FINAL REPORT

Building a Book Genre Classifier Model

Kavya Jeganathan, Nikhil Devadoss, Dat Duong

Brendan O'Connor

Compsci 490A

University of Massachusetts, Amherst

2 December 2020

Project Description

Abstract

This project is a text classification system that serves as a reliable way to label books by their genres given solely their summaries. Our finalized approach to solve this problem was to utilize various machine learning algorithms such as Random Forest, Support Vector Machines (SVM), and Logistic Regression to see if we could accurately predict the genre of a book using its summary. After running our models, we found that it is possible to predict the genre of a book using just its summary with reasonable accuracy.

Introduction

This project serves as a way to classify books by genres based off of a summary of each book. This project is one that has a useful real world application because of the expansive nature of book genres. There are many different types of genres, and a book has the potential to fall under many of these genres, especially due to the fact that some genres are incredibly similar and sometimes indistinguishable. With all this in mind, the task of labeling books as a certain genre is nontrivial, which is something we aimed to solve using text classification. We decided that we were going to use text classification to see if we could determine the genre of a book based purely off of its summary. In order to do so, we decided to use multiple machine learning models (Logistic Regression, SVM, Random Forest) to see which model would work best for our specific needs. Our project was driven by the overarching question:is it possible to classify books by their genres based purely off of their summaries in an accurate manner? After creating, training, and testing our models, our results showed us that it is definitely possible and can be done with reasonable accuracy.

Related Work

- The project *Book Genre Classification with BERT* by Peltarior platforms uses a BERT based binary classification model on the CMU book summary dataset which we have used in our project as well in order to determine whether a book is of the genre science fiction or not. The CMU book summary dataset has summaries of over 16,000 books with multiple labelled genres, but for this project the dataset was modified in order to only have 2,500 books that fell into the science fiction genre and 2,500 which were not science fiction. For feature encoding, the project uses a SummaryCropped feature which uses the Text encoding present in the dataset, and a Science Fiction feature which uses the binary encoding present in the dataset, so that 1 is the positive class and 0 is the negative class. The train-test split for this project was 80-20. For BERT, sentence

vectors of cropped summaries were used for classification. The binary classifier accuracy was around 85% which is quite a high number.

- The project Classification of Book Genres by Stanford Students Holly Chiang, Yifan Ge and Connie Wu aims at classifying books into five different categories, namely, business, fantasy, history, science-fiction, or romance, by using the cover of a book. The dataset used for this project was “the original book cover images from the openLibrary API (Chiang, Ge, et al., 2015).” The images of the book covers were preprocessed using image processing techniques which “removed empty or plain text book covers for a higher quality dataset (Chiang, Ge, et al., 2015).” Features were extracted from the book covers using two NLP classifiers, namely, the Stanford NLP classifier (a probabilistic softmax classifier) and the word2vec classifier. “The word2vec classifier produces word vectors from a text corpus using a continuous bag of words (Chiang, Ge, et al., 2015).” Classification for both NLP methods were done on book titles and results show that the precision and recall of word2vec classifier was higher than that of the Stanford NLP classifier, which led to a higher f1-score for word2vec classifier.

Data

Initial Dataset

The dataset that was used was from Carnegie Mellon University which was CMU Book Summary Dataset (http://www.cs.cmu.edu/~dbamman/booksummaries.html). The original data is also available on Kaggle. The original dataset contained a total of 16,559 books with plot summaries from Wikipedia as well as other metadata like author, genre, and book id. The decision to use this specific dataset because it used multiple labels per summary, which gives a good spread of summaries in each label. If each summary only had a singular label, there would be a significant disparity between the number of summaries in some labels compared to others. Another aspect is most books are not constrained to one genre and are also subjective. For this project there was some preprocessing and data cleaning involved to this dataset to simplify the task for this project.

Data Cleaning

There were some data points with missing data such as no authors, no genres, or no summaries so those data points were removed. For data cleaning the features were reduced to the following 4 features: “book_name”, “author”, “genre”, “summary”. Genre in this case is the feature to classify and label for each book. For the feature, genre, there were notably many genres and some that were special genres or genres in which the sample size seemed small relative to others. For example, there were genres like

“Poetry” and “Non-fiction” that seemed to be rare. Other genres such as “Adventure Novel” and “Romance Novel” had a decent sample size but not as big compared to the genres chosen for this project. Thus, the set of genres was reduced to the following 5 genres: “Children’s Literature”, “Fantasy”, “Fiction”, “Mystery”, “Science Fiction”. These set of genres seemed to have an even distribution of books. Having an even distribution of genres running NLP machine learning models will result in a better classification of each genre as there are no rare occurrences of a genre. This also simplified the multilabel classification as more genres would make the task difficult. Notably genres that were similar to “Mystery” such as “Crime Fiction”, “Detective Fiction”, “Horror” were changed to “Mystery”. For every other genre it was removed and if a book did not contain any of the 5 genres it was removed as well to reduce unwanted books.

