NLP RECOMMENDATION ENGINE – MA3831

Building reading recommendation engines for students in Australian higher education – an investigation into the effectiveness of NLP techniques

Recommendation engines are an essential part of many of today's largest websites. Without them, it would be extremely difficult for users to find the products and content they are searching for. In Europe, it is possible to use existing and past reading list information from different University's to inform academics, librarians and publishers about comparative reading that might be applicable to any selected topic. This is not possible yet in Australia as no up-to-date central repository exists for reading list material. In this report, we examine the current approaches utilised for designing, training and evaluating recommendation engines based on textual data retrieved from the Trove API. We critically examine the existing methods and compare the performance of the content-based recommenders before and after implementing natural language processing techniques.

Nomenclature

NLP – Natural language processing

CBF – Content-based filtering

RE – Recommendation Engine

TF-IDF – Term Frequency – Inverse Document Frequency

Introduction

Content based recommendation systems use semantic information (frequently referred to as metadata) about the items in the system. In these systems, the features of the products, e.g., the title and the books contents, are represented most often in the form of a bag-of-words model. Features of a book can refer to its author, title, blurb, subject metadata, publisher, etc. Content-based recommendation engines utilise a variety of machine learning algorithms including kNN, Support Vector Machines and Naïve Bayes classifiers. As bag-of-words and matrix representations may contain hundreds of thousands of dimensions, techniques such as Singular Value Decomposition (SVD) may be employed to reduce dimensionality. The content may also benefit from natural language processing techniques to make use of semantic characteristics. Currently, university lecturers rely on librarians to do background research on relevant, compliant, and current reading material suitable for course teaching units as no engine capable of recommending suitable readings for a specified course exists. The ability to recommend reading material down to a book and at a journal level would be a useful product for academics, publishers and agencies supporting higher education. In this case, the input to the engine will be a new resource title and the output will be a list of recommended courses sorted by their relevance.

Dataset

The supplied dataset is a compilation of 45,755 unique resources from 3,657 university courses. The dataset contains the resource list for a number of university courses, the resource major title, minor title, an array of universal identifiers as

well as information regarding the authors and publication of the resource. Table 1 provides a summary of the 19 variables.

Variable	Descriptions
ID	University ID
COURSENAME	Name of Course
ITEM_COUNT	Number of items in reading list
TITLE	Major Title (Book, journal)
RESOURCE_TYPE	Book, journal, etc
SUBTITLE	Minor Title (article)
ISBN10S	Universal Identifiers
ISBN13S	Universal Identifiers
ISSNS	Universal Identifiers
EISSNS	Universal Identifiers
DOI	Digital Object Identifier
EDITION	Edition of publication
EDITORS	Names of editors
PUBLISHER	Publisher
DATES	Publication date
VOLUME	
PAGE_END	Pages selected
AUTHORS	Authors

Table 1: Data element dictionary

The data provided is misaligned, incomplete and messy. However, the ID, COURSENAME, TITLE, SUBTITLE and RESOURCE_TYPE are reliable fields that can be used throughout the task. It is worth noting that provided data set was insufficient to develop a recommendation engine or provide assessments of the recommendation systems quality. As such, the supplied data was supplemented with data retrieved from the Trove API.

Dataset Critiques

A range of data science methods were implemented to pre-process the original dataset before it could be augmented with data from an external API. The integrated development environment (IDE) PyCharm was utilised along with its associated language, Python. A select few third-party libraries were installed enabling the data-wrangling process to be made more seamless, namely: **re, pandas.**

The original dataset was stored in a Microsoft Excel (.xlsx) file. This file was first converted to a 'comma-separated values' (.csv) file. By making use of pandas **read_csv** function, the data set can be read into the python script. The original dataset contained 19 columns. Of these, only 4 were necessary for later tasks: COURSENAME, TITLE, SUBTITLE and RESOURCE_TYPE with the remaining columns being dropped. A new column titled COMBINED_TITLE was produced using a lambda function. This function combined the TITLE and SUBTITLE if they were not equal otherwise just the TITLE was used to populate the column. After the creation of this column, the TITLE and SUBTITLE columns were dropped, leaving a data frame with the columns: COURSENAME, COMBINED_TITLE and RESOURCE_TYPE.

