Content-Based Recommender System for Books, and Predicting a Book's Subject using Naïve Bayes with NLP

Jarman Giffard (JC225731)

Table of Contents

Executive Summary

This report details the use and results of a naïve bayes multinomial classifier to predict a book's subject based on data collected through API, and to build a content-recommender using the NLP statistic term frequency–inverse document frequency (TF-IDF) built upon these derived fields.

1. Introduction

In Australia, text books are recommended to schools by an independent statutory authority called the Australian Curriculum, Assessment and Reporting Authority (ACARA). In conjunction with teachers, schools, and governments, textbooks are selected and then adopted by individual schools. There exists opportunity to streamline and improve upon this process via the use of machine learning by removing potential bias from selectors, generating greater consistency between schools, exposing unseen gaps in textbook content between individual schools, and partially automating the selection process to save time and money.

2. Data

The input data provided consists of 5 variables and 1,804 observations – seen below:

```
school_book_import
```

	School_ID	State	Year	Subject	ISBN
0	8	VIC	0	ENGLISH	9781741250879
1	9	NSW	0	ENGLISH	9780648237327
2	15	NSW	0	ENGLISH	9781742990682
3	15	NSW	0	ENGLISH	9781741351750
4	15	NSW	0	ENGLISH	9781742152196
...	...	...	...	...	...
1799	2	VIC	12	ACCOUNTING	9781108688772
1800	2	VIC	12	ACCOUNTING	9781108469913
1801	19	VIC	12	ACCOUNTING	9781108688772
1802	1	NSW	12	LANGUAGES	9781568364575
1803	6	VIC	12	SCIENCE	9781876703431

1804 rows × 5 columns

Figure 1: first and last 5 observations in the input dataset

School_ID represents the masked school identifier, allowing individual schools to be categorized and the State and Year columns represent Which Australian State or Territory it resides in plus the grade the textbook is assigned to. The Subject column details 30 possible subjects, a frequency distribution can be seen in appendix Appendix 2; Count of books by subject. The ISBN is the International Standard Book Number which is used as the key to lookup individual books from external providers as part of the API calls used later.

As there are some textbooks used by multiple schools there are some ISBNs duplicated which are accounted for programmatically later.

2.1 Data generation

Two external data sources are leveraged in the data enrichment process, the Trove API, and the Google API. The Trove API is managed by the National Library of Australia and provides data for a variety of mediums including newspapers, books, and magazines. The Google API used in this exercise is specific for books and similar to Trove returns fields such as Title, Authors, Publish Year, and generally a Description of the content of the material.

The API calls for this exercise were wrapped in a custom python function with individual columns extracted from the embedded json format by extracting values from the returned object – a matter of slicing lists and dictionaries on appropriate key values.

```
def trove_rlvcny_vl(obj,isbn_inner):
    try:
        relevancy_value =
obj["response"]["zone"][0]["records"]["work"][0]["relevance"]["value"]
    except KeyError:
        print("no relevancy_value for "+str(isbn_inner))
        relevancy_value = ''
    return relevancy_value
```

As seen above, the extracted variable ‘relevancy_value’ is embedded inside a total of six dictionaries and two lists and in the event is not populated will return a blank value via exception handling.

The list containing ISBNs is then passed through the main call_trove() function one by one, separated with a one second sleep duration to ensure the program remains compliant with the terms and conditions of the API documentation.

```
#run the selected ISBNs through the Trove API
def call_trove(lookup_list):
    for i in lookup_list:
        ISBN_trove_returned = trove_isbn_lookup(i)
        sleep(1)
    return ISBN_trove_returned
```

```
    print("Finished exporting Trove!")
```

The Trove data is then output to CSV with column headings:

```
#export google/trove data so we don't have to rerun the api calls each time and instead use the
previously exported data
def export_file(data,columns,filename):
    with open(filename, 'w',encoding='utf-8',newline='') as f:
        # using csv.writer method from CSV package
        write = csv.writer(f)
        write.writerow(columns)
        write.writerows(data)
```

The program checks for the existence of this CSV. If available, the program uses that CSV in the subsequent analysis, and if it does not exist (i.e. if this is the first time the program is being run) the Trove API calls are first executed and then the CSV is exported.

```
try: #if trove data has already been saved to computer then import that and use that
    print("trying to import Trove data")
    trove_data = pd.read_csv(trove_filename)
    print("import successful")
except FileNotFoundError:
    print("can't import, will produce file")
    try: #if doesn't exist in env or computer, then run api calls and export it for next time
        ISBN_trove_returned = call_trove(distinct_isbn) #
        export_file(ISBN_trove_returned,trove_fields,trove_filename) #now export the Trove API
data to a csv
        print("exported data - now importing")
        trove_data = pd.read_csv(trove_filename)
    except:
        print("beats me, you're on your own")
```

This removes the need for repeated API calls each time the program is run from the same machine and instead uses the output data produced previously.

Data from Google's API is accessed by concatenating the API URL (https://www.googleapis.com/books/v1/volumes?q=isbn:) with individual ISBNs from the list containing unique ISBNs. This list of unique ISBNs is used rather than the full input list to reduce the overall requests made and avoid redundant calls with data that's already been returned.
The response data is then encoded into utf-8 format to avoid data loss from unrecognized characters, then deserialised from json into a python object.
Individual columns are then extracted such as book Title, Subtitle (a field akin to a tagline), Category, Maturity Rating (possible options are MATURE and NOT_MATURE), and Text Snippet (containing a description of the content of the book). Each column is encompassed inside try/except statements in the event that one of the fields is missing – in which case it is returned as a blank string. The observation is appended to a list and then explicitly returned at the end of the function.

```
def google_isbn_lookup(isbn):
    with urllib.request.urlopen(base_api_link + i) as f:
        text = f.read()


    decoded_text = text.decode("utf-8")
    obj = json.loads(decoded_text) # deserializes decoded_text to a Python object

    volume_info = obj["items"][0]


    try: #TITLE
        title = volume_info["volumeInfo"]["title"]
        #title = title.strip()
    except KeyError:
        print("no title for "+str(i))
        title = ''

    try: #SUBTITLE
        subtitle = volume_info["volumeInfo"]["subtitle"]
        #subtitle = subtitle.strip()
    except KeyError:
        print("no subtitle for "+str(i))
        subtitle = ''

    try: #CATEGORY
         #converts list to str
        category = ''.join(volume_info["volumeInfo"]["categories"])
        #category = category.strip()
    except KeyError:
        print("no category for "+str(i))
        category = ''

    try:
        maturity_rating = volume_info["volumeInfo"]["maturityRating"]
        #maturity_rating = maturity_rating.strip()
    except KeyError:
        print("no maturity_rating for "+str(i))
        maturity_rating = ''

    try:
        text_snippet = volume_info["searchInfo"]["textSnippet"]
        #text_snippet = text_snippet.strip()
    except KeyError:
        print("no text_snippet for "+str(i))
        text_snippet = ''

    print("appending for " + str(i))
    isbn_list = [isbn,title,subtitle,category,maturity_rating,text_snippet]
    ISBN_google_returned.append(isbn_list)

    return ISBN_google_returned
```

