Book Genre Classification using RoBERTa with different Classifiers

Text Mining Project
(732A81)

Author:
Abhijeet Anand
(abhan872)

Abstract

Book genre classification is the task of predicting the genre of a given book based on its content. In this project, we explore the use of the RoBERTa language model and several classifiers, including XG Boost Classifier, Multi-layer perceptron (MLP) classifier, support vector classifier, and others, for book genre classification. We use a dataset of books in 6 different genres and evaluate the performance of our approach using various metrics. Our experimental results show that the use of RoBERTa and the classifiers we tested achieved an overall accuracy of 70% with Support vector Classifier. This demonstrates the effectiveness of using RoBERTa and these classifiers for book genre classification and highlights the potential of these approaches to improve the organization and recommendation of books in various settings.

1 Introduction

Book genre classification is the task of predicting the genre of a given book based on its content. This can be useful for a variety of applications, such as recommending books to readers or organizing books in a library or online bookstore. In recent years, machine learning techniques have been widely used to classify books into various genres.

One such approach is to use a language model like RoBERTa (Liu et al., 2019) to extract features from the title and summary of a book and feed those features into a classifier, such as a support vector machine (SVM), a random forest, Multi-Layer Perceptron and others. These classifiers can then learn to predict the genre of a book based on the extracted features.

Overall, the use of RoBERTa and other machine learning techniques has shown great promise in the task of book genre classification, and these approaches have the potential to improve the organization and recommendation of books in various settings.

2 Theory

2.1 RoBERTa

RoBERTa (Robustly Optimized BERT Approach) is a transformer-based language model developed by researchers at Facebook AI in 2019 (Liu et al., 2019). It is a variant of the popular BERT model (Bidirectional Encoder Representations from Transformers) and is designed to improve upon it by addressing some of its limitations.

One of the key differences between RoBERTa and BERT is that RoBERTa was trained on a much larger dataset, which includes a combination of both books and web pages (Liu et al., 2019). This allows RoBERTa to better capture the nuances of natural language and improve its performance on a variety of tasks.

RoBERTa also differs from BERT in its training process. It uses a dynamic masking strategy that allows the model to predict a masked token based on the context of the surrounding tokens, rather than the entire sequence (Liu et al., 2019). This approach has been found to be more effective at improving the model's generalization abilities (Liu et al., 2019).

RoBERTa has achieved state-of-the-art results on a number of benchmarks, including the GLUE and SQuAD datasets (Liu et al., 2019). It has also been used to improve performance on a variety of natural language processing tasks, such as machine translation, text classification, and question answering (Liu et al., 2019).

Overall, RoBERTa is a powerful language model that has shown promise in a wide range of NLP tasks.

2.2 XG Boost Classifier

XG Boost (eXtreme Gradient Boosting) is a decision tree-based ensemble machine learning algorithm that is used for classification and regression tasks (Chen and Guestrin, 2016). It is an efficient and accurate algorithm that has been widely used in industry and has achieved state-of-the-art results on many benchmarks (Chen and Guestrin, 2016).

One of the key features of XG Boost is its ability to handle large and imbalanced datasets, as well as its ability to handle missing values (Chen and Guestrin, 2016). It also has a number of tuning parameters that allow the user to customize the behavior of the model, such as the learning rate and the maximum depth of the trees (Chen and Guestrin, 2016).

Overall, XG Boost is a powerful and popular machine learning algorithm that is well-suited for a variety of tasks, including classification and regression.

2.3 Multi-Layer Perceptron Classifier

A multi-layer perceptron (MLP) is a type of artificial neural network that is composed of multiple layers of interconnected nodes, or "neurons." The MLP is a feedforward network, which means that information flows through the network in only one direction, from input to output.

MLPs are often used for classification tasks, where the goal is to predict a discrete class label for a given input. The input layer receives the input data, and each subsequent layer processes and transforms the data using a set of weights and biases. The output layer produces the final prediction based on the transformed data.