The data frame was then filtered to remove any null RESOURCE_TYPES before each value in the data frame was passed through a series of regular expressions. These regular expressions converted the fields to lowercase before replacing any characters that were not in the range a-z, 0-9 or a single quote or double quote with a space. Single and double quotes were then removed. The replacing of unwanted characters with a single space led to the injection of double spaces throughout the data frame. As such another regular expression was employed to remove any

occurrence of double spaces. All elements were finally striped of leading and trailing spaces before the RESOURCE_TYPES were filtered to only contain “books”, “articles” and “journals”.

A final filtering was then implemented so that the dataset contained resources that only appeared in a single course. This filtration was employed to assist in the training of the model as having the same resource with the same subject metadata appear in two different course names will lead to a higher error in the recommendation engine. Due to hardware constraints, the subset was further reduced by removing the courses that utilised less than 3 resources. After filtering, the data, from here on referred to as the pre-processed data, contained 1,176 courses and 18,762 unique resources.

Resource Augmentation

The provided dataset required the creation of a labelled dataset from the ontology and supplementation of the dataset from an external API. The API chosen was the Trove API from the National Library of Australia. Trove is an Australian online library database aggregator and service which includes full text documents, digital images, bibliographic and holdings data of items which are not available digitally (Wikipedia contributors, 2021).

For each of the unique resources in the cleaned dataset, the Trove API was used to obtain the subject metadata. This process first required the querying of the Trove’s version 2.2 search endpoint with a list of parameters. Table 2 provides a description of the parameters used.

Parameter	Description
q	The resource COMBINED_TITLE
zone	The RESOURCE_TYPE
n	The number of resources to return (1)
encoding	The return data format (json)
key	The users Trove API key (sike)

Table 2: Trove API parameters

This first query searches the Trove database for an **n** number of resources matching the resource name passed in the **q** parameter in the zone matching that passed in the **zone** parameter. As the value for **n** passed into the search parameters was one (1), only a single result is returned. The work ID of this returned result is then used to query a second Trove endpoint, the version 2.2 metadata API. This endpoint only uses the **encoding** and **json** parameters from the first query whilst adding a third parameter, **reclevel**. Setting the **reclevel** parameter to ‘full’ returns all the stored metadata tags for the work id. The possible formats for the subject metatags can be either a single string containing all the subjects, or a list of singular subject tags. As such, each resource’s subject tags were converted to single string that then underwent a similar cleaning process to that of the main data set; converting the string to lowercase, removing symbols and removing double spaces. This data was then stored in a new column titled. The pre-processed data was then merged with the augmented resource list to create a dataset with 18,762 resources and 4 columns: COURSENAME, COMBINED_TITLE, RESOURCE_TYPE, SUBJECTS**.** This merged data, from here on referred to as the utilised dataset, was also saved as a csv titled ‘merged_data.csv’

NLP Techniques

A multitude of natural language processing techniques were utilised on the sub-setted data set. These techniques included: stopword removal, lemmatization, and the use of n-grams.

Stopword removal refers to the process of removing uninformative words such as: “and”, “the”, “if”. A predefined list of English stopwords exists in the NLTK.corpus package. A brief manual examination of the subject metadata showed that the tags “periodicals” and “journal” appeared in a large portion of the journal and article resource types. As such, this word was appended to the stopwords list as it provided minimal information on the content of the article.

Lemmatization reduces the inflected words ensuring that the root word belongs to the English language. This root word is called a lemma. This process is significantly slower than stemming. However, the dataset is small enough that the extra computational time did not impact the timeline of the project.

n-grams are contiguous sequences of an n number of subjects taken from the subject metadata. For this task, unigrams and bigrams are explored.

Exploratory Analysis

There are 18,762 resources in our utilised data set. We implemented the standard 80% of randomly sampled resources for training the model and the remaining 20% for testing the engine. This test set contained 3,753 rows and 17,614 columns. As shown in Figure 1, the number of subject tags per resource varies significantly but centres around a mean value of approximately 15 words per resource.

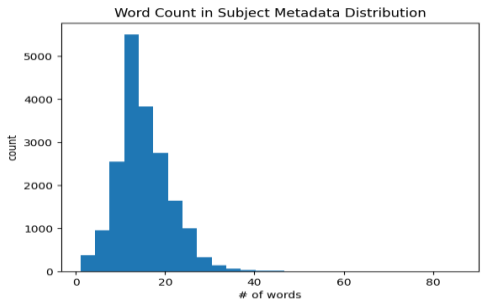

Figure 1: Distribution of the number of subject tags per resource

We can also visualise the top 5 most common unigram and bigram occurrences in the dataset. As shown in Figure 2 and Figure 3, the most commonly occurring words provided little to no information on the contents of the resource and prove to be a perfect use case for stopword removal. After removing the stopwords, the top 5 most common subjects tags were: “australia”, “social, “management”, “law”, “history”; subject tags that provide a greater deal of information about the resource.