The function is then invoked and called for each unique ISBN in a FOR loop with several try/except statements to account for HTTP 429 error codes which is an indication the server has rejected the request due to too many attempts calls in a short period of time. This is circumvented through the use of the sleep() function from the time module and applies 20 minute breaks in between these server rejections. This excessive sleep duration results in the program taking a long time to run the first time it is executed – but ensures that the calls fall well within the API agreement stipulations.

```
#run the selected ISBNs through the google API
def call_google(lookup_list):
    for i in lookup_list:
        try:
            ISBN_google_returned = google_isbn_lookup(i)
        except KeyError:
            print("empty record - skipping to next ISBN")
            continue
        except HTTPError:
            print("\n too many requests, sleeping 20 minutes")
            sleep(1200)
            print("\n resuming api call")
            print("\n")
            try:
                ISBN_google_returned = google_isbn_lookup(i)
            except HTTPError:
                print("\n second 429 error - sleep for 10 minutes")
                sleep(600)
                print("\n resuming api call")
            except KeyError:
                print("empty record - skipping to next ISBN")
                continue

    print("Done!")
    return ISBN_google_returned
```

Additionally - and as seen below, the data collected from these API calls is then output as a CSV file which is then imported for next time so that these API calls also only need to be run once.

```
try: #if google data has already been saved to computer then import that and use that
    print("trying to import google data")
    google_data = pd.read_csv(google_filename)
    print("import successful")
except FileNotFoundError:
    print("can't import, will produce file")
    try: #if doesn't exist in env or computer, then run api calls and export it for next time
        ISBN_google_returned = call_google(distinct_isbn) #
        export_file(ISBN_google_returned,google_fields,google_filename) #now export the Trove
API data to a csv
        print("exported data - now importing")
        google_data = pd.read_csv(google_filename)
    except:
        print("beats me, you're on your own")
```

```
In [260]: google_data.tail()
```

Out[260]:

	ISBN	g_title	g_subtitle	g_category	g_maturityRating	g_snippet
958	9780730365464	Maths Quest	Mathematical Methods, VCE Units 1 and 2	Mathematics	NOT_MATURE	NaN
959	9781107628199	Physics for the IB Diploma Coursebook with Free Online Material	NaN	Education	NOT_MATURE	Chapters covering the Options and Nature of Science, assessment guidance and answers to questions are included in the free additional online mater...
960	9780415684811	Global Politics	A New Introduction	Political Science	NOT_MATURE	The book engages directly with the issues in global politics that students are most interested in, helping them to understand the key questions an...
961	9781420229431	Queensland Senior Physical Education	NaN	Physical education and training	NOT_MATURE	The third edition of Queensland Senior Physical Education has been updated to bring to life the QSA Senior Physical Education Syllabus.Each Focus ...
962	9781471858161	Edexcel a Level French (Includes AS)	NaN	NaN	NOT_MATURE	This title develops language skills through reading, listening, speaking and writing tasks, plus translation and research practice.

Figure 2 showing a sample of data returned from the Google API

Finally, the three datasets are merged together on the primary key ISBN through the pandas merge function – using a left join from the input dataset to the Trove and then to the Google dataset.

```
#convert key we will join on to string
google_data["ISBN"] = google_data["ISBN"].astype("string")
school_book_import["ISBN"] = school_book_import["ISBN"].astype("string")
trove_data["ISBN"] = trove_data["ISBN"].astype("string")

#merge provided data with trove dataset
provided_trove = pd.merge(school_book_import, trove_data, how="left", on=["ISBN"])
#merge the above dataframe with google to create 'base' dataset
base = pd.merge(provided_trove, google_data, how="left", on=["ISBN"])
```

This master dataset is named as 'base' and a new variable created titled 'all_string'; created by concatenating all the descriptive fields sourced from the Google and Trove API.

```
#concatenate all descriptive strings together into a single column
base["all_string"] =
base["t_title"]+base["t_subject"]+base["g_title"]+base["g_subtitle"]+base["g_snippet"]
```

This 'all_string' variable becomes the source of many other variables created later on used as model inputs.

2.2 Data wrangling

In order to improve the accuracy of the model on the content of the descriptive fields sourced from the Trove and Google APIs it is necessary to remove glyphs that offer little predictive power. This junction in text pre-processing is beneficial as almost all bodies of text contain punctuation which can diminish the predictive capabilities of models built on them (Etaiwi & Naymat, 2017).

A function is created to replace all digits, special characters, and punctuation marks with a single space. This was necessary as exploratory analysis of some of the descriptive fields indicated that there were some words separated with punctuation "like.this!sentence", so simply the removing the punctuation marks altogether would have resulted in a string "likethissentence" – reducing the volume of text that can be meaningfully parsed. Consecutive spaces were then replaced with a single space, offsetting the numerous spacing incurred from the above exchange. Finally, a list of stop words sourced from the NLTK (Natural Language Tool Kit) python module is then loaded and matches from any of the words in descriptive strings sourced from the APIs removed. This step in "…data pre-processing has become an NLP standard in both research and industry" (Sarica, 2021) and serves to counteract the effect of frequently-used words which offer diminished predictive power in models; a phenomenon credited to as Zipf's law (Rani & Lobiyal, 2020).

Next, three distinct additional columns are created: 'snow_stemmed', 'lemmatised', and 'noun'. The first two columns reflect the NLP text processing operations Stemming and Lemmatising – the former representing the procedure to principally remove suffixes and prefixes from different forms

of the same compound word (i.e. “walks” and “walking” ideally becomes “walk”), while the latter takes into consideration the morphological source of the word using context and arrives at a similar but generally cleaner result. The chief difference between these two actions is that stemming is a “crude heuristic process that chops off the ends of words”, while lemmatizing is a somewhat more elegant approach that aims “…to remove inflectional endings only and to return the base or dictionary form of a word” (Stemming and lemmatization, 2008).

