mother	0.1870958	City of Endless Night	0.187871
Laddie: A True Blue Story	0.1836869	Tales and Novels Volume 06	0.1742
Dangerous Days	0.1812732	Mary Minds Her Business	0.166781

Table 12 Sample 2

Table 13 Sample 3

Sample 3 Results		Sample 4 Results	
Title	Similarity score	Title	Similarity score
Kid Wolf of Texas A Western Story	0.1812355	The Enchanted Barn	0.1986325
The Outcry	0.1732205	Kid Scanlan	0.1905182
Rebels of the Red Planet	0.1650904	Rick and Ruddy: The Story of a...	0.1838232
The Boy Ranchers on Roaring River..	0.1600539	Thy Name Is Woman	0.1792555
Code Three	0.1599315	Exile from Space	0.1779391

Table 14 Sample 4

Sample 5 Results	
Title	Similarity score
The Speedwell Boys and...	0.2008913
Frank in the Woods	0.1924015
Mildred Keith	0.180011
The Burning Secret	0.1726189
At the Sign of the Sword...	0.1641272

Table 15 Sample 5

The final 5 recommendation from the samples were then combined at the end to get the 5 recommendations with the highest similarity scores (Table 16).

All 5 Results combined	
Title	Similarity score
Tom Tufton's Travels	0.2256962
His Grace of Osmonde...	0.2076221
Greyfriars Bobby	0.2020951
The Speedwell Boys and...	0.2008913
The Enchanted Barn	0.1986325

Table 16 Results from samples combined

A similarity score of 1 would mean that the books are identical, while a score of 0 would mean that they have nothing in common. As seen from the results in tables 5 – 10, some of the classical books had a similarity score of 0.2, while many others were in the range between 0.19 – 0.15. Overall, having a 0.2 similarity score would mean that 20% of the words within the classical book match the words within one of the modern books in the favourites list. And that 20% is without the existence of stopwords (that would be common among all the books) and with less relevant words reduced through TF-IDF, which would have reduced the similarity somewhat.

Knowing whether the recommendations are reasonably accurate is not easy though – the only thing that can be done is looking at what the books are about and deciding from that. Looking at the summaries and reviews for the modern books and the 5 classical books in table 14, a few clear themes (genres) can be seen between the two – romance, family, adventures, some mysteries, and fantasy mixed in as well though more on the side of the modern books. As far as can be seen from that type of a check, the recommendations did seem to be valid.

## 5.1 Testing

Some testing was done to check how well the models and the recommendation system work. The test cases are outlined below.

### 1. LDA Model – Topic assignment consistent

**Description:** With the same set of books used for the LDA model, those books should be assigned into the same clusters of topics.

**Precondition:** The set of books went through the pre-processing steps and a dtm was made from them.

**Assumption:** The same set of books is used.

#### Testing Steps:

- a. *Install and load the required packages.*
- b. *Load in the classical book dataset.*
- c. *Set.seed(20) and then subset a number of books to be used for the testing.*
- d. *Run the code for downloading the book text.*
- e. *Run the code for pre-processing and stopwords removal.*
- f. *Run the TF-IDF code.*
- g. *Create a DTM (document-term matrix).*
- h. *Run the code for building the LDA model with 9 topics (k=9).*
- i. *Extract the topics assigned for the books, one main topic for each classical book.*
- j. *Save the results.*
- k. *Repeat multiple times and compare the results.*

**Expected result:** The books should be assigned to the same topic clusters each time the code is executed.

**Actual result:** As expected.

### 2. SOM Model – Topic assignment consistent

**Description:** With the same set of books used for the SOM model, those books should be assigned into the same clusters of topics.

**Precondition:** The set of books went through the pre-processing steps and a dtm was made from them.

**Assumption:** The same set of books is used.

#### Testing Steps:

- a. *Install and load the required packages.*
- b. *Load in the classical book dataset.*
- c. *Set.seed(20) and then subset a number of books to be used for the testing.*

- d. *Run the code for downloading the book text.*
- e. *Run the code for pre-processing and stopwords removal.*
- f. *Run the TF-IDF code.*
- g. *Create a DTM (document-term matrix).*
- h. *Define the SOM grid.*
- i. *Run the code for building the SOM model.*
- j. *Run the post-processing steps, using Kmeans to divide the SOM results into 9 clusters (k=9).*
- k. *Extract the topics assigned for the books, one main topic for each classical book.*
- l. *Save the results.*
- m. *Repeat multiple times and compare the results.*

**Expected result:** The books should be assigned to the same topic clusters each time the code is executed.

**Actual result:** As expected.

### 3. K-means Model – Topic assignment consistent

**Description:** With the same set of books used for the K-means model, those books should be assigned into the same clusters of topics.