MLPs have been widely used in various applications, including image classification, natural language processing, and speech recognition (Bengio et al., 1994). They are generally considered to be powerful models for learning non-linear relationships in data.

2.4 Random Forest

Random forest is a machine learning algorithm that constructs a set of decision trees and combines their predictions to classify or predict a target variable (Breiman, 2001). It is a popular method for both classification and regression tasks and has been used in a wide range of applications, including image and speech recognition, natural language processing, and bioinformatics ((Breiman, 2001); (Liaw and Wiener, 2001)).

Random forest works by training multiple decision trees on different subsets of the training data and using the majority vote of the trees to make a final prediction (Breiman, 2001). This process helps to reduce overfitting and improve the generalization performance of the model.

Random forest has several advantages, including its ability to handle large datasets, high dimensionality, and missing values ((Breiman, 2001); (Liaw and Wiener, 2001). It is also relatively easy to implement and can produce good results with minimal parameter tuning (Breiman, 2001).

2.5 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a widely-used optimization algorithm for training machine learning models, particularly in deep learning. It is a variant of gradient descent where the model's parameters are updated incrementally with each training example, rather than using the entire dataset to compute the gradients. This makes SGD more computationally efficient, especially when working with large datasets. The original paper on SGD was published in 1951 by Robbins and Monro (Robbins and Monro, 1951) and it has been widely used and improved upon since then.

2.6 Adaboost

AdaBoost (Adaptive Boosting) is a popular ensemble learning algorithm that is used to improve the performance of weak classifiers. It is a meta-algorithm that combines multiple weak learners to form a stronger classifier. The algorithm was first introduced by Yoav Freund and Robert Schapire in their 1996 paper "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting" (Freund and Schapire, 1997). Adaboost is an iterative algorithm that adjusts the weights of the training examples at each iteration based on the error of the previous iteration's classifier. It continues to add classifiers until a certain accuracy or a maximum number of classifiers is reached. Adaboost is commonly used in a variety of applications such as image and speech recognition, computer vision and bioinformatics.

2.7 Support Vector Classifier

Support Vector Classifiers (SVCs) are a type of supervised machine learning algorithm that are used for classification tasks. The idea behind SVCs is to find the best boundary (or "hyperplane") that separates the different classes in the feature space. The boundary is chosen such that it maximizes the margin, which is the distance between the boundary and the closest data points from each class. The data points that are closest to the boundary are called support vectors. The algorithm was first introduced by Vladimir Vapnik in his 1995 paper "The Nature of Statistical Learning Theory" (Vapnik, 1995). SVCs have been widely used in various fields such as text and image classification, bioinformatics, and financial analysis. SVCs are commonly used with a kernel trick, which allows them to handle non-linearly separable data, by mapping the data into a higher dimensional space.

2.8 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure that is widely used in text mining and natural language processing tasks as a way to represent the importance of a word in a document. The measure is composed of two parts: term frequency (TF) and inverse document frequency (IDF). TF measures the number of times a term (word) appears in a document, while IDF measures the rarity of the term in the entire corpus of documents. The idea behind using TF-IDF is that a term that appears frequently in a document, but not in many documents throughout the corpus, is likely to be more informative and relevant to the topic of the document than a term that appears frequently across all documents. The TF-IDF measure was first proposed by Karen Spärck Jones in her 1972 paper "A statistical interpretation of term specificity and its application in retrieval" (Sparck Jones, 1972). Since then it has been widely used in various text mining and information retrieval tasks such as document retrieval, text classification, and feature extraction.

3 Data

The data used in this project is taken from Kaggle. The data initially has 4 columns namely index, title, genre and summary with 4657 rows. The data has 10 genres to be classified namely thriller, fantasy, science, history, horror, crime, romance, psychology, sports, and, travel. However, it is observed that the number of books in these genres vary a lot as shown in Fig. 1. It is also observed that the word count in the summary varies a lot well with as minimum as 1 word and maximum as 5663 words shown in Fig 2. Duplicate books with same title or summary but classified in different genres also observed.