The variables are created using the appropriately named lemmatize() and stem() functions sourced from the NLTK module via list comprehension, applied against the ‘all_string’ column, and then concatenated back into their own distinct variables:

```
def lemm(x):
    #Lemmatization
    x = ' '.join([lemmer.lemmatize(w) for w in x.rstrip().split()])
    return x

def stemm(x):
    #stemming
    x = ' '.join([snowball.stem(w) for w in x.rstrip().split()])
    return x
```

Word stemming was applied to the concatenated string ‘all_string’ by using the Snowball algorithm. The overall impact of this was minimal, with most strings not seeing much of a significant difference. An example is shown below where the word “students” was truncated to “stud”.

```
In [245]: #======= ======= ======= ======= ======= ======= ======= =======
          pd.options.display.max_colwidth = 150
          testcase2 = base[base['ISBN']=='9780648237327']
          print(testcase2['all_string'])
          print(testcase2['snow_stemmed'])

          1    kluwell home reading yellow levelsuitable lower primary students
          Name: all_string, dtype: object
          1    kluwell home reading yellow levelsuitable lower primary stud
          Name: snow_stemmed, dtype: object
```

Figure 3 showing the effect of stemming on a sample string

The third column ‘noun’ uses the NLTK functions word_tokenize() and pos_tag() to first split the ‘all_string’ column into separated words, then parse the string to identify and preserve only the nouns. This form of grammatical tagging is another tool in NLP feature extraction to highlight words that have the capacity to be more significant indicators of the content of an observation, while dropping less relevant features (Afrin, 2001)

```
def nouns(x):
    tk = nltk.word_tokenize(x)
    x = [word for (word, pos) in nltk.pos_tag(tk) if (pos == 'NN' or pos == 'NNP' or pos ==
'NNS' or pos == 'NNPS')]
    x = ' '.join(x)
    return x
```

An additional four compound variables are then created

- stem_lemm – string is first stemmed and then lemmatized
- lemm_stemm – string is first lemmatized and then stemmed
- lemm_noun – string is first lemmatized and then only nouns preserved
- noun_lemm – first nouns are preserved and then lemmatized

```
base['lemmatised'] = base['all_string'].apply(lemm)     # lemmatise every word.
base['snow_stemmed'] = base['all_string'].apply(stemm) # Stem every word.
base['noun'] = base['all_string'].apply(nouns)     # keep nouns only

base['stem_lemm'] = base['snow_stemmed'].apply(lemm)    #after lemm, apply stem
base['lemm_stem'] = base['lemmatised'].apply(stemm)     #after stem, apply lemm

#now extract nouns
base['lemm_noun'] = base['lemmatised'].apply(nouns) #after lemm, keep noun
base['noun_lemm'] = base['noun'].apply(lemm) #after nouns, apply lem
```

These variables form part of the subsequent tests to identify whether there are differences in model performance when using different text preprocessing procedures exclusively and/or whether the order of particular procedures demonstrate material impact.

2.3 Data subset

From the cleaned 'base' dataset, a train and test subset is produced – consisting of 80% and 20% of total observations respectively. This is achieved through use of the sample() function sourced from the Random module in Python by first subsetting 80% of the rows into the training dataset, and then assigning observations that are not in the training dataset into the test dataset (i.e. are mutually exclusive).

```
train = clean_base1.sample(frac=0.8,random_state=seed) #seed value = 800
test  = clean_base1.drop(train.index)
```

Additional subsetting was then performed to create entirely distinct training and test datasets, subset by the Subject value. Although there are 30 subjects in the raw input dataset, 73% of observations fall into either English, Math, Languages, or Science – while the bottom 15 subjects account for only 5.99% of observations (a grouped tally can be found in Appendix 2; Count of books by subject). This strong class imbalance is expected to impact the classification results so two splits are made – one to exclude the bottom 15 subjects, while the other keeps the top 3.

3. NLP Recommender System and Naïve Bayes Classifier

There are two primary machine learning algorithms used in this exercise; term frequency-inverse document frequency (TF-IDF) used in the recommender component, and a multinomial Naïve Bayes classifier for predicting a book's subject category. This naïve bayes algorithm leverages feature vectors and is the preferred approach for text categorization (compared to alternate naïve bayes algorithms such as Gaussian and Bernoulli) (Bannerjee, 2020), while TF-IDF is useful for "…examining the relevance of key-words to documents in corpus" (Qaiser & Ali, 2018) and is contrasted by alternate algorithms such as Bag of Words (BoW) and word2vec.

This particular flavour of naïve bayes holds the same presuppositions of its primary counterpart in that it assumes that each feature used is independent of each other. That is to say, this model assumes that each word used in the description or title of a book is entirely independent of all *other* words. As expected, this assumption is one that is rarely proven true (hence the 'naïve'); the presence of certain features usually **are** valid indicators for other features in a dataset. This holds true for NLP exercises where the presence of particular words alongside other words is not a random characteristic but instead a likely representative tool – one that holds practical predictive power.

The primary objective of content recommender systems is to develop a profile for each item which represents significant attributes of that item (ML - Content Based Recommender System, 2020). These defining characteristics are then utilised by a model to find alternate items that are similar. There are different types of recommender systems available based on the type of data science problem posed, but the biggest deciding factor for the type of system to use is generally dependent on the type of data available. Collaborative, Utility, and Demographic recommender systems require information about the user such as their interests, preferences, and past actions such as purchases or browsing habits (Classifying Different Types of Recommender Systems, 2015). As a result, the recommender system required in this instance is a Content-based recommender system as user details are unknown while the details of the products *are* known. Established content recommender systems can be found on streaming services such as Netflix and online shopping services such as Amazon – both of which employ the use of natural language processing content recommenders to suggest items to their customers (Netflix Recommender Systems: How Netflix Uses NLP For Personalization, 2022).

Cosine similarity scores are calculated for each book against each other book and stored as a matrix using the linear_kernel() function from the sklearn module. This produced score measures the angle between two vectors in a multidimensional plane. The comparative vectors are a result of the TF-IDF calculation for each of the words contained in the 'all_string' column. Defined hyperparameters for the TF-IDF transformation include specifying that unigrams, bigrams, and trigrams be captured by stating *ngram_range=(1, 3)*. This specification results in input string vectors to be split into list of lists, consisting of single words, and the same words paired with their first and secondary partners (e.g. "the big cat blinked" becomes [["the","big","cat","blinked'],["the big", "big cat", "cat blinked"]…]). Opting to use bigrams and trigrams in addition to unigrams has historically yielded positive results in NLP models (Muhammad Fuad & Vijayakumar, 2019)

4. Results

Multiple iterations of the recommender system were run to produce recommendations using different books, as well as multiple iterations of the Naïve Bayes classifier to predict the Subject of the book differently processed descriptive variables. An overview of the Naïve Bayes iterations and the accuracy, recall, precision, and F1 scores for each is shown below in table 4.

