Classifying it the right way!

Anvesh Koganti, Aditya Nair, Nitish Dewan, Sharath S Bhargav

1 Introduction

A book can be classified in multiple ways based on different attributes and in various categories. One such classification is the Library of Congress classification (LoCC). It serves as the de facto classification system across most major libraries. In this work, we present a system that can classify a text document into one of the classes from the LoCC. During the course of the project, we tried multiple approaches to find the most suitable techniques that can be applied to classify a text. Trained on the open-sourced books dataset from Project Gutenberg, we have used classification techniques such as k-nearest neighbours, decision trees, Support Vector Machines, Multi-Layer Perceptrons etc. along with different type of word embedding, which include Bag of Words, TF-IDF etc. in our attempts to determine the best combination that classifies a book well. Additionally, we have drawn insights from the books and their contents to perform initial analysis on book length, distribution of content of book relative to each other etc. by using various visualising techniques.

This project aims to help book-houses and libraries categorize new/existing e-books confidently without having to rely on external sources or manual reading. It can also help authors writers categorize their piece of work appropriately, which might otherwise be cumbersome and time-consuming due to factors like multiple genres or their own bias. It could possibly help them visualize if the contents of their works actually aligns with the genre they originally planned.

2 Data

The vast number of books available publicly (licence-free) on Project Gutenberg have been used as part of this project. The books available on Gutenberg are mapped to Library of Congress Classification (LoCC) (Ref. Appendix Figure 8), which is a universal standard for book classification in all major libraries. Using the credible data available on Project Gutenberg and LoCC, we aim to use machine learning to evaluate the text and predict the class for a document.

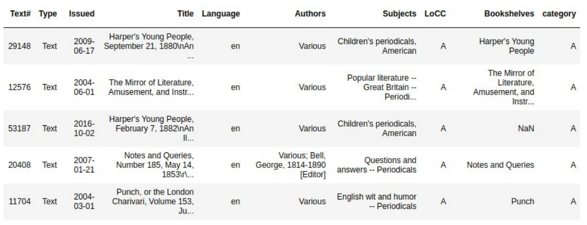
Project Gutenberg repository contains more than 50,000 books spanning across multiple genres and languages. We have chosen to sample 1400 English books across the entire collection such that each class has a relatively equal representation. The books we have used are UTF-8 encoded text files.

Once we have the text files for each of the books, we need the book-LoCC mapping information as well. This mapping information is present in a CSV file available on the Gutenberg website. We chose to link the books and their corresponding record in the CSV file based on the unique book IDs. The LoCC tag has 2 letters followed by a series of numbers (eg. AS213). We chose to only consider the first letter of the LoCC tag because it represents the primary class the book belongs to. For instance a book might have a LoCC tag 'TPxxx'. The first letter 'T' signifies that the book belongs to the class 'Technology' and the second letter 'P' gives information about the sub-class Here 'P' signifies a subclass 'Chemical Technology' under the parent 'Technology' class. For the scope of this project, we have chosen to limit the model to just main class prediction owing to computational limitations.

Apart from the LoCC related information we have a lot of metadata related to each book in the mapped CSV file. It contains issue date (date on which the book was hosted on the Gutenberg repository), Title, Author name(s), Subjects, and Bookshelves.

3 Analysis/EDA

Exploratory Data Analysis (EDA) was performed on multiple features of the dataset. The results are as following.



Text#	Type	Issued	Title	Language	Authors	Subjects	LoCC	Bookshelves	category
29148	Text	2009-06-17	Harper's Young People, September 21, 1880\nAn ...	en	Various	Children's periodicals, American	A	Harper's Young People	A
12576	Text	2004-06-01	The Mirror of Literature, Amusement, and Instr...	en	Various	Popular literature -- Great Britain -- Periodi...	A	The Mirror of Literature, Amusement, and Instr...	A
53187	Text	2016-10-02	Harper's Young People, February 7, 1882\nAn Il...	en	Various	Children's periodicals, American	A	NaN	A
20408	Text	2007-01-21	Notes and Queries, Number 185, May 14, 1853\r\...	en	Various; Bell, George, 1814-1890 [Editor]	Questions and answers -- Periodicals	A	Notes and Queries	A
11704	Text	2004-03-01	Punch, or the London Charivari, Volume 153, Ju...	en	Various	English wit and humor -- Periodicals	A	Punch	A

Figure 1: Sample of input data.