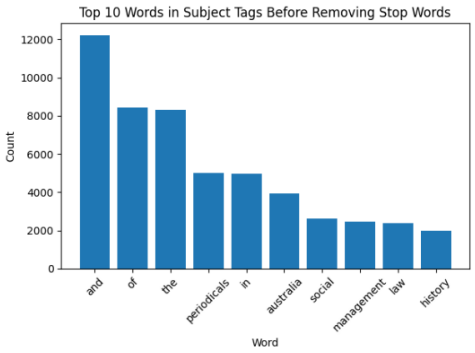

Figure 2: Top 10 most occurrences before stopword removal

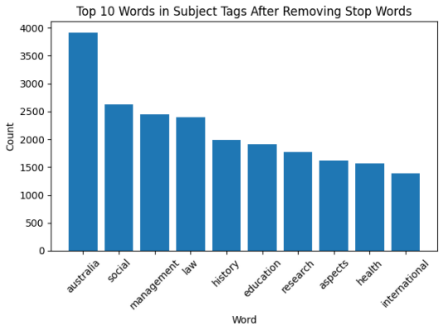

Figure 3: Top 10 most occurrences after stopword removal

Removing stopwords also reduces the number of unique words in the subject tags to 17,496 from 17,614.

Implementing both lemmatization and stopword removal on the utilised dataset results in 15,402 unique subject tags and a slight reshuffle of the top 10 most common subject tags as illustrated in Figure 4.

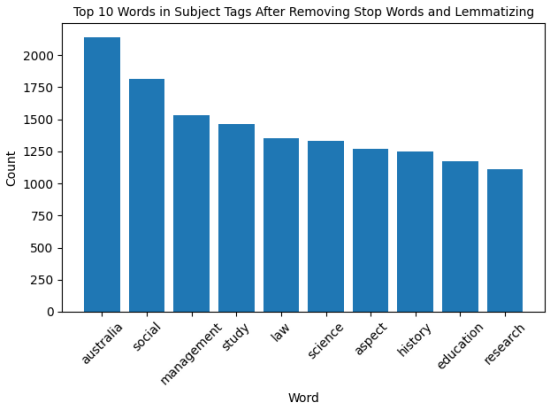

Figure 4: Top 10 most occurrences after stopwords removal and lemmatization

Figures 5, 6 and 7 illustrate the most common bigrams before stopword removal, after stopword removal and after lemmatization, respectively.