				Recall		Precision		F1	
Test	**Column**	**Desc**	**Accuracy**	**Micro**	**Macro**	**Micro**	**Macro**	**Micro**	**Macro**
1	all_string	no further processing	58.41%	0.584	0.125	0.584	0.1	0.584	0.107
2	snow_stem	stemmed words using Snowball	28.50%	0.285	0.041	0.285	0.011	0.285	0.018
3	lemmatised	lemmatisation applied	58.41%	0.584	0.125	0.584	0.1	0.584	0.107
4	stem_lemm	words stemmed and then lemmatised	28.50%	0.285	0.041	0.285	0.011	0.285	0.018
5	lemm_stem	words lemmatised and then stemmed	28.50%	0.285	0.041	0.285	0.011	0.285	0.018
6	noun	only nouns (single, plural, proper)	69.63%	0.696	0.166	0.696	0.145	0.696	0.152
7	noun_lemm	only nouns then lemmatised	58.41%	0.584	0.125	0.584	0.1	0.584	0.107
8	lemm_noun	lemmatised and then only nouns	58.41%	0.584	0.125	0.584	0.1	0.584	0.107
9	all_string	subset of books: removed the bottom 15	65.82%	0.658	0.214	0.658	0.18	0.658	0.191
10	all_string	using a subset of books: only the top 3 (ENG,MATH,SCI)	100%	1	1	1	1	1	1

Fig 4: a breakdown of tests conducted and ensuing results

There are several notable observations from this table – but firstly a note on the performance metrics. For the recall, precision, and F1 metrics - both the Micro and Macro averaged values have been provided. Despite falling under the same banner, the calculations behind these differ and interpretations of each these are distinct; the macro-averaged calculation computes the average

independently for *each* class (i.e. Subject), while the micro computes the aggregate across all classes before averaging. At a glance it is apparent that for all tests the micro-averaged results are significantly more than their macro counterpart – an indication that there is a class imbalance in the predictions, and that the model is much more accurate (biased) at predicting Subjects with more observations.

For test 1 the input response variable is the variable 'all_string'; this column is the concatenated string containing all descriptive fields sourced from the Trove and Google APIs and then stripped of punctuation and stop words – no stemming or lemmatization has been applied. Overall accuracy (micro-averaged recall/precision/f1) is relatively poor at 58.41% but when considering there are 30 possible subjects to allocate test observations to is still promising.

Tests 2, 4, and 5 – using the stemmed words, stemmed then lemmatized string, and lemmatized then stemmed string respectively – yielded the least promising results with micro-averaged recall, precision, and F1 scores sitting at 0.285. The macro-averaged F score computed in all three tests is 0.018 – an extremely poor result. As these were the only tests using the stemmed words it would appear that this text processing technique not only yielded zero benefit but in fact negatively impacted the feature selection; resulting in the micro-averaged recall, precision, and F1 score being halved – and macro-averaged metrics reduced to $1/10^{th}$ of the raw 'all_string' variable.

Tests 3, 7, and 8 utilised the lemmatized variable, the noun-extracted then lemmatized variable, and the lemmatized and then noun-extracted variable respectively. The results for these mirrored those for Test 1's 'all_string' variable across the board – showing no improvement or reduction in recall, precision, or F1 scores for either the micro or macro averages. This is a somewhat unexpected result due to the nature of the changes on the response variables in play, as some books saw dramatic reductions in the length of text and variety of words. The lack of impact on the model results is likely due to the equality of variable transformations. That is to say, if all the words are altered in the same way then there is ultimately *no* change to the quality of features that can be extracted.

Test 6 used the variable 'noun' – a string of only the nouns extracted from the concatenated variable 'all_string'. This test bore the single most tenable results with an accuracy of 69.63% and a macro F1 score of 0.152. These results still show plenty of opportunity for improvement but relative to the other tests are a clear winner. These results are consistent with the previous tests, indicating

the features in this column were successfully used to allocate books to Subjects that had many books – but with diminished success at allocating books to Subjects that were less frequent in the dataset.

The results of this test also produced a unique disparity with the prior tests in that the confusion matrix revealed that this test was the only one to assign books to the Subject “Languages” – and did so with 100% success (24/24).

	Predicted				
	ENGLISH	LANGUAGES	MATHEMATICS	SCIENCE	All
TRUE					
ACCOUNTING	2	0	0	0	2
AGRICULTURE	1	0	0	0	1
BUSINESS STUDIES	6	0	0	0	6
COMPUTER SCIENCE	2	0	0	0	2
DESIGN AND TECHNOLOGY	2	0	0	0	2
DRAMA	4	0	0	0	4
ECONOMICS	3	0	0	0	3
ENGLISH	61	0	0	0	61
ENGLISH LITERARY STUDIES	5	0	0	0	5
FOOD TECHNOLOGY	4	0	0	0	4
GEOGRAPHY	3	0	0	0	3
HISTORY	8	0	0	0	8
HUMANITIES	3	0	0	0	3
LANGUAGES	0	24	0	0	24
LEGAL STUDIES	2	0	0	0	2
MATHEMATICS	0	0	36	0	36
MUSIC	6	0	0	0	6
PDHPE	4	0	0	0	4
POLITICS AND LAW	2	0	0	0	2
PSYCHOLOGY	3	0	0	0	3
RELIGIOUS EDUCATION	3	0	0	0	3
SCIENCE	0	0	0	28	28
VISUAL ARTS	1	0	0	0	1
WORK STUDIES	1	0	0	0	1
All	126	24	36	28	214

Fig 5: full confusion matrix for test 6 – ‘noun’.

Note the additional column for LANGUAGES compared with the confusion matrix produced for test 1 in Appendix 5; Confusion matrix for test 1: all_string

Unlike prior tests which resulted in the model assigning books to the top three most frequent Subjects in the list English, Math, and Science (Appendix 5; Confusion matrix for test 1: all_string), this test was the only one to include the 4 category “Languages”. Based on the total count of books assigned by Subject (bottom row) it would seem that the variables used in prior tests did not

provide sufficient distinction between the subjects “English” and “Languages” resulting in the model assigning test cases to just “English”.

Tests 9 and 10 utilised a deeper slice of the dataset on Subject, by keeping the top 14 and 3 subjects respectively with the response variable ‘all_string’. Test 9 resulted in 65.8% accuracy – less than test 6 (nouns), but surpassed test 6 in a 25% increase in the macro-averaged F1 score landing at 0.191. When accounting for the change in test conditions this highlights that this improvement is purely due to fewer subjects to select from, and not due to an objective enhancement to the predictive power of the model from additional functions or alternate hyperparameters. This is further solidified by test 10 which results in perfect scores across the board due to there only being three subjects to choose from.