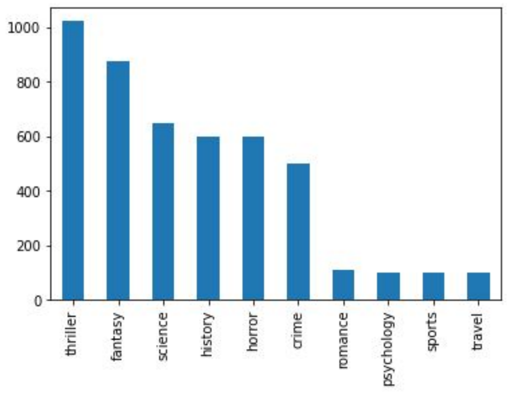


Figure 1: Number of books per genre

Thus, during data cleaning, column index, genres with less than 150 books, duplicate entries, and books having summary less than 100 words or more than 1500 words are removed. Finally a dataset with 3102 rows and 6 genres (fantasy, science, crime, history, horror, and, thriller) is exported to be pre-processed.

4 Methods

From the previous exported data, there were 3102 rows with two interesting columns tilte and summary that will be used for modelling. But before we proceed for modelling, we need to preprocess the data in order to keep only important words/keys in the correct format.

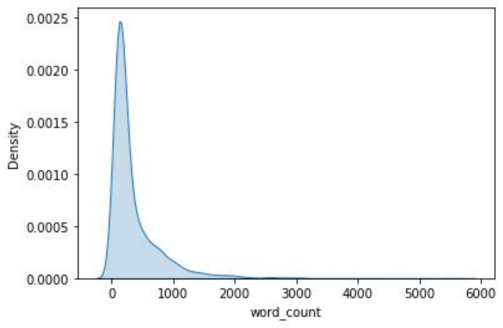


Figure 2: Density plot based on word count

4.1 Basic cleaning

In the basic cleaning, we are processing title and summary columns. At first all the data present in these columns were made lower case followed by removing punctuations, stopwords. Further, words with length less than 3 were also eliminated. Finally, single characters, HTML tags, new line characters and multiple spaces were also removed. Most of these tasks were done with the help of regex library.

4.2 Data Preprocess

In this section, basic cleaned data is taken to further process with advance processings like tokenization and lemmatization. At first, the columns title and summary are combined to make a new column 'title_summary'. The above mentioned processing is done on this column. Finally this data is exported to prepare document representation.

4.3 Document Representation

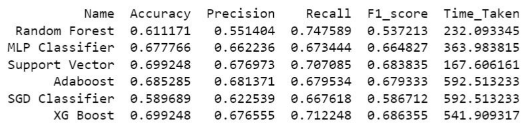
This is one of the most important step in any NLP task, where the text data is converted to number representation. There are several techniques to do this task including TF-IDF and many others. In this project, RoBERTa (Robustly Optimized BERT Approach) a state-of-art mechanism is used. To get this representation, the column 'title_summary' preprocessed data is used as a list. With the help of tensorflow_hub this data is passed through processor to generate the input sequence and fed to encoder to get the output. The 'sequence_output' from the outputs has the shape of ([3102, 128, 512]). This is iterated over the range of length of data to get the final document representation and is exported.

4.4 Modelling

The data is divided into train and test with the ration of 70/30. In this project, we are using several classifiers to do the classification namely 'Random Forest', 'Multi Layer Perceptron', 'Support Vector', 'Adaboost', 'Stochastic Gradient Descent' and 'XG Boost'. All the models fairly performs well. All the classifiers were hypertuned with the help of Grid Search Methon. The results from all the classifiers along with time taken is stored in a dataframe and will be discussed later in this report.