Preprocessing of Features

There was other preprocessing involved before running classification tasks. For the “author” feature, each author was modified into a one hot encoding vector. By turning the authors into a hot vector, it may help with classifying books especially with books by the same author who wrote the same genre. For “book_name” and “summary”, the title was added into summary as presumably the words of the title may help with classifying genres thus combined into summary. Now with the titles added to the summaries, tokenization was run for each summary to lowercase all words, remove punctuation, and remove stop words. CountVectorizer from Scikit-Learn was used to tokenize the summaries and used to remove the stop words.

Final Dataset

The final dataset that was used, the number of books was reduced from 16,559 books to 6,403 books. There are three features total. “author” which is a one hot encoding vector for each book. “summary” is the combined title and summary of each book and tokenized. “genre” is the feature to classify and is the set of 5 genres (Children’s literature, Fantasy, Fiction, Mystery, Science Fiction) where each book can have more than 1 genre. The whole dataset has a total of 1,464,839 sentences, 80,630 distinct word tokens, 80,320 distinct word tokens without stop words, and 2,055 distinct authors.

Project Methodology

Method

To run the genre classification task using the final dataset, a random 80/20 split was chosen for the data, creating a fixed training data and testing data for all models that are

being used. The number of books in the training set is 4,283 and the number of books in the testing set is 2,120. Then the following 3 NLP methods below were used to represent the summaries of the books.

BOW

Using the tokenized summaries, it was transformed into a bag of words (BOW) with word counts for each summary. Each summary contained a word vector with the size of the vocabulary of the whole dataset with entries corresponding to word counts for the words in that particular summary. Then the machine learning models Logistic Regression and Random Forest, all from the Scikit-learn library were run with BOW to see which model would perform better. Since the project was to perform multilabel classification for each of the models OneVsRestClassifier was applied which essentially classified each genre separately one at a time and in the end combined all the classification predictions into one vector.

GLoVe

To see if there was an improvement in classification, BOW was replaced for word embeddings. In this case GLoVe was used (https://nlp.stanford.edu/projects/glove/). The word embeddings vectors were dimension 50, were pretrained, and the words were collected from Wikipedia. To obtain the word embedding vectors for the summaries,for each summary, the average of word embedding vectors was computed. The average embedding vector corresponds to the sum of each word embedding vector for each word in the summary divided by the total word count of the summary. In the end each summary is transformed into a word embedding vector of dimension 50 representing the average of all the word's embedding vectors. Then machine learning models Logistic Regression and Support Vector Machine (SVM) using Support Vector Classifier (SVC), were used with OneVsRestClassifier.

BERT

The final method that was used was BERT. Using pre-trained BERT, sentence vectors were used for the summaries. For each summary, each sentence in the summary was concatenated to make one large sentence which could then be used to make a sentence vector using BERT. Then, on the BERT summary sentence vectors, after slicing the portion that we wanted to declare as our features (a 2d tensor sliced from the original 3d tensor), Logistic Regression was run on said features for classification.