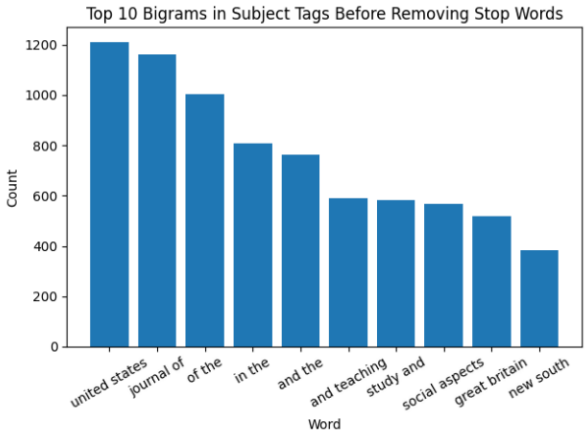

Figure 5: Top 10 most occurring bigrams before stopwords removal

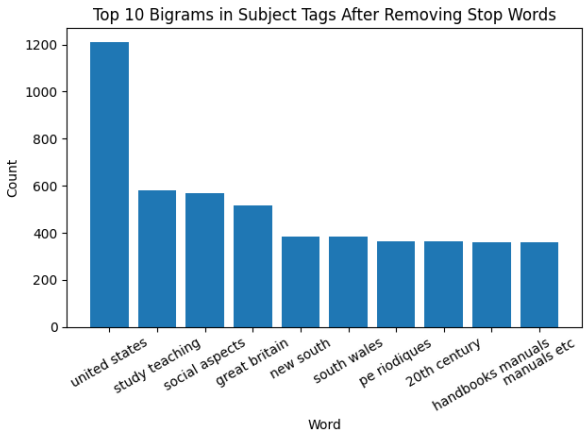

Figure 6: Top 10 most occurring bigrams after stopwords removal

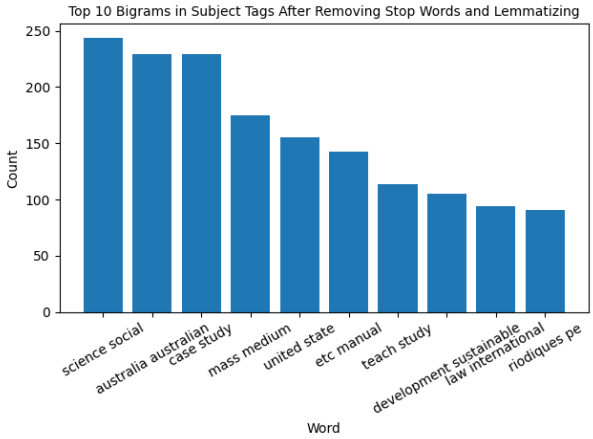

Figure 7: Top 10 most occurring bigrams after stopwords removal and lemmatization

The bigrams present after lemmatizing the complete dataset prove to be much less intuitive than those seen before and after stopword removal. This can be attributed to the lemmatizing and Parts of Speech (POS) tagging functions used which may not be fully accurate in the tag it assigns to a word, resulting in an incorrect lemma being chosen.

Classification Algorithm

We used the python library, Scikit-learn which is a package that contains various classification and clustering algorithms. Of particular interest are k Nearest

Neighbours (kNN), Support Vector Machines (SVM) and Gaussian Naïve Bayes (GNB) classifiers. kNN is one of many supervised learning machine learning classifier algorithms where the learning is based on the Euclidean distance of one data point from another (José, 2019). The Euclidian distance can be defined as:

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

The Support Vector Machine algorithm aims to find a hyperplane in an N-dimensional space such that the hyperplane has the maximum margin between the data points of each class (Gandhi, 2018).

Gaussian Naïve Bayes is an extension of the Naïve Bayes classifier that assumes the predictors are sampled from a Gaussian distribution. A Naïve Bayes classifier is a probabilistic machine learning model that is based on Bayes theorem, finding the probability of y happening, given that X has occurred with X being the evidence and y being the hypothesis.

$$P(y_i|X) = \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{(-\frac{(y_i-\mu_i)^2}{2\sigma_X^2})}$$

In this case, y, is the course name while, X, represents the features. As the denominator remains static, it can be removed, and a proportionality introduced.

$$X = (x_1, x_2, x_3, \dots, x_n)$$

$$P(y|x_1, \dots, x_n) \propto P(y)\prod_{i=1}^{n} P(y_i|X)$$

The problem is also multivariate, as such, we need to find the class, y, with the maximum probability (Gandhi, 2018).

$$y = argmax_y P(y)\prod_{i=1}^{n} P(y_i|X)$$

A multitude of other classifiers exist however, only these three were selected for further experimentation. The scikit-learn package also contains functions for analysing the accuracy, precision and recall of the algorithms, with accuracy being employed as the evaluation metric. Our experiments compared the accuracy for unigrams and a combination of unigrams and bigrams, with the value of k in the kNN varying from 1 to 10. An overview of the best results is shown in Table 3.

Unigrams	
kNN	14.58% @ k=1
SVM	15.87%
GNB	19.26%
Unigrams & Bigrams	
kNN	11.83% @ k=1
SVM	15.47%
GNB	23.10%

Table 3: Normal data accuracy.

From the table above, it can be seen that the Gaussian Naïve Bayes model employing the use of just unigrams proved to be the most accurate model by recommending the correct university course 19.26% of the time when supplied with a list of subject metadata tags.

Utilising Trigrams was not viable as the feature set became too large and when attempting to train the algorithms, we ran into memory errors. While GNB appears to be a superior classifier, its' training and prediction time was significantly larger

than the other algorithms especially when utilising unigrams and bigrams as this data set contained 117,819 columns compared to the 17,614 from just unigrams. The full time taken breakdown can be seen in Table 4. Note: for kNN and SVM, the mean fit and predict times are used.

