A BOOK RECOMMENDER SYSTEM USING NATURAL LANGUAGE PROCESSING

Dhruvisha Gosai

Table of Contents

Executive Summary

Schools and universities use a list of textbooks for any selected topic to inform the academics, librarians, and publishers about an existing and any past reading lists that have been used. This data is often used in creating a recommendation system that informs the schools of alternative textbooks available and assist teachers in choosing new textbooks. This process is often time consuming and resource intensive when it comes to choosing a relevant book based on the subject and exploring its genres to align with the curriculums, followed by creating new teaching plans (Penguin Books Ltd., 2021).

Such an issue is addressed in this paper by leveraging the Data Science solution of Natural-Language-Processing (NLP) to create a content-based recommender system which extracts the data from open-source APIs such as Google books and Trove (Australian online library database (Trove, 2022)). The NLP model is fit through a machine learning (ML) algorithm Extreme Gradient Boosting (XGBoost) to create a recommender system that recommends similar books based on a subject of interest.

1. Introduction

Education is a fundamental human right that is essential for children to develop critical thinking necessary to participate effectively in a free society (ACT Human Rights Commission, 2022). And textbooks serve as a structure around the curriculum set out by the government. Not only that it is a mandatory requirement but there is also a positive correlation between access to textbooks and learners' results lists (The Conversation, 2015). In 2012, a province in South Africa was deemed to be in crisis after the 'Department of Basic Education' failed to deliver the textbooks to the learners (The Conversation, 2015).

With textbooks being an integral part of the education system, selecting a book that covers relevant course material for a grade is the key. In 1991, Los Angeles used to have a committee gathering comprised of at least 55% of the teachers and others professional related to the subject area once ever eight years to pick an eligible textbook (Yarber, 1991). This was later changed to selecting textbooks once every two years. Even in recent times, American school systems face yet another problem where there is a difference in the editions of a textbook – California versus Texas for instance, where American history textbooks differ in ways that are shaded by partisan politics (Goldstein, 2020).

Such delay in updating the textbooks or discrepancies in the contents would result in imparting out-of-date and/or misleading information to the students. Which is why there is a dire need to leverage the advancements in the Data Science that allows in creating an NLP recommender system that removes the presence of human bias, provides a wider choice in books, and reduces time in manually verifying a book genre (Tantillo, 2014) to ensuring if it fits the curriculum.

This project focuses on three main procedures mentioned below when creating a content-based recommender system and discusses the results and limitations, providing a bases for future work.

1. Use API to extract the information on the books for each subject.
2. Use NLP to extract relevant information from book title, description, and category.
3. Implement ML algorithm for recommender system for subjects.

2. Data

For this project, Jupyter Notebook with Python version 3.8.5 was used alongside a list of relevant Python Libraries as provided in Appendix. As a base dataset for this model, the data was provided by the university with 1,804 records that consisted of information about 'School_ID', 'State', 'Year', 'Subject', and 'ISBN'. This data by itself was not enough to create the NLP recommenders which is why external resources such as Google and Trove were used as to extract the information for each ISBN using API to enhance the data. Below sections discuss how the data was generated, wrangled, and subset before the implementation of the ML model.

2.1 Data generation

To create a reliable recommender system, it is essential to have the complete data on the books currently used by the schools. Using the Pandas library, data was imported using read_excel() function and the 1,804 records were checked for nulls and NAs – there were none. After reviewing the descriptive statistics and exploratory analysis (discussed in Section 3.2), 'ISBN' column was extracted to prepare the data to be passed for API data mining. Due multiple schools using the same textbooks, there were duplicate 'ISBN' which were removed to only have distinct records, these book IDs were stored in a list which was then used in a loop to gather the information for individual books.

Two APIs were selected – Google and Trove, which captured majority of the information for the books required to create the model except 39 'ISBN' which retrieved no information back. Google books API was chosen due to being open source and with collection of more than 40 million titles (Lee, 2019). There were 30 books however, that had no information via Google API. This is where using the Trove API was effective. Data was enriched using these two sources and a new column was created that concatenated all the derived columns after the API calls. A list of columns extracted from API calls can be referred under API columns retrieved. All the API calls made were passed with parameters (code snippet available under Parameters passed through for API calls) for Google and Trove to reduce the run-time and retrieve only the required information which was then appended after each call for ISBN and the attributes collated.

These two lists were converted to data frames and merged using the INNER JOIN with the initial supplied data (keeping only the 'Subject' and 'ISBN' columns). The final dataset consisted of 1,032 unique books where at least some sort of information was available, and the retrieved columns of data were concatenated to create a new column ready to be cleaned to create a corpus for recommender system.

2.2 Data wrangling

Once the base corpus was ready, the full description column was passed through various checks to prepare a clean corpus for NLP. Starting with removal of regular expressions (regex), the string was converted to lower case and sentences of the string split into individual words. From a list of English stop words derived from nltk package, the cleaned string had these words removed and ready for lemmatization (Methods section provides further details into why Lemmatisation was preferred over stemming). A noun-based book description was created which was assumed to capture majority of the meaning around the background of the book. All the techniques are presented under Text cleaning to prepare corpus.

To reduce the corpus size, two techniques were used – Term Frequency - Inverse Document Frequency (TF-IDF) and Non-Negative Matrix Factorization (NMF). Both the techniques were implemented using scikit-learn Python package.