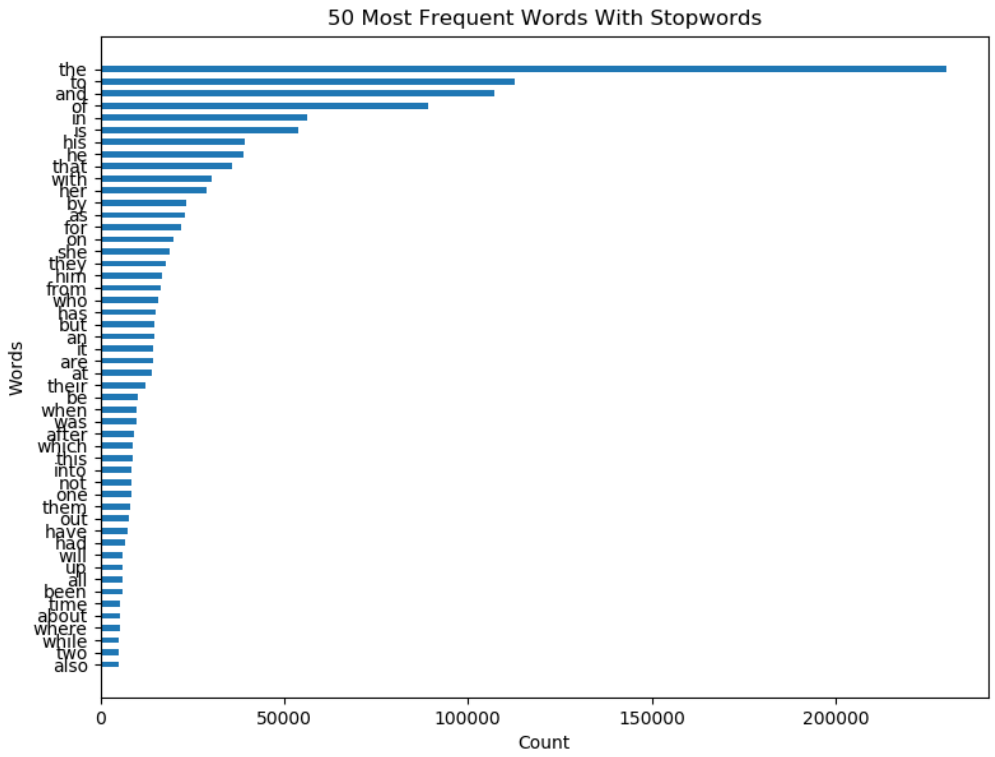
Classification Report

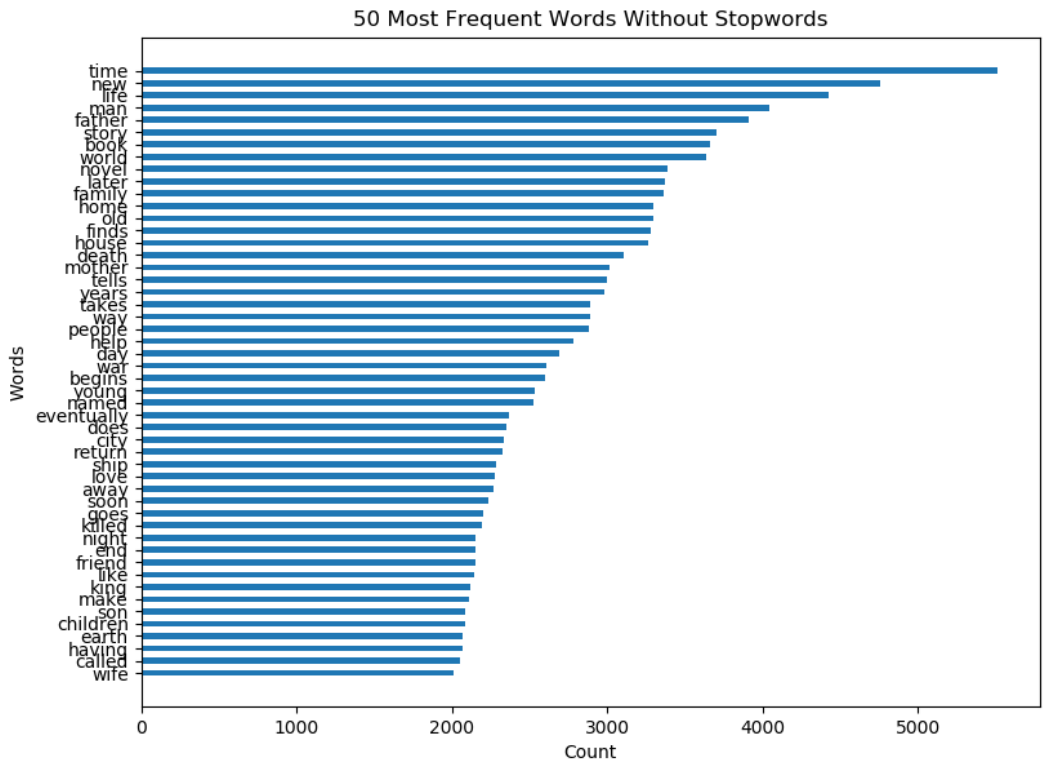
When running all the different machine learning models and techniques, the training data was fitted and then the models classified the genres of the testing data. Then a classification report was computed comparing the classification predictions and the actual labels to see the precision, recall, and f1 scores of each of the genre labels. Using the classification report was where the comparison of each model was analyzed.