**Precondition:** The set of books went through the pre-processing steps and a dtm was made from them.

**Assumption:** The same set of books is used.

#### Testing Steps:

- a. *Install and load the required packages.*
- b. *Load in the classical book dataset.*
- c. *Set.seed(20) and then subset a number of books to be used for the testing.*
- d. *Run the code for downloading the book text.*
- e. *Run the code for pre-processing and stopwords removal.*
- f. *Run the TF-IDF code.*
- g. *Create a DTM (document-term matrix).*
- h. *Run the code for building the Kmeans model with 9 topics (centers=9).*
- i. *Extract the topics assigned for the books, one main topic for each classical book.*
- j. *Save the results.*
- k. *Repeat multiple times and compare the results.*

**Expected result:** The books should be assigned to the same topic clusters each time the code is executed.

**Actual result:** As expected.

### 4. Recommendation system – final recommendations

**Description:** There should always be 5 final recommendations, no matter how many books within the user's favourites list.

**Precondition:** There are multiple sample users with different number of books within their favourites list.

**Assumption:** The same code is being used for the recommendation system, without any adjustments.

**Testing Steps:**

- a. *Install and load the required packages.*
- b. *Load in the modern and classical books datasets.*
- c. *Follow the pre-processing steps for the modern books dataset.*
- d. *Create multiple users by subsetting a different number of modern books to their 'favourites list'.*
- e. *Set.seed(20) and then subset a number of books to be used for the testing.*
- f. *Run the code for downloading the book text.*
- g. *Combine the datasets for the classical books and the modern books from one of the user's favourites list.*
- h. *Run the code for pre-processing and stopword removal.*
- i. *Run the TF-IDF code.*
- j. *Create a DTM (document-term matrix).*
- k. *Run the code for building the LDA model with 9 topics (k=9).*
- l. *Extract the topics assigned for the books, one main topic for each classical book.*
- m. *Extract the classical books that were assigned the same topic as one (or more) of the books from the favourites list.*
- n. *Re-combine the extracted books with the words that were contained in each of the books, using a 'word\_counts' table produced during previous steps.*
- o. *Create a 2<sup>nd</sup> DTM, this new one containing only the books that had the same topics as the books in favourites list + the books from the favourites list.*
- p. *Run the code for creating a similarity matrix.*
- q. *Loop through the results, extract the top 5 recommendations for each of the books in favourites list.*
- r. *Sort the results, taking the top5 recommendations that have the highest similarity score.*
- s. *Print out and save the results.*
- t. *Repeat for each of the multiple users created at the start, each one containing a different number of favourite books.*

**Expected result:** There should be 5 final recommendations each time.

**Actual result:** As expected.

## 5. Recommendation System – Shiny UI – Adjusting the number of books in favourites list

**Description:** When a user changes the number for the number of books within the favourites list, the output of the UI should reflect that.

**Precondition:** The Shiny app is running and the results with default values are displayed.

**Assumption:** The Shiny app is working with no errors; the default values can be changed.

**Testing Steps:**

- a. *The Shiny app is running, and the results are loaded in.*
- b. *Check the sidebar for the default value, there should be a number 2 under the favourite books section.*
- c. *Check the main section of the app, under the heading “Sample User’s Favourites” there should be 2 books displayed.*
- d. *Change the number of favourite books in the sidebar (change the ‘2’ to any other number).*
- e. *Wait for the results to load.*
- f. *There should be the same number of books under the “Sample User’s Favourites” as the number entered in the sidebar.*

**Expected result:** The results in the main section of the UI match the values inputted in the sidebar.

**Actual result:** As expected.

## 6. Recommendation System – Shiny UI - Adjusting the number of books in classical books list

**Description:** When a user changes the number for the number of books within the classical books list, the output of the UI should reflect that.