2.3 Data subset

Final dataset derived from Section 2.2 was split into 80% training and 20% test data with random state of 110 and for evaluation, the 25% of the training dataset was used to validate the model and assess the performance. Accuracy, F1 scores, precision scores and recall scores were used to evaluate the performance

3. Method

Once the supplied data was imported, exploratory analysis was done to understand the spread, extract data and identify the most suitable ML and NLP algorithms that had to be performed for the most accurate recommender system.

3.1 Exploratory analysis

Supplied data was examined for the data spread to identify the skewness. As observed in Figure 3.1, it was evident that if a recommender system was created based on this available data before sampling the data based on normal distribution.

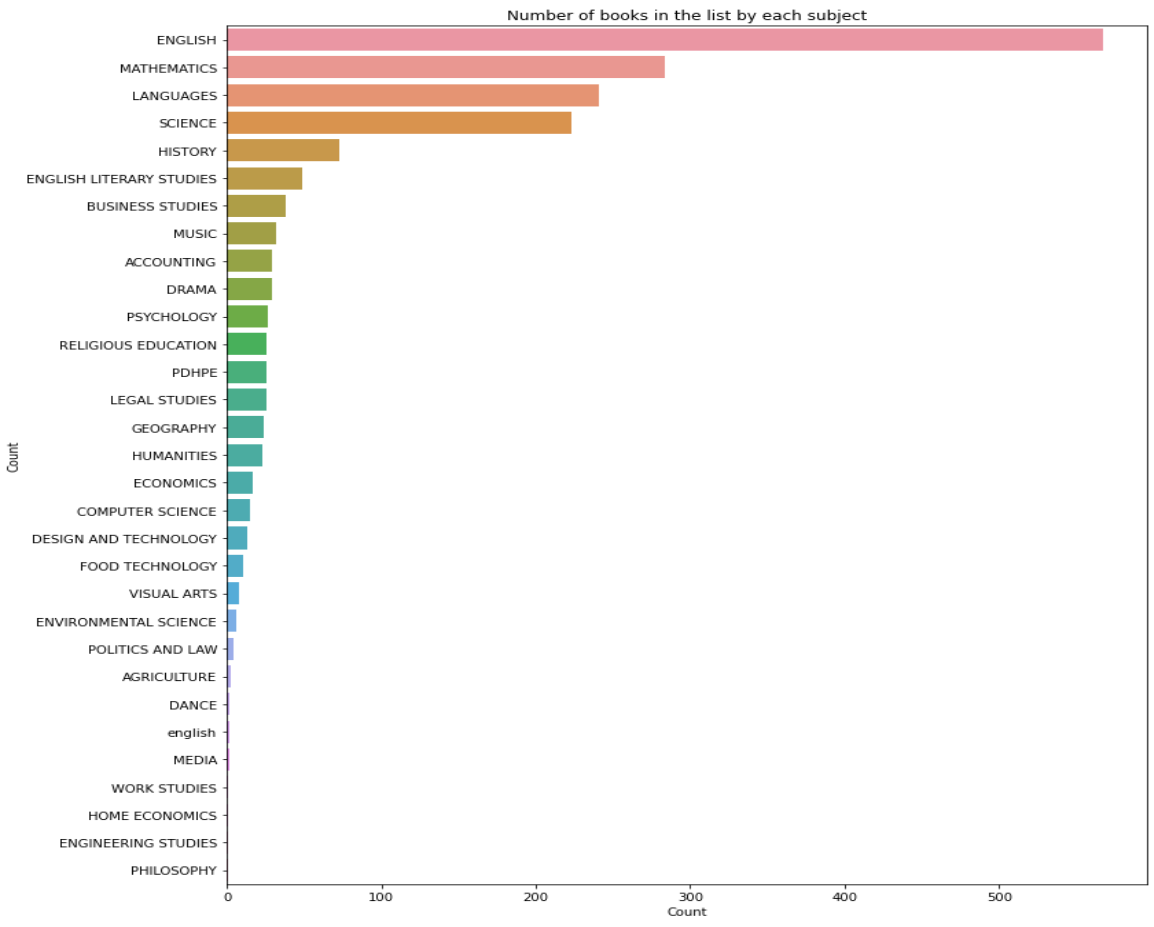


Figure 3.1 Number of books per subject to understand data spread

Once the API data was sourced, optimum techniques to prepare the corpora was explored. Initial decision was to use Stemming for the pre-processing step which aimed to convert a word to their stem (Scott, 2019). This may be useful in cases where verbs are used to drive the recommender system. In this case though, it was identified early in the process that doing so was stripping down the words to illogical word step. Hence it was decided to use Lemmatisation which reduced the word to its root synonym. Unlike Stemming, Lemmatisation makes sure that the reduced word is a dictionary word (Scott, 2019). This assisted in further narrowing down to remove all word types that weren't of Noun category. The reason why Nouns were kept is because subjects are Nouns and for this recommender system, it would work better and help reduce the feature list. The final corpus is presented with the frequency count for the top 50 words (Top 50 words derived from the corpus).

3.2 Feature Extraction

NLTK has a list of stop words which alongside SKLearn's collection of text feature extraction methods such a Count Vectorizer were used to count the number of times a word occurs in the text. This parameter was used initially to understand the maximum number of features (words) to use and the name of the stop word list. However, TF-IDF scores the words by its importance rather than counting. Hence, to reduce the number of features created from the complete corpus, TF-IDF, a statistical measure primarily used to determine the mathematical significance of a word in a document (Akdogan, 2021) was used.