Results

Baseline Method

Our baseline algorithm involved tokenizing each book summary text from our dataset so that we had a breakdown of the various words in the text. At the lowest level, we could use the results of tokenizing the text to determine which words were most commonly found in the text. We did the analysis of most common words found in the text using both stop words as well as without stop words where stop words refer to the words which do not add much meaning to a sentence such as the, a, etc. Below are two graphs depicting our results.





Experiment Descriptions based on Models

1. Logistic regression with BOW

We decided to start off with Logistic Regression and see how the model would perform with our dataset and chosen labels. We also compared the results when keeping and removing a hot vector for author names, however we saw no change in the results. We kept the vector as we plan on testing to see if word embedding has a positive impact on the results. One major concern that we thought after looking through the results is that some predictions have no labels assigned to them, which means that our Logistic Regression model was unable to score some summaries, but realized that is the nature of OneVsRestClassifier and multilabel classification since it classifies a genre at a time.

2. Random Forest

We used Random Forest with the same BOW representation of summaries. After trying multiple different values for the number of estimators, we came to the conclusion that 100 (the default value) gave a good balance between accuracy and speed. We also utilized the random state to create a globally optimal decision tree, thus optimizing our

results. We decided to let the algorithm run on all the potential leaf nodes and maximum depth of the trees in order to get a potentially more accurate model.

3. Logistic Regression with Word Embeddings (GloVe)

We use GloVe on our tokenized book summaries which are tokenized using a tokenizer from sci-kit learn and then the embedding of all words in a summary are averaged and hence, we obtain a 1D vector of features which correspond to each summary. This feature vector is then passed into our logistic regression model which is used to finish classification and calculate precision, recall and f1-scores of each genre.

4. SVM with Word Embeddings (GloVe)

We then moved on to using Support Vector Machines. SVM is based on Kernel methods and in this case we are using Support Vector Classifier (SVC). We used GridSearchCV which is a model selection tool in Scikit-learn which uses cross validation and allows us to find the best parameters for the model. GridSearchCV was run on the training data using 5-fold cross validation and ran multiple permutations of different parameters to find the best set. From the gridsearch the best parameters for SVC has kernel = rbf (Radial basis Function), C = 0.10, gamma = 0.1. These parameters were then used for the final SVM model to get precision, recall, and f1-scores for each genre .

5. Logistic Regression with Word Embeddings (BERT)

We used a pre-trained BERT model and tokenizer in order to carry out our classification. From the summaries in our dataset, we first concatenated all sentences of summary in order to make one large run on “sentence” and then cropped this new sentence if the number of tokens of the summary went above 512 tokens. From here, we can tokenize the summaries which turns every sentence into a list of ids. Since the dataset for book summaries is currently a dataframe, before BERT can process this as input, we make all the vectors the same size by adding padding to any shorter summaries by adding in the token id 0 using basic python array and string manipulation. After adding the padding, we now have a tensor/matrix which can be passed to BERT. For this BERT classification we are mostly interested in BERT’s output for the [CLS] token, hence we select that slice of the cube to use, so features will be a 2d numpy array containing the sentence embeddings of all the summaries in our dataset. We then pass features to our logistic regression model in order to finish our classification and obtain precision, recall and f1-scores.

Statistics based on Models

1. Logistic regression with BOW

Genres	Precision	Recall	F1-Score
Children Lit.	0.67	0.53	0.59
Fantasy	0.79	0.62	0.70
Fiction	0.72	0.74	0.73
Mystery	0.79	0.67	0.72
SciFi	0.84	0.74	0.79

2. Random Forest

Genres	Precision	Recall	F1-Score
Children Lit.	0.99	0.30	0.27
Fantasy	0.97	0.22	0.36
Fiction	0.67	0.88	0.69
Mystery	0.95	0.31	0.46
SciFi	0.96	0.50	0.59

3. Logistic Regression with Word Embeddings (GloVe)

Genres	Precision	Recall	F1-Score
Children Lit.	0.53	0.54	0.55
Fantasy	0.60	0.68	0.64
Fiction	0.63	0.69	0.66
Mystery	0.69	0.71	0.70
SciFi	0.67	0.69	0.68