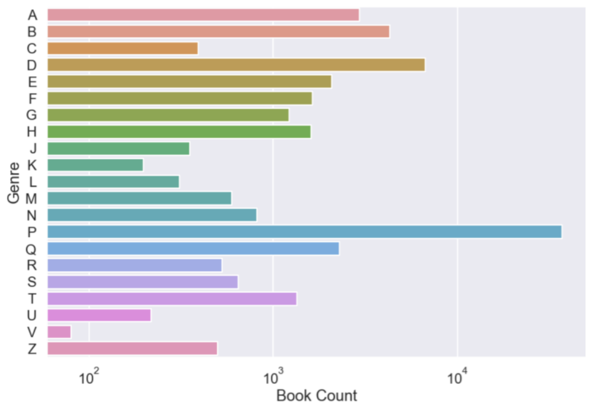


Figure 2: Book count across genre

EDA performed on number of books per Class shows that the corpus has the highest number of books for 'Language and Literature' class. The least number of books is found in class 'Naval Science' class.

4 Methods

4.1 Data Retrieval

The data was collected using an API for Project Gutenberg. The content and other meta-data of selected books was extracted based on book ID. Web scraping was performed using beautiful soup to get the mapping of LoCC to the respective genres.

4.1.1 Books

The python based 'GutenbergPy' package was used to retrieve text files for each book. After installing the package onto the local system, it needs to build a local cache of a few parameters related to Gutenberg before it can be used. The package needs either a SQLite or a MongoDB database to store the cache. The cache build time and cache size using SQLite is much lower than the alternative and therefore this was chosen. So, a SQLite database with the required database schema has been set up. The package needs to cache some metadata related to all the books in the Gutenberg repository before being able to download them. The caching operation took approximately 5 minutes to complete and once it was done, the system was ready to download any text from the Gutenberg repository just using the book ID.

The book IDs corresponding to 1400 sampled books were iterated upon and function calls were made to retrieve the book text based on book ID were made. The text corresponding to every book was stored as a pickle file for pre-processing at a later stage.

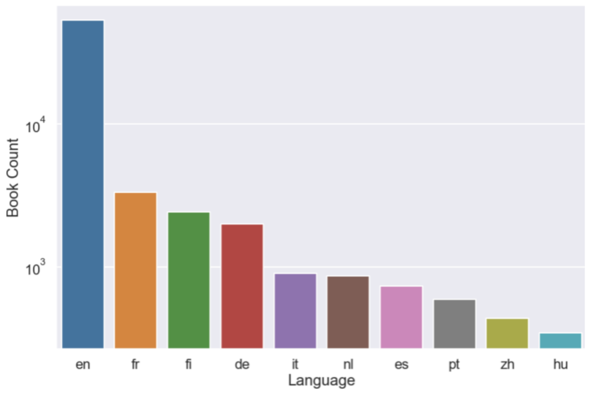


Figure 3: Book count across languages
Plot shows the distribution of books across different languages. Since a good enough number of books are in English (en) language, the other language books were discarded.

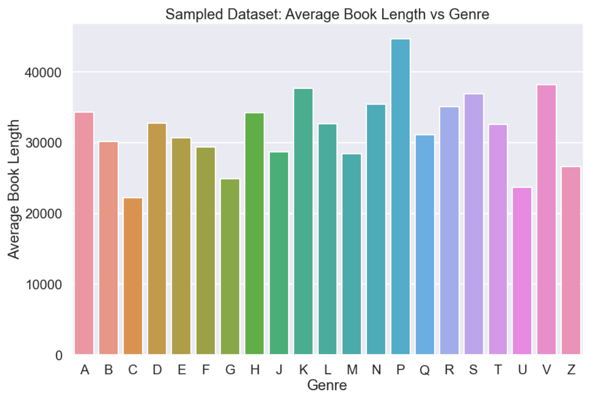


Figure 4: Word count across genre(post pre-processing). Graph visualizes the average Word Count across each Class. This shows that the average book length varies drastically depending on class.

4.1.2 LoCC Mapping

The mapping CSV file has only the first letter of the LoC classification for each book. In order to derive meaningful outcomes from the analysis, having the full form of the abbreviations used is very important. This information is present on the

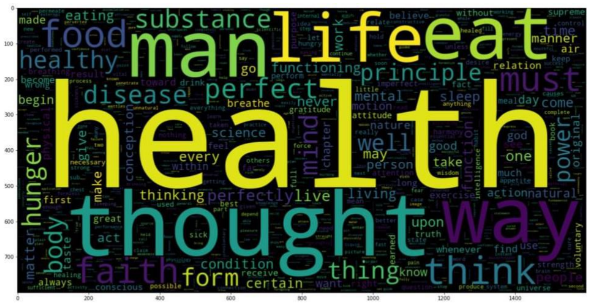


Figure 5: Word Cloud for a book on ‘Medicine’ Class.
We see common words like ’Health’, ’Life’, ’man’, ’eat, etc correlating with medicine