5 Results

With the hypertuned parameters all the classifiers were run and the results were collected. As shown in the Fig. 3 all the classifiers produce similar results. However, XG Boost Classifier and Support Vector Classifier tops the table with 69.9% of accuracy each.



```
             Name  Accuracy  Precision    Recall  F1_score  Time_Taken
    Random Forest  0.611171   0.551404  0.747589  0.537213  232.093345
   MLP Classifier  0.677766   0.662236  0.673444  0.664827  363.983815
   Support Vector  0.699248   0.676973  0.707085  0.683835  167.606161
         Adaboost  0.685285   0.681371  0.679534  0.679333  592.513233
   SGD Classifier  0.589689   0.622539  0.667618  0.586712  592.513233
         XG Boost  0.699248   0.676555  0.712248  0.686355  541.909317
```

Figure 3: Results from all classifiers

The line chart for accuracy with respect to different classifiers can be seen in Fig. 5. Here, we clearly see that all the classifiers produce accuracy in the range of 58% to 70%. With XG Boost and Support having the highest accuracy of 69.9% each, we also observe that SGD Classifier has the least accuracy of around 58%.

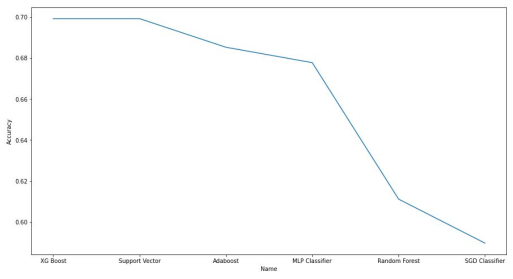


Figure 4: Accuracy from all classifiers

The entire modelling was done over the CPU. So, the time consumption is more than usual. While the accuracy were very close, the time taken by each classifier varied a bit. While looking at the Fig. 5. We see that Adaboost and SGD Classifier consumes most of time while Support vector consumes the least.

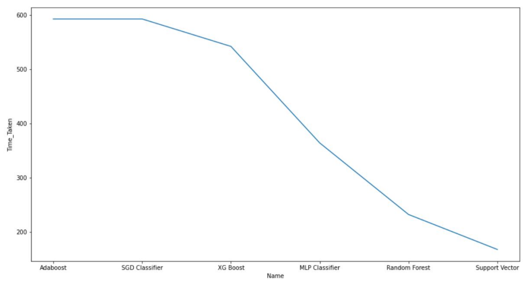


Figure 5: Time taken by classifiers

The confusion matrix is a useful tool for evaluating the performance of a classification model, and it can be easily computed using a variety of software tools and libraries. Looking at the confusion matrix from XG Boost Classifier shown in Fig. 6, we clearly see that genre 'Fantasy' is wrongly predicted as Thriller in vast, while other genres are quite well predicted.

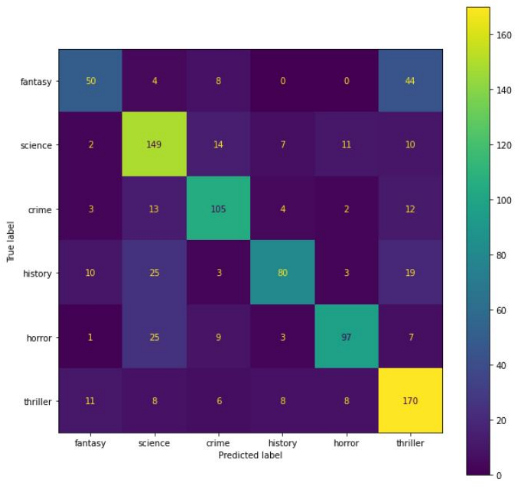


Figure 6: Confusion Matrix

Discussion

The results generated from this project seems good with around 70% accuracy. However, it creates a curiosity that why are the genres fantasy and thriller are misclassified in vast? Upon investigation for both the classes through word cloud shown in Fig. 7 and Fig. 8, we observe that both the genres are very closely related as we see from the word clouds that the most common words like 'find, life, time, world' are very common in both genres. Thus, the misclassification between these two genres are in vast.