Content-based Recommender

Evaluating content-based recommender models that do not rely on user-ratings or combine with alternate recommenders to form a hybrid recommender is a difficult task. Unlike other statistical models which function by predicting the answer to a question and then measured by metrics that examine whether the prediction was correct, there is no earnest way to say whether the recommendations made by a content-recommender are ‘correct’.

An example of this can be seen below, where ISBN 9780310709626 (“The Beginner’s Bible”) is provided to the recommender function and the titles of the top 10 matching books returned:

```
In [276]: recommend_book('9780310709626')

528                                        9780564098354 -- youth bible
298                                  9781599829234 -- catholic youth bible
104                                       9780849907685 -- holy bible
556                           9780970181671 -- heart wisdom teaching approach
288    9781599822211 -- breakthrough bible old testament activity booklet
63                                     9781400316908 -- first study bible
130                                       9780141316253 -- boy overboard
151                                  9780061441714 -- nrsv catholic edition
47                                          9780142410394 -- twits
257                            9781785583575 -- blitz beginner theory book
dtype: string
```

Fig 6 Example of Catholic Youth Bible feature selection

As seen, almost all of the titles include the word “bible” or orientate in some way towards religious themes; such as “catholic”, “bible”, “church”, and “scriptures”.

So, it would strongly indicate that the recommender is functioning in a recognizably correct manner. It remains to be confirmed though whether these are *the* best possible suggestions from the input list however, it might be that there are books in the dataset that are superior submissions. Alternative recommender approaches, text preprocessing methods, or hyperparameters may result in different books scoring higher – but whether these books are “better” remains an unanswerable question as there remains no objective means to verify without supplementary data such as click-through-rates (CTR).

That being said, from a consumer perspective the returned results are strong matches to the subject material and would appear to be wholly relevant when compared to the content of the nominated book – a direct result of the implicit ranked reversal ordering applied when the recommend_book function is invoked. Repeated demonstrations of the recommender can be found in *Appendix 1; examples of recommendations from different books* further strengthening the legitimacy of the recommender system from at least a subjective viewpoint.

5. Recommendations and Data limitations

Recommendations

The accuracy of the provided data was revealed to be questionable during exploratory analysis and subsequent ML model application. The “Subject” column in the provided dataset was found to be often misaligned with the “Subject”/”Category” columns returned from Trove and Google respectively and inspection of rows with inconsistencies indicated that the Trove and Google values were generally more correct:

```
In [259]: #======= ======= ======= ======= ======= ======= ======= =======
          pd.options.display.max_colwidth = 150
          testcase3 = base[base['ISBN']=='9780310709626']
          print("Provided Subject = " + testcase3['Subject'] + " -- Google Subject= "+ testcase3['g_title'] + "Trove Subj= " + testcas
          #print(testcase3['g_title'])
          #print(testcase3['t_title'])

          9781742152196

          5    Provided Subject = ENGLISH -- Google Subject= beginner's bibleTrove Subj= beginner's bible
          dtype: object

Out[259]: 9781742152196
```

Figure 7: “Beginner’s Bible” labelled as an ENGLISH book, rather than Religious Studies

In the above example it can be seen that although the provided Subject was “ENGLISH” the Trove and Google fields stated this ISBN to be a “Beginner’s Bible”. Subsequent searching online reveals this ISBN to be “The Beginner's Bible” verifying that both the external data providers are accurate – and the provided data is unreliable.

Considerable effort was invested in attempting to clean these fields to produce and overarching “clean_subject” field derived from the three data sources, however due to inconsistencies and the sheer volume of possible returned categories for these fields it was proven to be challenging to reconcile these values into a clean dataset. A direct consequence of this is to call into question the results of the Naïve Bayes classifier whose target variable is predicting the datasets “Subject” column based on the concatenated descriptive fields for each book. It is likely that if effort were to be invested in manually hardcoding the correct Subject for each book it would lead to superior accuracy in the model. This approach is often implemented in ensuring a stable ‘baseline’ of accurate training datasets. However this approach is not without risks – a recent study found that roughly 30% of Google’s sentiment analysis training dataset is incorrectly labelled (Chen, 2022), showing that complex NLP tasks are difficult to solve with machine learning and humans.

Limitations

During the initial data analysis of the input dataset it was discovered that there were many duplicated ISBNS – indicating that many schools use the same books. This is unsurprising but has offers opportunity for improvement if a model of this nature were to be ever released in an official capacity; schools could benefit from visibility of what textbooks other schools are using for ideas. That is to say, if there’s one school in Victoria using a grade 12 math textbook that is not used by any other school in Australia – it could prompt management at this school to investigate why this is and whether there are ‘better’ books on the market available. This investigation would be especially appropriate if there are 50 other grade 12 math classes in Australia using a specific book which may indicate that this is the preferred math book for this cohort. Thus, simple summary statistics made available to schools for visibility are likely to lead to quick-wins on this front, while recommender models may offer diminishing returns from a cost-benefit perspective.

7. Conclusion and Future work

Based on the results of the Naïve Bayes multinomial classifier there are several key observations:

- Stemming negatively impacted the useful features in the response variables
- Lemmatization had a positive impact on the number of useful features
- Extracting and keeping only nouns had the most apparent improvement in quality being the only test run to include the subject "Language" as an option – and then correctly assigning all the "Language" books to it.
    - Despite showing strength in the classifier, use of the 'noun' variable in the Recommender Model resulted in some unusual and inappropriate recommendations – especially when compared to the results of the raw 'all_string' variable (Appendix 5; Confusion matrix for test 1: all_string)

The limitations in performance noted for the naïve bayes classifier likely stem from two key data issues in the input dataset; the first being that many books are assigned a "Subject" that is incorrect (e.g. having the Subject ENGLISH when it's a Math textbook), and a strong imbalance in the frequency of observations favouring some subjects while others have very few.

Use of Naïve bayes classifiers in NLP is certainly not unheard of but does appear to be less common than alternate ML models based on google search results.

Google term search	Results (approx)
"nlp" "cnn"	2,780,000
"nlp" " nn "	2,290,000
"nlp" "logistic regression"	695,000
"nlp" "lda"	573,000
"nlp" "knn"	460,000
"nlp" "naive bayes"	393,000

Search results were based on "nlp" and stated ML model in quotes; google.com/search?q="nlp"+"logistic+regression"

Although the results of this application of naïve bayes in solving NLP problems do not constitute an unmitigated success, it is apparent that there are strong markers indicating that greater success is attainable through alternate target variables and higher quality datasets. Future investigations on this dataset would likely benefit from sampling procedures such as ROSE or SMOTE to reduce the class imbalance apparent between Subjects.