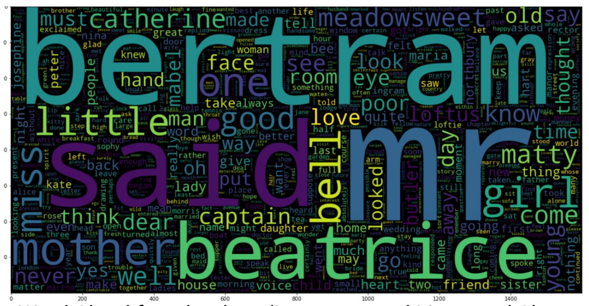


Figure 6: Word Cloud for a book on ‘Language and Literature’ Class
We see proper nouns like ’bertram’, ’beatrice’, ’bell’, etc possibly indicating Shakespearean influence

Library of Congress Classification Outline Number page. A web scraper was built to read the page and retrieve the full form corresponding to each letter using functions from the ‘BeautifulSoup’ package. This data was stored in a dictionary for further use in EDA and deriving insights from the model output.

4.2 Data Cleanup

The pickle files of the books were loaded and run through a pre-processing pipeline one at a time. The text is converted to lowercase and is tokenized using the ‘word_tokeize’ function from the ‘nltk.tokenize’ package. Then the contractions present in the tokens are expanded using functions from the ‘contractions’ library. The punctuation marks are then removed followed by removal of the common english stop words from the ‘nltk stopwords’. Lemmatization is then performed using methods from the ‘WordNetLemmatizer’ package. The resultant tokens are stemmed using methods from the ‘PorterStemmer’ package. Finally any tokens having length less than 2 characters are removed. These cleaned tokens are stored as pickle files for use during the feature engineering stage.

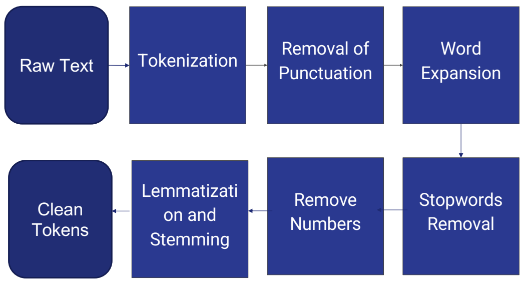


Figure 7: Pre-Processing pipeline

4.3 Feature Engineering

For pre-processed data to be used with training models they had need to converted into vectors. The following word embedding techniques were used to convert them into vectors.

1. **Bag of Words:** The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

2. **TFIDF**: Short for term frequency–inverse document frequency, TFIDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

3. **Word2Vec**: Word2vec represents each distinct word as vectors that are chosen carefully such that a simple mathematical function indicates the level of semantic similarity between the words represented by those vectors.
   We’ve used 2 different kinds of Word2Vec models

   - Self-trained Word2Vec model: A Word2Vec model trained on a corpus of Gutneberg books
   - Pre-trained Word2Vec model: An open-sourced pre-trained Word2Vec model provided by Google, trained on new articles.

Bag Of Words and TFIDF returned approximately 380k features per book. We used Singular Value Decomposition (SVD) dimensional reduction technique to reduce the number of features from 380k to 300. In comparison, Word2Vec mddels were using 300 features, formed using dimensionality reduction.

4.4 Models

Machine Learning models are classified into two main categories - supervised learning and unsupervised learning. Supervised Learning is used to solve classification and regression problems. Unsupervised Learning, on the other hand, is mainly used for getting insights for an unlabeled data in the form of clustering. A supervised learning algorithm is based on the idea that a model can be trained to learn from existing labelled data and predict a class or a value for an unlabeled dataset. Various supervised learning techniques are currently in use, however in this project, we have employed only seven of those.

1. **k Nearest Neighbor:** The k Nearest Neighbor algorithm, or kNN, is a simple algorithm that stores all available classes and classifies new data by a majority vote of its k neighbors. The data gets assigned to the class that is most common amongst its K nearest neighbors measured by a distance function.

2. **Logistic Regression:** The Logistic Regression algorithm is used to estimate discrete values based on a given set of independent attributes. It predicts the probability of occurrence of an event by fitting data to a logit function.

3. **Support Vector Machines:** In Support Vector Machines, we plot each data item as a point in high-dimensional space. The idea behind SVM is that the data which may seem inseparable in the present set of dimensions might become separable in a high dimension space. A decision boundary is generated to classify various classes in this high dimensional space.

4. **Decision Tree (Random Forest):** Decision Trees is a common classification technique. It allows the dataset to be split on various attributes at different levels. Once splitting is done, label for a new test data is predicted using this split. In Random Forest, we have a collection of decision trees.

5. **Naive Bayes:** This classification technique is based on Bayes' Theorem with the primary assumption that the attributes are independent. The technique involves calculating the probability of each label occurring given a specific attribute.