4. SVM with Word Embeddings (GloVe)

Genres	Precision	Recall	F1-Score
Children Lit.	0.81	0.57	0.67
Fantasy	0.81	0.71	0.76
Fiction	0.74	0.87	0.80
Mystery	0.87	0.74	0.80
SciFi	0.89	0.79	0.84

5. Logistic Regression with Word Embeddings (BERT)

Genres	Precision	Recall	F1-Score
Children Lit.	0.82	0.55	0.66
Fantasy	0.84	0.69	0.76
Fiction	0.76	0.84	0.80
Mystery	0.89	0.71	0.79
SciFi	0.90	0.77	0.83

Final F1-Score Statistics

Genres	Children Lit	Fantasy	Fiction	Mystery	SciFi
LogReg with BOW	0.59	0.70	0.73	0.72	0.79
Random Forest	0.27	0.36	0.69	0.46	0.59
LogReg with GloVe	0.55	0.64	0.67	0.70	0.68
SVM with GloVe	0.67	0.76	0.80	0.80	0.84

LogReg with BERT	**0.66**	**0.76**	**0.80**	**0.79**	**0.83**

Analysis of Results

- SVM with GloVe word embeddings has the highest f1-scores and random forest has the lowest f1-scores. Also all the logistic regression models have lower f1-scores compared to SVM with GloVe word embeddings model, but logistic regression with GloVe word embeddings has the lowest f1-scores amongst all the logistic regression models. Comparing both SVM and Logistic regression, most probably SVM has better f1-scores since it attempts to "find the best margin (distance between the line and the support vector) that separates the classes" which ultimately leads to the reducing of the risk for errors on the data, whereas logistic regression does not take this approach and hence the risk for error on the data may be more. Also, both the precision and recall for the SVM with GloVe embeddings model are quite high when compared to the Logistic regression with GloVe model where precision and recall are quite low, hence leading to the f1-score of SVM with GloVe to be quite high and the f1-score of Logistic regression with GloVe to be quite low.

- Logistic regression with GloVe word embeddings has lower f1-scores when compared to Logistic regression with BOW. After reading about GloVe we discovered a lot of different biases or stereotypes which might be affecting the word associations being made. Interestingly, for example, male names are more associated with mathematics, science, and mystery than female names, whereas female names are more associated with the arts. So genres such as science fiction with science based words could have been more heavily biased towards associations with male dominance which may have not been the correct association hence leading to lower precision scores for Logistic Regression with GloVe compared to logistic regression with BOW.

- Logistic regression with BERT word embeddings has the highest f1-score within all logistic regression models whereas logistic regression with GloVe has lowest f1-scores. Comparing BERT and GloVe we notice that GloVe word embeddings are context independent which means that they output only one vector or embedding for each word which combines all different senses of the word into one vector. BERT on the other hand generates different word embeddings for a word that captures the context of a word which is its position in a sentence. For example, if we had a sentence such as, "He used his cell phone from his prison cell," BERT would make different vectors for the two vectors for the word 'cell'. The first vector would be closer to words such as 'iphone, samsung, etc.' whereas the second vector would be closer to words like 'criminal, crime, jail,

etc.’ This would probably help BERT in getting better recall and precision scores leading to a higher f1-score. Also, during tokenization BERT takes into account sub-words which GloVe does not, which might also in the long run help increase f1-scores.

Discussion and Future Work

After testing with various models and variations of those models, we found that it was indeed possible to classify books based on their summaries. With this knowledge, we believe that there is a lot of room to improve our models and utilize other ML algorithms to further improve the accuracy of our classifier. More specifically, utilizing deep learning and neural networks to create a more effective model could make a huge impact on the accuracy of our model and allow us to make more accurate predictions.

One assumption we made with this project was that we had a dataset that was pre annotated and had complete summaries. In a real world application of this project, it would be incredibly useful to utilize more effective methods of classification, such as neural networks, to be able to continue to make accurate predictions with less complete summaries. If we were to continue working on this project, we would also want to be able to expand our project’s scope to encompass all types of media and literature. Because our project is focused on classifying genres, it would be interesting to see if our current methods for classification would be more successful than other forms of media, such as news or movies, as opposed to books.

All in all, this project ended up answering our initial research query, letting us know that it is feasible to classify books using their summaries. Furthermore, this project allowed us to delve deeper into various machine learning algorithms and implement our own models for text classification. With rigorous trial and error, we were able to fine tune our models to produce positive results that could potentially have an impact in a real world scenario.