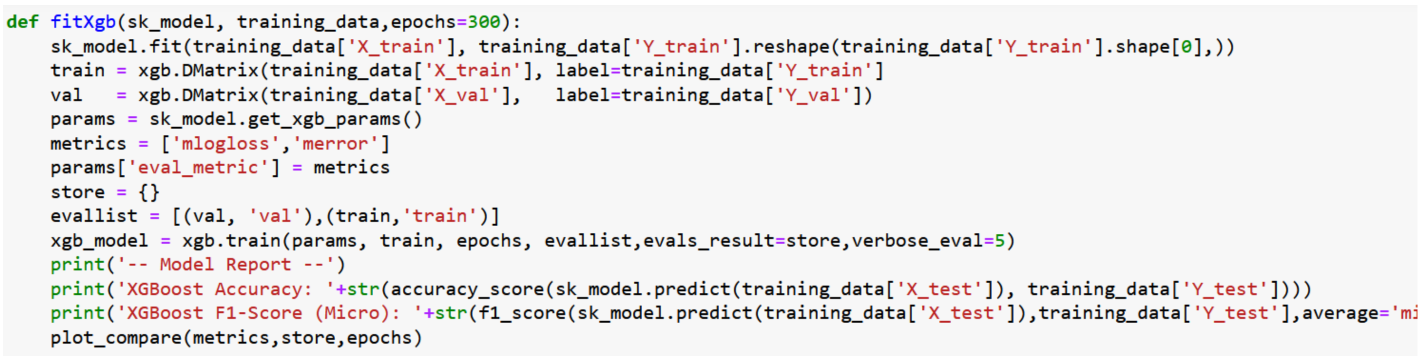
This measure was used to extract the keywords, it was used to derive the IDF weights for each word (provided under Appendix 1.5). Once the vectorizer was created, fit_transform method was used to apply it to a set of data. Feature selection wasn't restricted at this stage as the data was required to consist of all the features for Non-Negative Matrix Factorization (NMF) to form new logical groups for 80 topics. NMF was used as an unsupervised ML method to model the topics. With a cap of 0.85 for max_df and 3 for min_df to remove the terms that appeared too frequently or a smaller number of times in the corpus, this type of topic modelling was useful to reduce the features of the input corpora. This is done using factor analysis method that provides less weightage for the less coherence words (Akdogan, 2021). But the corpus required further reduction in the dimensions.

NMF was run for 80 components with a random state of 32 to derive 80 different topics, each with a created bucket which was later joined back to the complete dataset to assess the model accuracy.

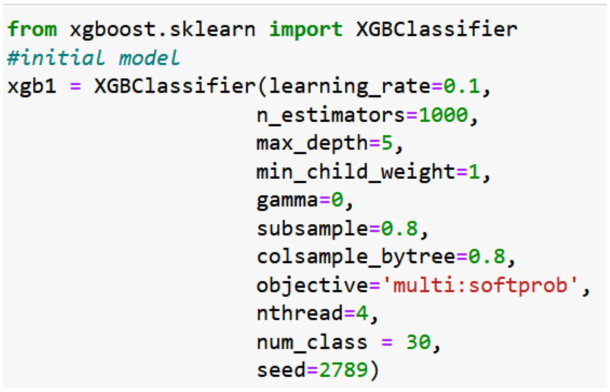
3.3 NLP Recommender System

The output from the Feature Extraction was used to generate a basic recommender model that didn't use any sophisticated system to predict the Subjects. This basic recommender was developed using cosine similarity. Any other distance metrics for large dimensions would have been near the boundary of a hypercube and could have created a lasso regularization effect in variable selection for large datasets. Hence cosine similarity was chosen that essentially uses the angle between the two data points instead of the distance. This is an efficient method compared to the normalized distances. A basic assessment of performance process was formed which looked up the ISBN and recommended books using the cosine similarity.

An additional, more sophisticated supervised ML model called Extreme Gradient Boosting (XGBoost) was introduced which used "boosting" technique for its multiple decision trees to minimize the errors (NVIDIA, n.d.). TF-IDF vectorized matrix, which was created as part of feature extraction, was used as X (dependent variable), joined to the 30 subjects as Y (independent) variable. Using the hyperparameters with metrics 'mlogloss' for multiclass log loss (a loss function used in multinomial logistic regression (Scikit Learn, n.d.)),'merror' multiclass classification error rate, XGBClassifier function from sklearn Python library was used with 300 epochs (number of complete passes of the



```
def fitXgb(sk_model, training_data,epochs=300):
    sk_model.fit(training_data['X_train'], training_data['Y_train'].reshape(training_data['Y_train'].shape[0],))
    train = xgb.DMatrix(training_data['X_train'], label=training_data['Y_train']
    val   = xgb.DMatrix(training_data['X_val'],   label=training_data['Y_val'])
    params = sk_model.get_xgb_params()
    metrics = ['mlogloss','merror']
    params['eval_metric'] = metrics
    store = {}
    evallist = [(val, 'val'),(train,'train')]
    xgb_model = xgb.train(params, train, epochs, evallist,evals_result=store,verbose_eval=5)
    print('-- Model Report --')
    print('XGBoost Accuracy: '+str(accuracy_score(sk_model.predict(training_data['X_test']), training_data['Y_test'])))
    print('XGBoost F1-Score (Micro): '+str(f1_score(sk_model.predict(training_data['X_test']),training_data['Y_test'],average='mi
    plot_compare(metrics,store,epochs)
```



```
from xgboost.sklearn import XGBClassifier
#initial model
xgb1 = XGBClassifier(learning_rate=0.1,
                    n_estimators=1000,
                    max_depth=5,
                    min_child_weight=1,
                    gamma=0,
                    subsample=0.8,
                    colsample_bytree=0.8,
                    objective='multi:softprob',
                    nthread=4,
                    num_class = 30,
                    seed=2789)
```