**Precondition:** The Shiny app is running and the results with default values are displayed.

**Assumption:** The Shiny app is working with no errors; the default values can be changed.

**Testing Steps:**

- a. *The Shiny app is running, and the results are loaded in.*
- b. *Check the sidebar for the default value, there should be a number 20 under the classical books section.*
- c. *Check the main section of the app, under the heading “Summary of the sample dataset”, there will be various blocks of information.*
- d. *Check the first row, second column, which is named ‘title’, the ‘Length:’ valuer under it should correspond to the default value (20).*
- e. *Change the number of classical books in the sidebar to a different number than the default.*
- f. *Wait for the results to load.*

- g. Check the same place as before in the main section of the UI, the 'Length:' value should be the same as the number inputted by the user.*

**Expected result:** The results in the main section of the UI match the values inputted in the sidebar.

**Actual result:** As expected.

## 6.0 Conclusions

For the project overall, it was a good thing to have a big enough dataset of classical books to analyse, though on the opposite side of that was the lack of data on modern books – the summaries were very short and quite lacking. The RAM limitations also impacted the project as the models and recommendations could only be done on approximately one thousand books at a time instead of using the full 13 thousand classical book dataset. The results, because of that, would have been impacted somewhat because, with more books to work with, the final recommendations could have turned out better.

All in all, by making use of the good things and finding ways to overcome the issues that arose, the project was completed, the aims were achieved, and the end results were good for the current scale of the project.

## 7.0 Further Development or Research

With more time and/or resources, there are many ways in which the recommendation system could be improved. First, it would already be an improvement if the code could be executed with the whole 13 thousand of classical books.

Besides that, having a bigger list of modern books in addition with better/longer summaries (or even some reviews for each of the books to further supplement the summaries) would contribute towards better and more accurate final recommendations.

Another improvement that could be made with more time would be to maybe create something like an actual system through which users could log in and set up their favourite books list which would then be the one used for making classical book recommendations.

## 8.0 References

References are mostly in relation to tutorials read/used throughout the project. The references are divided into subsections that relate to the topic for which it was used for.

### LDA

Robinson, J., n.d. 6 Topic modeling | Text Mining with R. [online] Tidytextmining.com. Available at: <<https://www.tidytextmining.com/topicmodeling.html#alternative-lda-implementations>> [Accessed 1 March 2021].

### K-means

Jaiswal, S., 2018. K-Means Clustering in R Tutorial. [online] DataCamp. Available at: <<https://www.datacamp.com/community/tutorials/k-means-clustering-r>> [Accessed 1 April 2021].

Uc-r.github.io. n.d. K-means Cluster Analysis · UC Business Analytics R Programming Guide. [online] Available at: <[https://uc-r.github.io/kmeans\\_clustering](https://uc-r.github.io/kmeans_clustering)> [Accessed 1 April 2021].

Kassambara, A., n.d. K-Means Clustering in R: Algorithm and Practical Examples - Datanovia. [online] Datanovia. Available at: <<https://www.datanovia.com/en/lessons/k-means-clustering-in-r-algorith-and-practical-examples/>> [Accessed 1 April 2021].

Usai, D., 2020. RPubs - Segmenting with Mixed Type Data - A Case Study Using K-Medoids on Subscription Data. [online] Rpubs.com. Available at: <<https://rpubs.com/DiegoUsai/613066>> [Accessed 1 April 2021].

R-bloggers. 2016. Clustering Mixed Data Types in R | R-bloggers. [online] Available at: <<https://www.r-bloggers.com/2016/06/clustering-mixed-data-types-in-r/>> [Accessed 1 April 2021].

### SOM (Self Organising Maps)

Rpubs.com. 2019. RPubs - Self Organizing Maps. [online] Available at: <<https://rpubs.com/AlgoritmaAcademy/som>> [Accessed 1 April 2021].

The Bowman Lab. 2020. Tutorial: Self Organizing Maps in R. [online] Available at: <<https://www.polarmicrobes.org/tutorial-self-organizing-maps-in-r/>> [Accessed 1 April 2020].