Classifier	Fit time (s)	Predict time (s)
Unigrams		
kNN	0.020	1.506
SVM	5.079	7.254
GNB	5.673	720.669
Unigrams & Bigrams		
kNN	0.020	1.434
SVM	5.789	7.079
GNB	207.18	5152.279

Table 4: Normal data Fit and Predict times.

NLP Improvements

Implementing the same classifiers on the utilised data set after stopword removal as well as after lemmatization provided varying results. As removing stopwords reduced the dimensionality of the vectorized unigram testing data set by 118 columns, the unique differences between each resource was also reduced. The bigram testing matrix contained 112,434 columns. This resulted in a slightly higher accuracy score for all the models. Table 5 shows the full results from testing stopword removal.

Unigrams	
kNN	14.97% @ k=1
SVM	16.91%
GNB	19.77%
Unigrams & Bigrams	
kNN	12.15% @ k=1
SVM	15.71%
GNB	23.29%

Table 5: Data with stopwords removed accuracy.

The reduced dimensionality did improve the fit and predict times when compared to the dataset that did not have its stopwords removed as seen in Table 6.

Classifier	Fit time (s)	Predict time (s)
Unigrams		
kNN	0.028	1.800
SVM	4.316	6.502
GNB	5.341	701.25
Unigrams & Bigrams		
kNN	0.019	1.12
SVM	4.862	6.514
GNB	186.035	4766.628

Table 6: Data with stopwords removed Fit and Predict times.

Lemmatization further reduced dimensionality for both unigrams and the combination of unigrams and bigrams with unigrams having 15,402 columns and the combination testing data having 105,357 columns. This resulted in a lower accuracy for all the models as seen in Table 7. The fit and predict times also decreased for every classifier as can be seen in Table 8.

Unigrams	
kNN	12.34% @ k=1
SVM	16.91%
GNB	18.71%
Unigrams & Bigrams	
kNN	6.18% @ k=1
SVM	14.38%
GNB	20.33%

Table 7: Data with stopwords removed and lemmatized accuracy.

Classifier	Fit time (s)	Predict time (s)
Unigrams		
kNN	0.021	1.216
SVM	4.293	6.272
GNB	4.410	595.823
Unigrams & Bigrams		
kNN	0.020	1.155
SVM	4.689	6.307
GNB	170.735	4460.863

Table 8: Data with stopwords removed and lemmatized fit and predict times.

From the results presented, removing stopwords increased the accuracy of the classifier whilst reducing the fit and

predict times. Lemmatizing the data further reduced the fit and predict times but lead to a reduction in accuracy. The most appropriate classifier would be one that recommends the most appropriate course with the highest accuracy, as such, the Gaussian Naïve Bayes classifier utilising unigrams and bigrams from the dataset with stopwords removed.

Conclusion

A plethora of natural language processing techniques exist such as, stopword removal, lemmatization and n-grams, which can be employed in the content-based recommendation of university resources. A recommendation engine built for informing academics and librarians about comparative reading that might be applicable to any selected topic would prove invaluable and significantly reduce the research time taken for material selection. Such an engine could be built using a classifier such as Gaussian Naïve Bayes which proves to be accurate, albeit computationally slow. This recommendation engine will also benefit from some Natural Language Processing techniques to reduce dimensionality and improve accuracy. The only technique that would be implemented from the tested techniques would be stopword removal as both lemmatization and the use of n-grams proved to reduce the accuracy of the system. The implemented RE will need to take a resource title as an input as well as the resource type (book, journal, article), from which the Trove API can be used to obtain the resources subject metadata tags. These tags would then be fed through a function to remove stopwords before being passed into a Gaussian Naïve Bayes classifier. The classifier will then return a list of the most relevant university courses sorted by the probability estimate.

References

Wikipedia contributors. (2021, February 21). *Trove*. Wikipedia. https://en.wikipedia.org/wiki/Trove

José, I. (2019, June 26). *KNN (K-Nearest Neighbors) #1 - Towards Data Science*. Towards Data Science. https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d#:%7E:text=KNN%20(K%20%E2%80%94%20Nearest%20Neighbors),(a%20vector)%20from%20other%20.

Gandhi, R. (2018b, July 5). *Support Vector Machine — Introduction to Machine Learning Algorithms*. Towards Data Science. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Gandhi, R. (2018a, June 26). *Naive Bayes Classifier - Towards Data Science*. Towards Data Science. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c