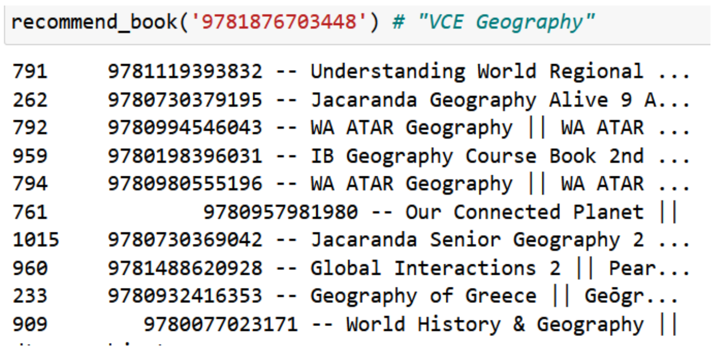
Figure 3.3: Hyperparameters for XGBoost

training dataset through the ML algorithm). Figure 3.3 displays all the specific hyperparameters set before running the model.

4. Results and Discussion

There were two recommender systems (RS) created as part of this project – a basic one using the similarity and an ML model. This section provides an overview of the results produced by both the models.

Cosine similarity RS was created to explore how well would the RS provide recommendations for basic functionality such as providing feedback based on a book title when searched for ISBN. A simple example was explored where top-10 recommendations using the ISBN search for the 'VCE Geography' book, yield a list of books that were similar based on the title of the searched book. Figure 4.0.1 provides an insight into how the recommender system output looked like. This basic recommender employed simple logic of calculating the cosine angle between the two points.



```
recommend_book('9781876703448') # "VCE Geography"
```

```
791     9781119393832 -- Understanding World Regional ...
262     9780730379195 -- Jacaranda Geography Alive 9 A...
792     9780994546043 -- WA ATAR Geography || WA ATAR ...
959     9780198396031 -- IB Geography Course Book 2nd ...
794     9780980555196 -- WA ATAR Geography || WA ATAR ...
761             9780957981980 -- Our Connected Planet ||
1015    9780730369042 -- Jacaranda Senior Geography 2 ...
960     9781488620928 -- Global Interactions 2 || Pear...
233     9780932416353 -- Geography of Greece || Geōgr...
909        9780077023171 -- World History & Geography ||
```

Figure 4.0.1: Example of recommender system

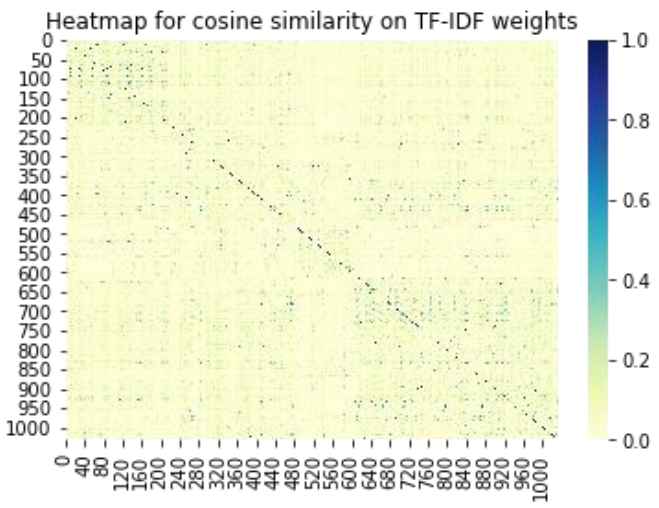


Figure 4.0.2: Heatmap with cosine similarity score

Furthermore, a heatmap with the similarity scores was created to identify if there were any particular words in the matrix that were significantly similar. However, as noted in Figure 4.0.2, there weren't any significant similarity patterns noticed.

Using the TF-IDF, NMF was created to extract the top-80 topics, each consisting of 10 largest values based off the weightings generated through the NMF model. Upon inspecting, NMF seemed to not perform as goof as presumed. This was expected because of the difference in the frequency of the topic terms. Due to the poor performance of NMF, it was not selected as an efficient RS.

```
XGBoost F1-Score (Micro): 0.6231884057971014
```