Appendix

Appendix 1; examples of recommendations from different books:

```
In [285]: recommend_book('9780170373999') # "A+ Chemistry Notes VCE"

680
9780170373982 -- chemistry notes vce unit || chem notes unit jennifer willis
684
9780170373968 -- biology notes vce unit || biol notes vce biology unit diane canavan
683
9780170373951 -- biology notes vce unit || vce biology unit biol notes diane canavan
759
9781740982504 --  || chemistry year atar course units lyndon smith consulting editor
403    9780730373643 -- jacaranda chemistry || jacaranda chemistry vce units neale taylor angela stubbs robert stokes jaso
n wallace mirela matthews lakshmi sharma sue liston nicole dobson jennifer moloney maida derbogosian santina raphael sholt
o bowen w...
758
9781740982894 -- chemistry year || chemistry year atar course units lyndon smith consulting editor
703
9781741256765 -- excel || year chemistry geoffrey thickett
690
9780170354127 -- mathematical methods notes vce units || mathematical methods vce units notes sue garner
681
9780170374040 -- psych notes vce unit || psych notes vce unit peter milesi
682
9780170374057 -- psych notes vce unit || vce psychology unit psych notes peter milesi
dtype: string
```

[A+ Chemistry Notes VCE]

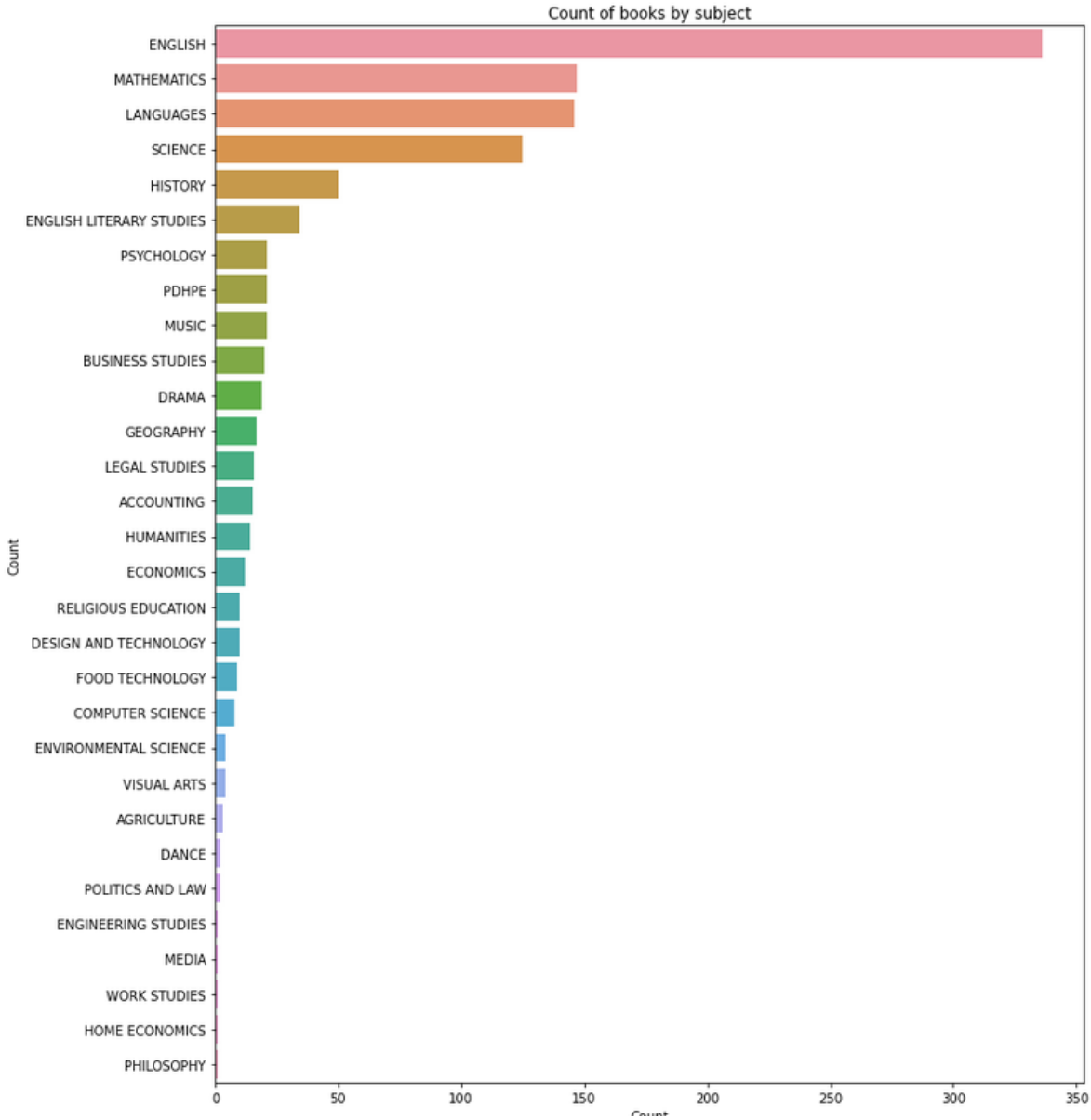
```
In [288]: recommend_book('9781741253160') # "Excel HSC Food Technology"

1025    9780170378611 -- food solutions || food solutions food studies units glenis heath heat...
945     9780170400572 -- food tech focus || food tech focus michelle downie rosalie gualtieri ...
746                    9780947225667 -- furnishing industry study || furnishing industry study
892     9780957742888 -- new advanced mathematics || new advanced mathematics complete hsc mat...
950                        9780170413220 --  || maths focus mathematics advanced margaret grove
707     9781741257267 -- physics || physics past hsc questions module year past hsc papers edi...
675     9780170459273 -- hsc year mathematics extension practice exams || hsc year mathematics...
780     9780855837716 -- nsw biology || nsw biology module heredity module genetic change kerr...
692     9781922034717 -- extension mathematics || extension mathematics past hsc papers worked...
703                                  9781741256765 -- excel || year chemistry geoffrey thickett
dtype: string
```

[Excel HSC Food Technology]