Tanner, D., 2017. Self Organizing Maps and Text Mining - Visualizing Shakespeare (Part 1). [online] Clarkdatalabs.github.io. Available at: <[https://clarkdatalabs.github.io/soms/SOM\\_Shakespeare\\_Part\\_1](https://clarkdatalabs.github.io/soms/SOM_Shakespeare_Part_1)> [Accessed 1 April 2021].

Tanner, D., 2017. Self Organizing Maps and Text Mining - Visualizing Shakespeare (Part 2). [online] Clarkdatalabs.github.io. Available at: <[https://clarkdatalabs.github.io/soms/SOM\\_Shakespeare\\_Part\\_2](https://clarkdatalabs.github.io/soms/SOM_Shakespeare_Part_2)> [Accessed 1 April 2021].

Tanner, D., 2017. Introduction to Self Organizing Maps in R - the Kohonen Package and NBA Player Statistics. [online] Clarkdatalabs.github.io. Available at: <[https://clarkdatalabs.github.io/soms/SOM\\_NBA](https://clarkdatalabs.github.io/soms/SOM_NBA)> [Accessed 1 April 2021].

### **Recommender Systems**

Teichmann, J., 2019. How to build a Recommendation Engine quick and simple. [online] Medium. Available at: <<https://towardsdatascience.com/how-to-build-a-recommendation-engine-quick-and-simple-aec8c71a823e>> [Accessed 1 March 2021].

Sharma, A., 2020. Beginner Tutorial: Recommender Systems in Python. [online] datacamp. Available at: <<https://www.datacamp.com/community/tutorials/recommender-systems-python>> [Accessed 1 April 2021].

DAS, S., 2015. Beginners Guide to learn about Content Based Recommender Engine. [online] Analytics Vidhya. Available at: <<https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/>> [Accessed 1 March 2021].

Grimaldi, E., 2018. How to build a content-based movie recommender system with Natural Language Processing. [online] Medium. Available at: <<https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243>> [Accessed 1 March 2021].

KDnuggets. n.d. Building a Content-Based Book Recommendation Engine - KDnuggets. [online] Available at: <<https://www.kdnuggets.com/2020/07/building-content-based-book-recommendation-engine.html>> [Accessed 1 April 2021].

Livebook.manning.com. n.d. Chapter 10. Content-based filtering · Practical Recommender Systems. [online] Available at: <<https://livebook.manning.com/book/practical-recommender-systems/chapter-10/1>> [Accessed 1 April 2021].

### **Others**

Gist. n.d. A simple ISBN lookup that uses Python and the Google Books API to display basic information about a book.. [online] Available at: <<https://gist.github.com/AO8/faa3f52d3d5eac63820cfa7ec2b24aa7>> [Accessed 1 March 2021].

Robinson, D., 2020. gutenbergr: Search and download public domain texts from Project Gutenberg. [online] Cran.r-project.org. Available at: <<https://cran.r-project.org/web/packages/gutenbergr/vignettes/intro.html>> [Accessed 1 February 2020].

Adityawarman, E., 2020. How to use R in Google Colab. [online] Medium. Available at: <<https://towardsdatascience.com/how-to-use-r-in-google-colab-b6e02d736497>> [Accessed 1 February 2021].

Goodreads. n.d. Most popular books. [online] Available at: <[https://www.goodreads.com/book/popular\\_by\\_date](https://www.goodreads.com/book/popular_by_date)> [Accessed 1 April 2021].

Robinson, D., 2020. Gutenbergr: Search And Download Public Domain Texts From Project Gutenberg. [online] Cran.r-project.org. Available at: <<https://cran.r-project.org/web/packages/gutenbergr/vignettes/intro.html>> [Accessed 20 December 2020].

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P., 1996. The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM,.

Nlp.stanford.edu. n.d. An Example Information Retrieval Problem. [online] Available at: <<https://nlp.stanford.edu/IR-book/html/htmledition/an-example-information-retrieval-problem-1.html>> [Accessed 22 December 2020].

## 9.0 Appendices

### 9.1. Project Plan/Proposal

#### Objectives

The main objective is to create a recommender system that would take in a person's reading preferences which would be analysed and then a classical book would be recommended based on the analysis.

The recommender system would be either stand-alone or an add on to the existing amazon recommender system.