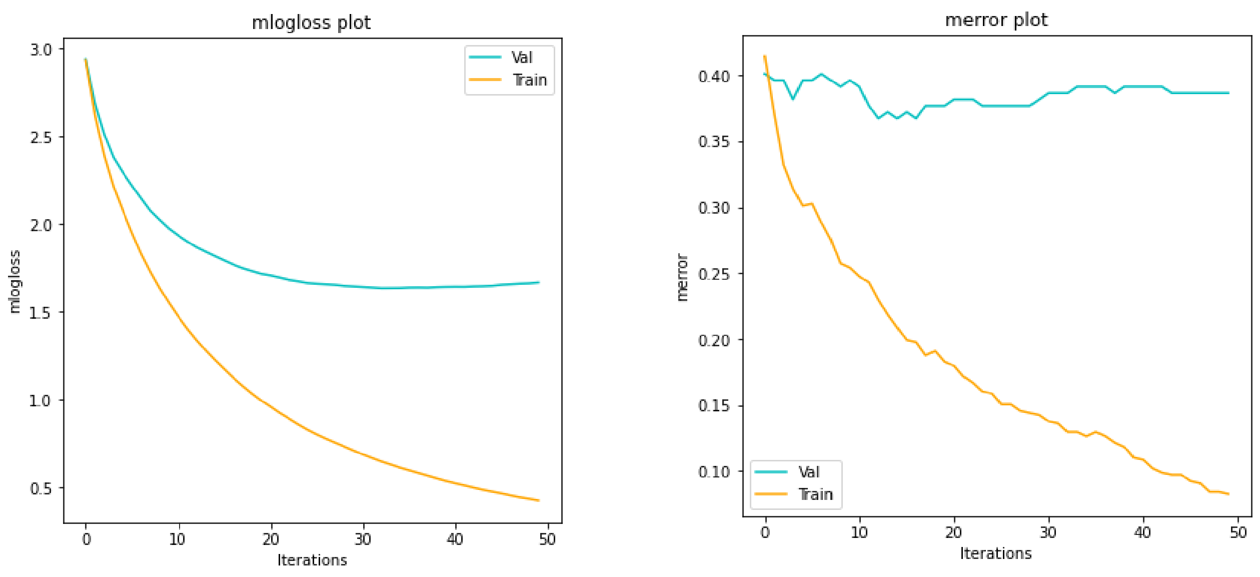
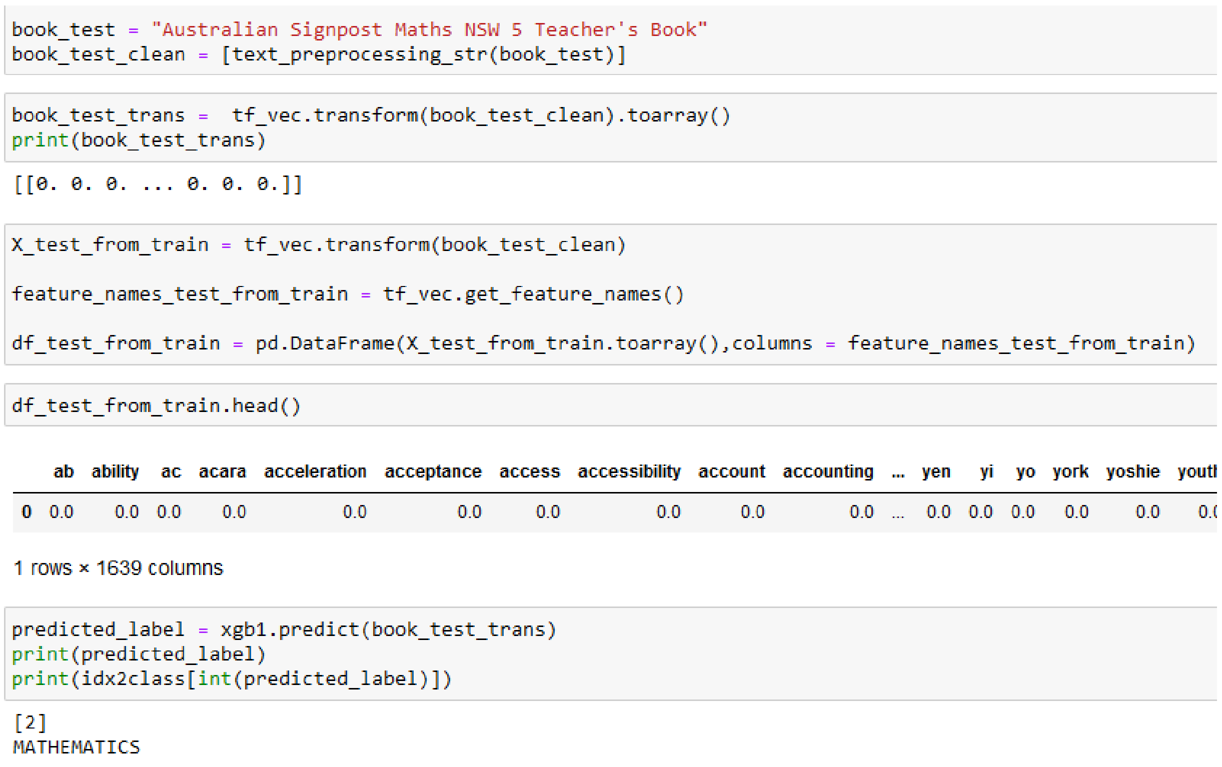


Figure 4.0.3: XGBoost results

Given the RS had to model 30 different subjects, XGBoost model with its hyperparameters set for multiclass predictions and passed through 300 iterations yielded the result as seen in Figure 4.0.3. Out of the two graphs, mlogloss plot denotes that the negative log-likelihood for error in the Training dataset was quite high at the start and reduced significantly as the interactions increased. The merror plot on the other hand denotes the classification error rate that went down with the increased iterations too. But a key difference is that in both the instances, the validation dataset was didn't seem to have a reduced error rate. Aim of the model should be to minimize both to optimize the model. Using the F1-score as an evaluation metric, the score of ~0.623 ensured that the model was performing well and tuning some parameters would increase the score for better model performance. The reason why F1-score was preferred evaluation metric as it elegantly sums up the predictive performance of a model by combining two otherwise competing metrics — precision and recall (LT, 2021). GBDT "boosting" also minimizes the bias and underfitting which why it more robust model to implement for content-based RS.