6. **Logistic Regression One-vs-Rest:** The standard Logistic Regression classifier works as a multi-class classifier, and training occurs based on a multi-class dataset. However, using a one-vs-rest approach, the classifier is trained by considering one label at a time as the positive class and remaining classes account for the negative class. This becomes the case of binary classification. The process is repeated for all the classes available, and the overall trained model is then used further.

7. **Support Vector Machines One-vs-Rest:** Similar to Logistic Regression one-vs-rest approach, this model convert multi-class classification to a case of binary classification. The trained model on all classes is then used for further prediction.

8. **Neural Networks:** Neural Networks is primarily an artificial network of functions, and parameters. The parameters include weights which are trained on every iteration. Often called Neurons, these take a combined input from the previous layer of neurons to generate a new output. A final layer produces the classification.

5 Results

Table 1 shows the accuracy scores of using different models with different word embedding techniques. TF-IDF vectorization embedding with a non-linear one-vs-rest approach model has the highest accuracy of 70%. For small textual data Non-Linear SVM models perform better than Multiperceptron classifiers. Our experiments show that we can achieve an accuracy of atleast 65% when we use weighted vectorization techniques. Table 2 shows the F1 scores. While TF-IDF vectorization technique is old, it does perform well as can be seen from both accuracy and F1 scores.

Table 1: Accuracy of different models with various embeddings

	k Nearest Neighbor	Support Vector Machines	Decision Tree (Random Forest)	Naive Bayes	Logistic Regression	Logistic Regression One-vs-rest Approach	Non-Linear SVM One-vs-rest Approach	Neural Networks
Bag of Words (BoW) - Word Count	0.4285714286	0.05	0.6071428571	0.2107142857	0.6142857143	0.5392857143	0.625	0.5428571429
TF-IDF Vectorization	0.5392857143	0.6571428571	0.6892857143	0.5321428571	0.6535714286	0.6392857143	0.7	0.6464285714
Self-trained Word2Vec Model	0.55	0.4892857143	0.6285714286	0.4892857143	0.6392857143	0.6428571429	0.6142857143	0.65
Pre-trained Word2Vec model by Google	0.5428571429	0.5821428571	0.6535714286	0.5035714286	0.4821428571	0.475	0.625	0.6785714286

Table 2: F1 score of different models with various embeddings

	k Nearest Neighbor	Support Vector Machines	Decision Tree (Random Forest)	Naive Bayes	Logistic Regression	Logistic Regression One-vs-rest Approach	Non-Linear SVM One-vs-rest Approach	Neural Networks
Bag of Words (BoW) - Word Count	0.444203282	0.03279067811	0.5958331971	0.208260289	0.5996964874	0.527725021	0.6033399913	0.5275732668
TF-IDF Vectorization	0.538481074	0.6419368736	0.6751092638	0.5071116361	0.6338118653	0.6148766518	0.6750613067	0.6315369599
Self-trained Word2Vec Model	0.5610251176	0.5328085688	0.6158991544	0.4798100172	0.6244677709	0.6260822739	0.588705615	0.6435646396
Pre-trained Word2Vec model by Google	0.5490466562	0.5741386743	0.6382242668	0.4872676088	0.4593979439	0.4527116653	0.5971072905	0.6639768072

6 Conclusion

While advanced techniques like deep neural networks, transformers are creating big waves in the domain of natural language processing, our project shows simple models like Non-Linear SVM and Random Forest work reasonably well. The trained models can be further fine-tuned depending on type of documents we want to classify into LoCC.

Full code can be found at https://github.com/sharathbhargav/CS418

Appendix

```
A=GENERAL WORKS
B=PHILOSOPHY.  PSYCHOLOGY.  RELIGION
C=AUXILIARY SCIENCES OF HISTORY
D=WORLD HISTORY AND HISTORY OF
EUROPE, ASIA, AFRICA, AUSTRALIA, NEW ZEALAND, ETC.
E=HISTORY OF THE AMERICAS
F=HISTORY OF THE AMERICAS
G=GEOGRAPHY.  ANTHROPOLOGY.
RECREATION
H=SOCIAL SCIENCES
J=POLITICAL SCIENCE
K=LAW
L=EDUCATION
M=MUSIC AND BOOKS ON MUSIC
N=FINE ARTS
P=LANGUAGE AND LITERATURE
Q=SCIENCE
R=MEDICINE
S=AGRICULTURE
T=TECHNOLOGY
U=MILITARY SCIENCE
V=NAVAL SCIENCE
Z=BIBLIOGRAPHY.  LIBRARY SCIENCE.  INFORMATION
RESOURCES (GENERAL)
```

Figure 8: LoCC mapping