#### Background

I have always been interested in books (especially fiction) and writing, so it seemed like a good idea to try and incorporate that into my project. What I came up with in the end was to gather some data relating to books (such as reviews, descriptions, or text of the book itself) and use that to try and predict what kind of books a specific person would like.

#### Technical Approach

The approach to the project will firstly include a lot of reading and research – into the KDD methodology, machine learning, natural language processing, text analytics, recommender systems, etc.

Then, the second step will be to source the data – such as, using the Goodreads API to source the data related to book reviews, or the Project Gutenberg for the data on classical books.

Lastly, all that will be left is to use the sourced book data to build the recommender system.

#### Technical Details

The project will be mainly using R and Python.

In terms of R, some of the main packages used will be gutenbergr, dplyr, tm, etc.

#### Evaluation

The system will first be tested by me – I will be trying to see if the classical books I get recommended are ones that I would be interested in.

I will be testing to see if the recommendations are valid. For example, if two people were to have identical book preferences, would the system recommend the same type of book? If not, would those recommended books at the very least be both valid as recommendations for both people?

At the end, I will be asking some acquaintances/family/friends to try out the system and see if the book they get as a recommendation is something they would want to read.

## Project Plan

Project Pitch			
Project proposal			Sun 08/11/20
➤ Research			Thu 12/11/20
in Text analytics			
Natural Language Processing			
Machine Learning			
KDD			
➤ Mid Point Implementation	33 days?	Sun 08/11/20	Tue 22/12/20
Research into text hashing			
Preliminary analysis of the classical book data			
Get the data from project Gutenberg			
➤ Final Implementation	132 days?	Sun 08/11/20	Sun 09/05/21
Research into making a mini search engine			
Make the mini search engine			
Download the text for all relevant classical books identified			
Look into the Amazon API to see if I can make my recommender system as an add-on to amazon recommender system			
Analyse a sample list of modern fictional books and identify the key words/topics to be used for recommendations			
based on the analysis, make a system that would recommend a classical book			
test to make sure it is automated enough and that the recommendations seem valid			
Documentation			Sun 09/05/21
Video Presentation			Sun 16/05/21

## 9.2. Reflective Journals

### October

During the month of October, the main thing I did was thinking about the ideas for the project I want to make. This included doing research, then deciding on the idea and then doing more research on the specific idea.

Around the end of October, I started writing up the project proposal.

### November

The deadline for the project proposal was the 8<sup>th</sup> of November, therefore the first week of the month was spent on finishing up the proposal.

The rest of the month was spent developing the idea further and doing research into various things that could contribute towards the overall project. Some of those things included KDD, text mining, text analysis, NLP (natural language processing), machine learning, recommender systems, Api's (more specifically available api's for websites such as goodreads and Project Gutenberg).

## December

The month of December is when I finally moved on from research and started working on some code.

I extracted the data from the Project Gutenberg website by using R package Gutenbergr, and then proceeded to pre-process and clean up the data to the point where only the data I would need for the project was left.

Using this data, I made some visualisations to be used for the report and presentation for the mid-upload which is due on the 22<sup>nd</sup>.

## February

Started working on topic modelling for classical books and sourcing the list of the modern books (with summaries for each of the books).

Started doing LDA on smaller samples of books (approx. 20 at a time) to make sure the code works as required.

## March

Moved all the code to Google Collab to overcome some of the hardware issues I had in relation to insufficient RAM.

Ran bigger samples (1000 books, then 1500 books) through the LDA code and produced visualisations for the top terms for several different topics.

Removed names from the list of words to make sure they do not show up in the final images then ran the samples again (5 different samples).

Made a custom stop word list for each of the 5 instances of the samples to further remove words that do not help in determining the final topics.

Kept increasing the list of the modern/more recent books to make sure I get some definitive results with the topic modelling.

## April

Implemented SOM and K means clustering on the same books as the LDA topic modelling for the sake of comparing the results. Started working and nearly finished the recommendations part of the system where, taken a sample user with several favourite books from a list of modern books, five classical books would be generated at the end as a recommendation. The recommendations are based on both, the topics assigned by the LDA and then similarity score.

Worked a bit on the report, test cases, visualisations.

## May

Finishing up with the code, documentation, video presentation.