Once the model accuracies were identified, an additional test was done out of curiosity to understand how well XGBoost predicted a subject. Taking an example of a book with the title 'Australian Signpost Maths NSW 5 Teacher's Book', a human would be able to identify it is a Mathematics book. When this book title was passed through the ML model, it predicted the book as a Mathematics book correctly.



```
book_test = "Australian Signpost Maths NSW 5 Teacher's Book"
book_test_clean = [text_preprocessing_str(book_test)]
```

```
book_test_trans =  tf_vec.transform(book_test_clean).toarray()
print(book_test_trans)
```

```
[[0. 0. 0. ... 0. 0. 0.]]
```

```
X_test_from_train = tf_vec.transform(book_test_clean)

feature_names_test_from_train = tf_vec.get_feature_names()

df_test_from_train = pd.DataFrame(X_test_from_train.toarray(),columns = feature_names_test_from_train)
```

```
df_test_from_train.head()
```

	ab	ability	ac	acara	acceleration	acceptance	access	accessibility	account	accounting	...	yen	yi	yo	york	yoshie	youtl
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.(

1 rows × 1639 columns

```
predicted_label = xgb1.predict(book_test_trans)
print(predicted_label)
print(idx2class[int(predicted_label)])
```

```
[2]
MATHEMATICS
```

Figure 4.0.4: Testing on XGBoost to predict the subject of a book

Meaning, when using the cosine similarity, the process provides recommendations based on a book and XGBoost can predict the accurate subject of the book. This RS, however, could have been improved with further testing and tunning the hyperparameters.

4.2 Limitations

There were two primary subgroups of limitations encountered in this research. One was the lack of performance measure for cosine similarity RS and another dealing with the data.

4.2.1 Data limitations

There were limitations around the provided data where it was identified that there were same books shared for the same Subject. On inspecting further, it seemed to be user error where an English book was added under the Subject Mathematics. Such user error means when testing the recommender, there is a high chance the system predicts positive but when compared to the independent variable – provided Subject, it will be classed as false negative.

Another issue that was prominent was that data skewness which resulted in introducing a bias in the model. Due to the high number of books from 'English', 'Mathematics', 'Language', and 'Science', the recommendation system was able to train and predict those classes well but when testing for 'Agriculture' or 'Media', it was deemed to perform bad. There were certain subjects which had only one book provided and comparing that book against the others, the classification was done incorrectly.

Further constrains on model occurred due to using unigrams which meant subjects such as 'Environmental Science' was classed as science instead of the correct classification.

4.2.2 ML Algorithm limitation

The corpus itself was well cleaned and recommender system implemented using the cosine similarity performed better based on the ISBN testing which was done. However, there were no statistical accuracy measures implemented which raise questions on the confidence in the model. However, the XGBoost model was more robust and the accuracy measures using the performance metrics provided better results by also accounting for the bias.

There was one problem though – due to the skewness in the data, there were issues where the model performed very well in predicting the books that were part of the top 5 subjects that accounted for most of the data spread. Such issue could have been addressed by sampling the data to include a normal distribution of the Subject books.

5. Conclusion

Data science and advancements in NLP means there is constant opportunity to employ the machines to get the job done with minimal risk and higher accuracy. As part of this research, the two models were created and implemented to provide a solution for recommender system that predicts the subject of a book and provides suggestions based on a book title. However, these two models aren't implemented in conjunction to predict the subject and provide recommendations based on that subject. There is a potential for future work on this model to enhance to incorporate the two models in conjunction to yield a sophisticated solution to both identifying the subject of a book and suggested related books for that subject without requiring any manual intervention – reducing the time, risk, and resource overhead involved in the manual recommender system.

5.1 Future work and Recommendations

There is an opportunity to extend this model to create a recommender system based on the curriculum requirements which could assist the academicians by recommending the books based on defined attributes under a curriculum. For instance, Australian curriculum for 'English' consists of literary text including novels, poetry, short stories, etc. (Australian Curriculum, Assessment and Reporting Authority, 2022). If the model was extended, it would look up for those string of words to provide an accurate recommendation for subjects.

There also is a desperate need to have future work done on the upgrading the model to provide recommendations based on multiple request features such as Subject and School Year, or School Year and State. Such model would be beneficial when deployed by the national curriculum database where instead of recommending list of books for schools to choose from manually, have a recommender system that uses the curriculum as an input to generate a recommended list of books.

References

ACT Human Rights Commission. (2022). *Right to Education - ACT Human Rights Commission*. Retrieved from ACT Human Rights Commission: https://hrc.act.gov.au/humanrights/guides-and-publications/detailed-information-enshrined-rights/right-to-education-3/

Akdogan, A. (2021, July 22). *Word Embedding Techniques: Word2Vec and TF-IDF Explained*. Retrieved from Towards Data Science: https://towardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08

Australian Curriculum, Assessment and Reporting Authority. (2022). *Texts*. Retrieved from Australiancurriculum.edu.au: https://www.australiancurriculum.edu.au/senior-secondary-curriculum/english/english/texts/

Goldstein, D. (2020, Jan 12). *Two States. Eight Textbooks. Two American Stories.* Retrieved from Nytimes.com: https://www.nytimes.com/interactive/2020/01/12/us/texas-vs-california-history-textbooks.html