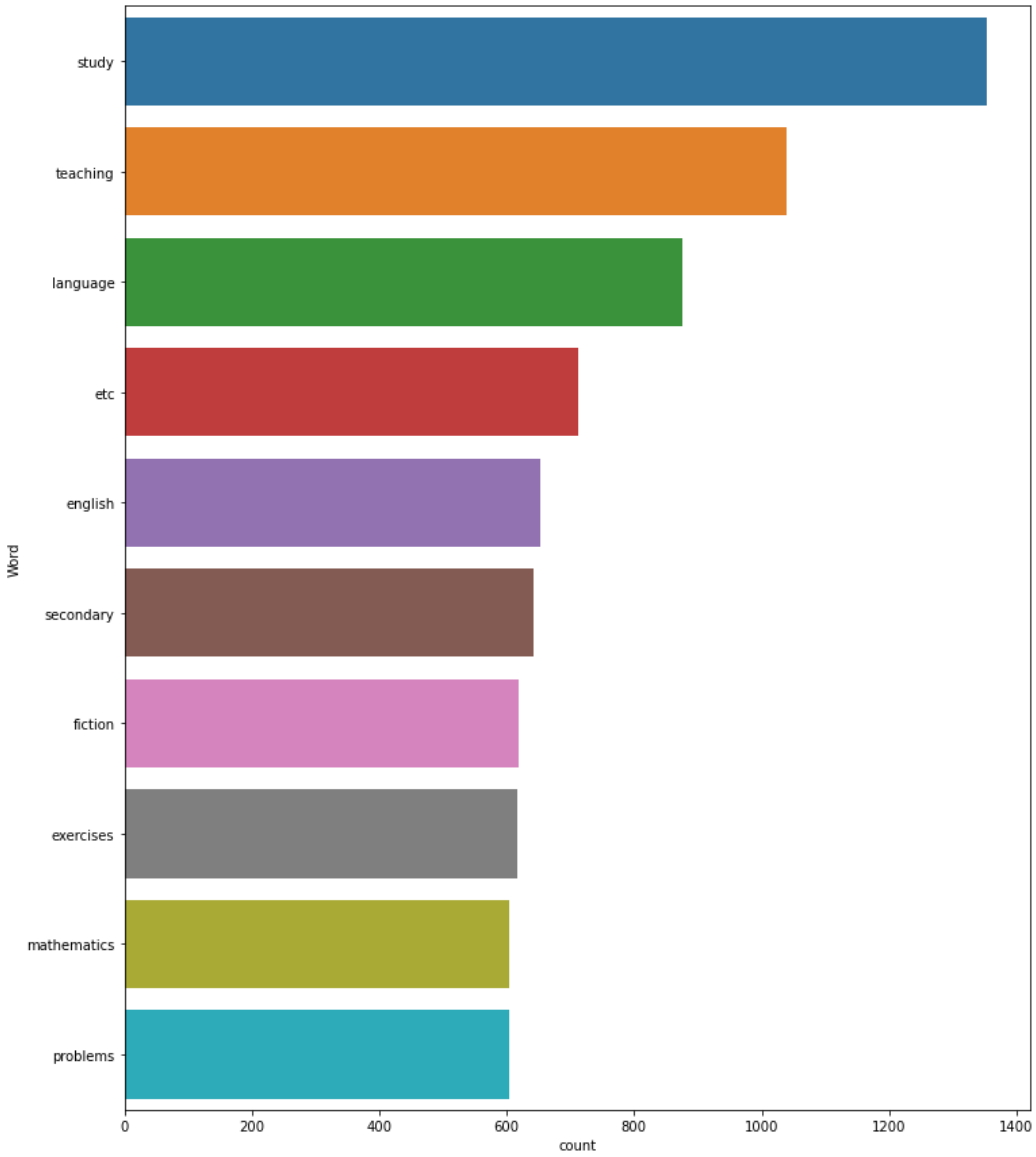
```
In [289]: recommend_book('9780980874921') # "Software Design & Development"

341    9780170411820 -- digital technologies australian curriculum workbook || digital techno...
243    9780170411813 -- digital technologies australian curriculum workbook || digital techno...
780    9780855837716 -- nsw biology || nsw biology module heredity module genetic change kerr...
931    9780170365505 -- society culture preliminary hsc || society culture preliminary hsc ka...
914                                     9781108448079 --  || cambridgemaths stage gk powers
944                                     9781108434638 --  || cambridgemaths stage gk powers
950                        9780170413220 --  || maths focus mathematics advanced margaret grove
692    9781922034717 -- extension mathematics || extension mathematics past hsc papers worked...
892    9780957742888 -- new advanced mathematics || new advanced mathematics complete hsc mat...
781    9780855837723 -- nsw biology || nsw biology module infectious disease module non infec...
dtype: string
```

[Software Design & Development]