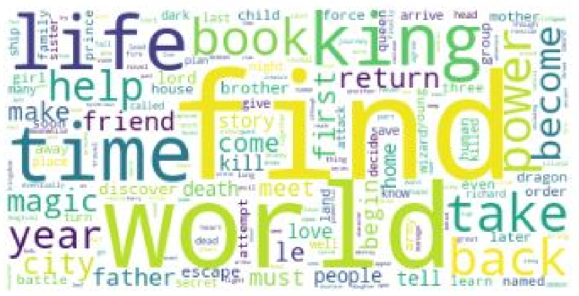


Figure 7: Word Cloud for genre Fantasy

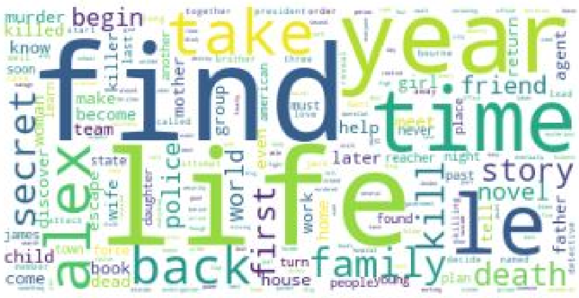
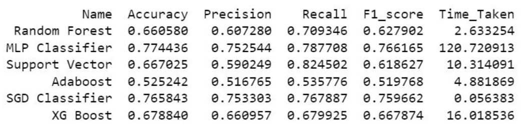


Figure 8: Word Cloud for genre Thriller

Looking back at the preprocessing, it was also noticed that the lemmatization was not very good as the past forms of the words were not taken back to its root version. For e.g. words like ‘drowned, returned, persuaded‘ were kept as it was, which could have been better, if it was lemmatized to ‘drown, return, and persuade‘ respectively.

On the other side, we observe the best accuracy of around 70% with RoBERTa, which is a very advanced way of document representation. To end the curiosity, TFIDF was also tried on the same preprocessed data to compare if it is worth to go for this advanced technique. All the classifiers were used here to classify, but none of their parameters were tuned. Basically, a simple document representation method were used along with all the classifiers in their most basic form. The results are shown in the Fig. 9



```
             Name  Accuracy  Precision    Recall  F1_score  Time_Taken
    Random Forest  0.660580   0.607280  0.709346  0.627902    2.633254
   MLP Classifier  0.774436   0.752544  0.787708  0.766165  120.720913
   Support Vector  0.667025   0.590249  0.824502  0.618627   10.314091
         Adaboost  0.525242   0.516765  0.535776  0.519768    4.881869
   SGD Classifier  0.765843   0.753303  0.767887  0.759662    0.056383
         XG Boost  0.678840   0.660957  0.679925  0.667874   16.018536
```

Figure 9: Result from TFIDF

we can notice that MLP Classifier and SGD Classifier gives the best accuracy of around 77%. While other classifier such as Support Vector and XGBoost are also very close with around 67% of accuracy.

It is also our point of interest that though both MLP and SGD give us the best accuracy, but time taken by SGD is extremely great which is less than 1 sec, while MLP takes around 120 sec.

Conclusion

To conclude, the classifiers along with RoBERTa and TFIDF document representations work pretty well with this dataset. The data initilally was highly imbalanced. In order to balance and predict well, the data was reduced a little. With bigger dataset and more balanced dataset along with suggested preprocessing and combining differerent document representations together, the results can be improved to a great extent.

Several different classifiers were used with the generated document representation with both RoBERTa and TFIDF. With RoBERTa, it was noticed that the accuracy with XG Boost and Support vector Classifiers were the best, however the time taken by Support Vector Classifier was great.

With TFIDF as document representation, it was noticed that the accuracy with MLP and SGD Classifiers were the best, however the time taken by SGD Classifier was extra-ordinary.

The conclusion of the comparison is sometimes simpler techniques give better results as we saw TFIDF gave us better results than RoBERTa. However, It would be very interesting to see how well will these classifiers work with bigger dataset and better preprocessings.

References

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Andy Liaw and Matthew Wiener. 2001. Classification and regression by randomforest. *Forest*, 23.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. Cite arxiv:1907.11692.

Herbert Robbins and Sutton Monro. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.