Lee, H. (2019, Oct 17). *15 years of Google Books*. Retrieved from Google: https://www.blog.google/products/search/15-years-google-books/

LT, Z. (2021, Nov 24). *Essential Things You Need to Know About F1-Score*. Retrieved from Towards Data Science: https://towardsdatascience.com/essential-things-you-need-to-know-about-f1-score-dbd973bf1a3

NVIDIA. (n.d.). *XGBoost*. Retrieved from NVIDIA: https://www.nvidia.com/en-us/glossary/data-science/xgboost/

Penguin Books Ltd. (2021, Mar 17). *Who decides which books you study in school?* Retrieved from Penguin.co.uk: https://www.penguin.co.uk/articles/2021/03/who-decides-books-studied-school-curriculum-english-literature-gcse-set-texts

Scikit Learn. (n.d.). *sklearn.metrics.log_loss*. Retrieved from Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html

Scott, W. (2019, Feb 15). *TF-IDF from scratch in python on a real-world dataset.* Retrieved from Towards Data Science: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Tantillo, S. (2014, July 20). *Why and How Should Teachers Make Required Reading Choices?* Retrieved from MiddleWeb: https://www.middleweb.com/16486/select-books-students-must-read/

The Conversation. (2015, Nov 09). *Why textbooks are a crucial part of every child's learning journey*. Retrieved from The Conversation: https://theconversation.com/why-textbooks-are-a-crucial-part-of-every-childs-learning-journey-50252

Trove. (2022). *Trove*. Retrieved from Trove.nla.gov.au: https://trove.nla.gov.au/

Yarber, M. L. (1991, Nov 07). *How Textbooks Are Chosen--and Who Makes the Decisions*. Retrieved from Los Angeles Times: https://www.latimes.com/archives/la-xpm-1991-11-07-we-1470-story.html

Appendix

dsad

1.1 Python Libraries

```
# Make relevant modules available
from platform import python_version
import pandas  as pd
import numpy   as np
import seaborn as sns
import matplotlib.cm as cm
import matplotlib.pyplot as plt
from matplotlib import pyplot
from time import time
# Modules for API requests
import requests
import re #regex
import csv
import json # To parse the JSON object
import nltk
from nltk.parse                     import CoreNLPParser
from nltk.corpus                    import wordnet
from nltk.corpus                    import stopwords
from nltk.stem.porter               import PorterStemmer
from nltk.stem                      import WordNetLemmatizer
from nltk.corpus.reader             import NOUN
from nltk.corpus.reader             import VERB
from nltk.tokenize                  import word_tokenize, wordpunct_tokenize, sent_tokenize
from nltk.chunk                     import ne_chunk
from nltk.tag                       import pos_tag
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set(stopwords.words('english'))
```

```
from tqdm import tqdm
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer,TfidfTransformer
from sklearn.model_selection         import train_test_split
from sklearn.cluster                 import KMeans
from sklearn                         import metrics
from sklearn.metrics                 import confusion_matrix, accuracy_score
from sklearn.naive_bayes             import GaussianNB
from sklearn.decomposition           import PCA
from sklearn.cluster                 import KMeans
from sklearn.preprocessing           import MultiLabelBinarizer
from sklearn.linear_model            import LogisticRegression
from sklearn.ensemble                import RandomForestClassifier
from sklearn.neighbors               import KNeighborsClassifier
from sklearn.pipeline                import make_pipeline,Pipeline
from sklearn.preprocessing           import StandardScaler
from sklearn.svm                     import SVC
from sklearn.multiclass              import OneVsRestClassifier # Binary Relevance
from sklearn.metrics                 import
f1_score,make_scorer,average_precision_score,recall_score # Performance metric
from sklearn.metrics.pairwise        import cosine_similarity
from sklearn.metrics.pairwise        import linear_kernel
from sklearn.preprocessing           import LabelEncoder # require for XGBoost to run
from sklearn.model_selection         import cross_val_score
le = LabelEncoder()
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score,
recall_score,precision_recall_curve, auc, roc_auc_score, roc_curve
import xgboost as xgb
from xgboost                         import XGBClassifier as XGBoostClassifier
from gensim.models                   import Word2Vec, KeyedVectors
import multiprocessing
from wordcloud import WordCloud
from pygments import highlight, lexers, formatters # To prettify the raw JSON data
from IPython.display import display
```

1.2 API columns retrieved

Google API

1. Title
2. Subtitle
3. Category
4. Description
5. Text Snippet

Trove API

1. Title
2. Category

1.3 Parameters passed through for API calls

Google API: An API key was originally passed for this but due to call limit, it was removed.

```
# Google API call
# ---------------
google_params  = {
                'q'        : isbn,
                #'key'      : google_api_key,
                'encoding':'json',
                'maxResults'       : 1
                 }
```

Trove API: This search was restricted for books only

```python
# Tove API call
# -------------
trove_params   = {
                'q'        : isbn,
                'zone'     : 'book', # Search in the book zone
                'reclevel': 'full', # This retrieves full details
                'key'      : trove_api_key,
                'encoding':'json',
                'n'        : 1
                 }
```