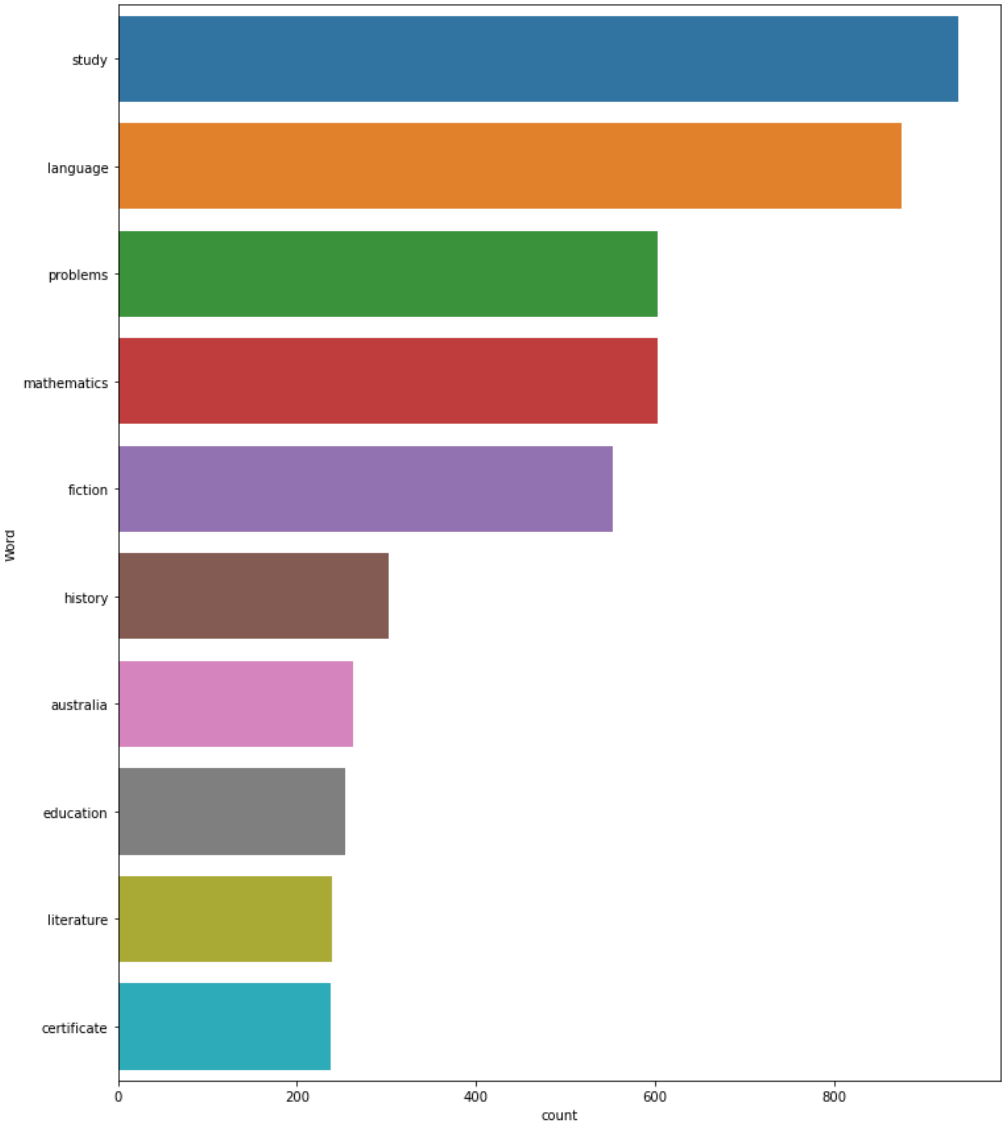
Appendix 2; Count of books by subject



Appendix 3; Top 10 count of distinct words for variable ‘all_string’



Appendix 4; Top 10 count of distinct words for variable ‘noun’



Appendix 5; Confusion matrix for test 1: *all_string*

Predicted	ENGLISH	MATHEMATICS	SCIENCE	All
True				
ACCOUNTING	2	0	0	2
AGRICULTURE	1	0	0	1
BUSINESS STUDIES	6	0	0	6
COMPUTER SCIENCE	2	0	0	2
DESIGN AND TECHNOLOGY	2	0	0	2
DRAMA	4	0	0	4
ECONOMICS	3	0	0	3
ENGLISH	61	0	0	61
ENGLISH LITERARY STUDIES	5	0	0	5
FOOD TECHNOLOGY	4	0	0	4
GEOGRAPHY	3	0	0	3
HISTORY	8	0	0	8
HUMANITIES	3	0	0	3
LANGUAGES	24	0	0	24
LEGAL STUDIES	2	0	0	2
MATHEMATICS	0	36	0	36
MUSIC	6	0	0	6
PDHPE	4	0	0	4
POLITICS AND LAW	2	0	0	2
PSYCHOLOGY	3	0	0	3
RELIGIOUS EDUCATION	3	0	0	3
SCIENCE	0	0	28	28
VISUAL ARTS	1	0	0	1
WORK STUDIES	1	0	0	1
All	150	36	28	214

Appendix 6: comparison of results from the content-recommender for “Beginner’s Bible” between the ‘noun’ variable and ‘all_string’ variable:

```
In [122]: #nouns only
          recommend_book_nouns(recc_isbn) # "Beginners Bible"

9      9781741352795 -- writing time f queensland beginner's al...
130    9780141316253 -- boy overboard || boy overboard morris g...
257    9781785583575 -- blitz beginner theory book || blitz beg...
98     9780224025720 -- matilda || matilda roald dahl illustrat...
331    9780199535064 -- complete short stories || complete shor...
33     9781741352801 -- writing time queensland beginner's alph...
123    9780670894383 -- game goose || game goose ursula dubosar...
180    9780141319001 -- girl underground || girl underground mo...
104                              9780849907685 -- holy bible ||
651    9781840222654 -- collected stories katherine mansfield |...
dtype: string
```

```
In [121]: #'raw' all_string
          recommend_book(recc_isbn) # "Beginners Bible"

528                              9780564098354 -- youth bible ||
298    9781599829234 -- catholic youth bible || catholic youth ...
104                              9780849907685 -- holy bible ||
556            9780970181671 -- heart wisdom teaching approach ||
288    9781599822211 -- breakthrough bible old testament activi...
63                           9781400316908 -- first study bible ||
130    9780141316253 -- boy overboard || boy overboard morris g...
151    9780061441714 -- nrsv catholic edition || holy bible new...
47     9780142410394 -- twits || twits roald dahl illustrated q...
257    9781785583575 -- blitz beginner theory book || blitz beg...
dtype: string
```

References

Afrin, T. (2001, 05 21). *Extraction of Basic Noun Phrases from Natural Language Using Statistical Context-Free Grammar*. Retrieved from semanticscholar: https://www.semanticscholar.org/paper/Extraction-of-Basic-Noun-Phrases-from-Natural-Using-Afrin/ca73eb2f183795ab2c106f654856b2e850d87aba

Bannerjee, P. (2020). *kaggle*. Retrieved from Naive Bayes Classifier in Python: https://www.kaggle.com/code/prashant111/naive-bayes-classifier-in-python

Chen, E. (2022, 07 14). *30% of Googles Emotions Dataset is Mislabeled*. Retrieved from surgehq: https://www.surgehq.ai/blog/30-percent-of-googles-reddit-emotions-dataset-is-mislabeled

Classifying Different Types of Recommender Systems. (2015, 11 14). Retrieved from bluepiit: https://www.bluepiit.com/blog/classifying-recommender-systems/

Etaiwi, W., & Naymat, G. (2017, 08 01). *The Impact of applying Different Preprocessing Steps on Review Spam Detection*. Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/pii/S1877050917317787?ref=cra_js_challenge&fr=RR-1

ML - Content Based Recommender System. (2020, 05 17). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/ml-content-based-recommender-system/

Muhammad Fuad, M. M., & Vijayakumar, B. (2019, 01). *A New Method to Identify Short-Text Authors Using Combinations of Machine Learning and Natural Language Processing Techniques*. Retrieved from researchgate: https://www.researchgate.net/figure/Accuracy-of-improved-classifiers-for-unigrams-and-bigrams_tbl3_336547614

Netflix Recommender Systems: How Netflix Uses NLP For Personalization. (2022, 01 30). Retrieved from ai-summary: https://www.ai-summary.com/summary-netflix-recommender-systems-how-netflix-uses-nlp-for-personalization/

Qaiser, S., & Ali, R. (2018, 07). *Text Mining: Use of TF-IDF to Examine the Relevance of*. Retrieved from researchgate: https://www.researchgate.net/publication/326425709_Text_Mining_Use_of_TF-IDF_to_Examine_the_Relevance_of_Words_to_Documents

Rani, R., & Lobiyal, D. K. (2020, 03 03). *Performance evaluation of text-mining models with Hindi stopwords lists*. Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/pii/S1319157820303232

Sarica, S. (2021, 08 5). *Stopwords in technical language processing*. Retrieved from journals: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0254937

Stemming and lemmatization. (2008). Retrieved from Stanford University: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html