1.4 Text cleaning to prepare corpus

Clean corpus

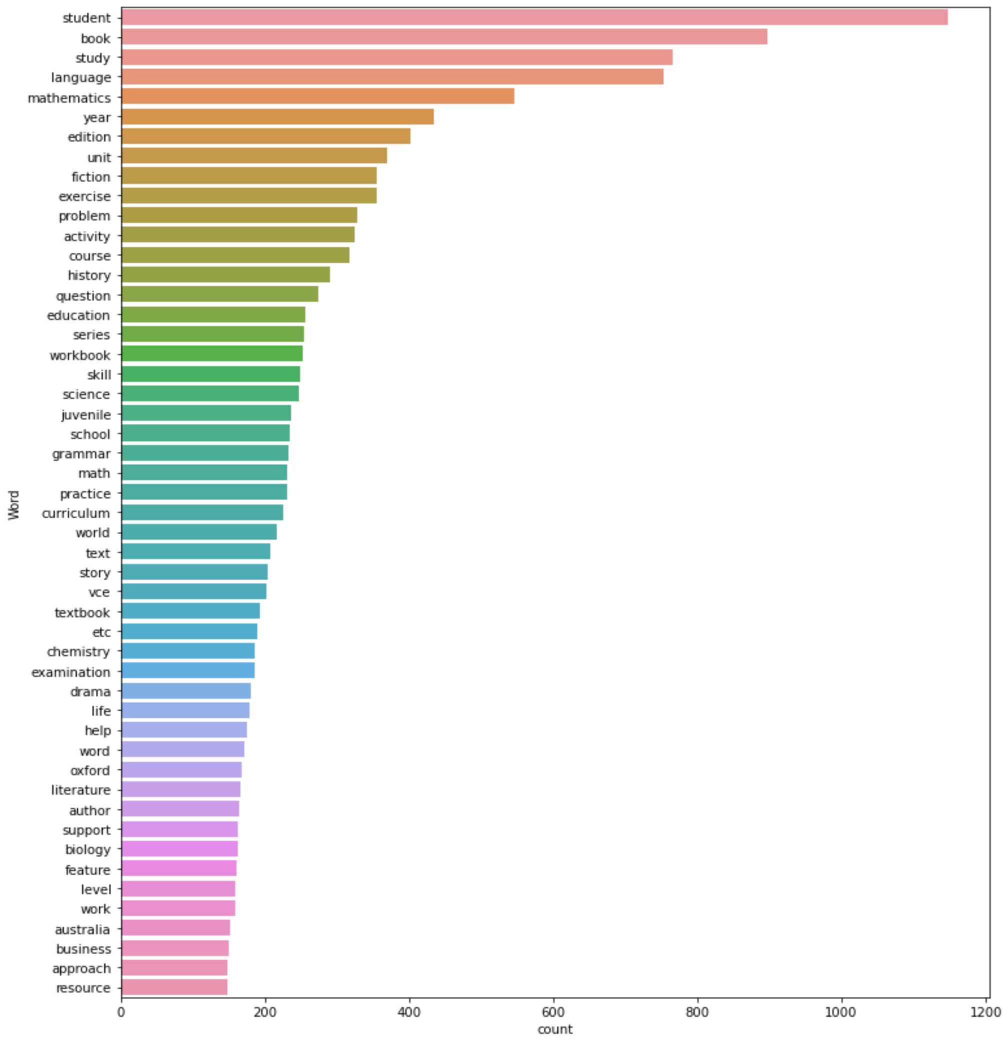
1. Lemmitisation
2. Stop words removal
3. Only keep Nouns

```python
def remove_by_regex(texts, regexp):
    output_texts = regexp.sub("",texts)
    return output_texts

def remove_urls(input_text):
    return remove_by_regex(input_text, re.compile(r"http.?://[^\s]+[\s]?"))

def text_preprocessing(uncleaned_text_corpus):
    corpus = []
    for i in range(0, len(uncleaned_text_corpus)):
        text = re.sub('[^a-zA-Z]', ' ',uncleaned_text_corpus[i])
        text = text.lower()
        text = remove_urls(text)
        text = text.split()
        # lemmatisation - no stemmer
        lm = WordNetLemmatizer()
        # remove stopwords
        all_stopwords = stopwords.words('english')
        all_stopwords.remove('not')
        # lemmatisation
        text = [lm.lemmatize(word) for word in text if not word in set(all_stopwords)]
        # only keep nouns
        text = [word for (word, pos) in nltk.pos_tag(text) if (pos == 'NN' or pos == 'NNP' or pos == 'NNS' or pos == 'NNPS')
        text = ' '.join(text)
        corpus.append(text)
    return corpus
```

1.5 Top 50 words derived from the corpus



Word	IDF Weight
student	1.895217
study	1.966413
book	2.005328
exercise	2.360493
problem	2.395045
language	2.476391
year	2.5065
edition	2.537545
unit	2.709114
course	2.834277
juvenile	2.865049
activity	2.877627
school	2.883977
etc	2.890366
question	2.890366
mathematics	2.916342
series	2.936276
education	2.984395
textbook	2.984395
examination	3.027568
skill	3.049873

world	3.065025
help	3.119941
fiction	3.119941
practice	3.169538
text	3.186632
workbook	3.186632
australia	3.19529
feature	3.230692
curriculum	3.258091
certificate	3.286262
support	3.286262
author	3.295832
life	3.305493
literature	3.305493
history	3.31525
resource	3.355255
approach	3.355255
work	3.375874
story	3.407623
note	3.418434
guide	3.429363
word	3.429363
level	3.440413
program	3.451586
science	3.451586
math	3.533503
teacher	3.558196
design	3.583514
vce	3.583514