Book Genre Prediction from Meta Data

Yaobang Deng
yad025@ucsd.edu
A13712124

Yikuan Xia
yix146@ucsd.edu
A14009432

Yuqi Kang
yuk033@ucsd.edu
A92048017

Abstract

How to predict the genre of a book without looking through the whole book? People can judge a book by reading its abstract, title or description, can a computer also learn from these features? We propose designing different feature vectors and using different classifiers to predict the genre of a book. We find out that the title, description, rating count and similar book contribute greatly to our predictive task. We mainly focus on designing efficient feature vectors and using Support Vector Machine, Logistic Regression as well as Multiple Layer Perceptron to approach our predictive task. In addition, we address and attempt to predict a harder multiclass and multilabel problem.

1 Introduction

Goodreads is a "social cataloging" website that allows individuals to freely search its database of books, annotations and reviews. Users can add books to their personal bookshelves, rate and review books and get suggestions for future reading choices based on their reviews of previous books [1]. The genre of a book is often a determinate factor for people to decide whether to read it or not. In the case where a person encounters a brand new book, with just the information of its name and description, it would be time-consuming to self-determine the book's genre by reading its description, and even the first several pages carefully. Therefore, a predictive model to classify a book's genre based on its other information seems to be necessary to save people's time and help them to make correct decision. We try multiple predictive models such as linear classifier and simple neural network, and has a performance of correct prediction as high as 71.66% on multiclass single label prediction problem. For the multiclass and multilabel problem, we try `RandomForestClassifier` and `MLPClassifier`, and obtain a considerable precision, recall and accuracy score.

2 Dataset

The dataset we use is collected from the website of Goodreads at late 2017[2]. It contains the information of books that are in the public shelves of each user. Such information includes its average rating, number of ratings and reviews, book title and description, the shelves that this book is in as well as the genres of the book, etc. All user id and review id are anonymized in this dataset. We obtain this dataset from Professor's website and only use it for the purpose of this assignment. Since we want to perform the task of genre prediction, we apply the complete dataset instead a subset of one genre. Major information of books in Goodreads public shelves is from the *goodreads_books.json* file. The following is some key fields that might be relevant to our purpose for us to further investigate:

- text_review_count: the number of text reviews of this book
- popular_shelves: a list of user-defined shelves that the book belongs to
- average_rating: the average rating of the book

- similar_books: a list of books that users who like the current book also like
- description: a text description of the book's content
- book_id: the identity id of the book
- word_id: the identity id of a list of same book in different editions
- series: a series id if the book is a part of a book series
- num_pages: the number of pages of the book
- num_ratings: the number of total rating
- author: the author id of the author who write the book
- publisher: the publisher of this book

Additional information about the coarse genre of a book is obtained form the file *goodreads_book_genres_initial.json* with the following field paired with according book_id:

- genres: a list of genres that the book is in, with a count number attached to each genre

Information about the book title is achieved from the file *goodreads_book_works.json* with the field associated with the according work_id:

- original_title: the title of the book
- best_book_id: the book id of a specific edition which is regarded as the best representative

2.1 Data Summary and Preprocessing

The complete dataset has a collection of 2,360,655 books, which includes 400,390 book series and 1,521,962 works. All the books are categorized into 10 genres.
To combine the three files we choose into one, we first add "cover", which is the title of book and "genre", which is the genre of the book as additional fields of the dataset extracted from *goodreads_books.json* file. The "cover" field has the content of "original_title" with matching work_id; the "genre" field has the content of "genres" with matching book_id.
In the further exploration, we noticed that for most books that share a same work_id, they also share a similar content of description and the same value of average rating. Since we plan to apply tfidf in model constructing, to avoid double-counting risk, from a list of books with the same work_id, we only choose one book whose book_id is the "best_book_id".
We also find that for a series of book, they always share the same average rating and title. Besides, only one book in such series has nonempty book description. Therefore, for each series of books, we only keep one book which has the text description feature.
As the document states that the book genre is generated from the popular_shelves, we removed this field in our dataset.
Even the dataset after preprocessing contains the information of around 1 million books. Operating such large dataset is extremely time-consuming. To save time, we randomly shuffle the whole dataset and achieve the first 100,000 books as our final dataset.

2.2 Data Analysis

The data analysis is based the 100,000 datapoint in the final dataset. For an overview, we generate the overall distribution of genres in this dataset (Figure 1). Since there might be multiple genres for a single book, for a simple demonstration, we choose the genre with the largest count number as the genre of the book.

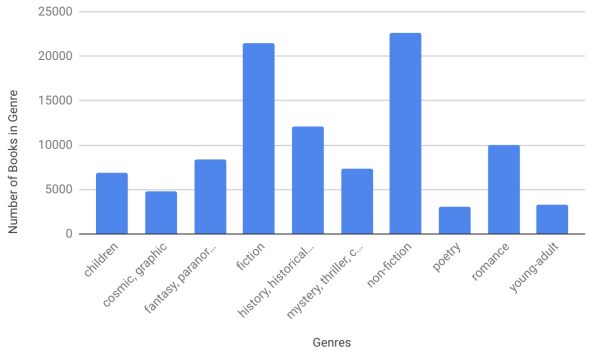

Figure 1: Genre Distribution of Dataset

This distribution is similar to the genre distribution of the complete dataset, reconfirming the reliability of the dataset we use. With the general picture in mind, we then explore the variation in features across

different genres. We first examine the relationship between average rating and genres (Figure 2). From this figure, we could not observe significant difference of average ratings in different genre.

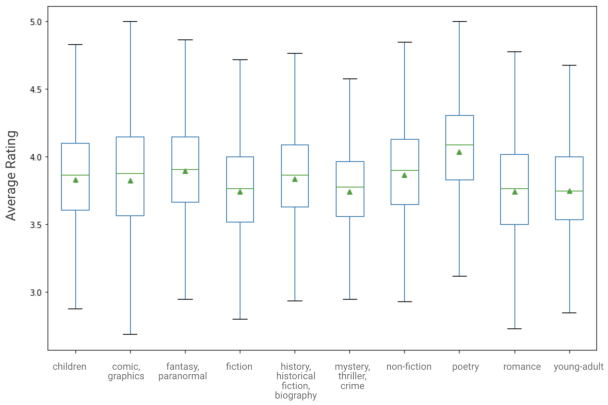

Figure 2: Average Rating v/s Genre

We then check the relationship between average page number and genres (Figure 3). Though there is visible difference of page number variation in some genres, such as "children", "comic, graphics" and "poetry". However, a considerable standard deviation value is also noticed.(showed in Figure 3)

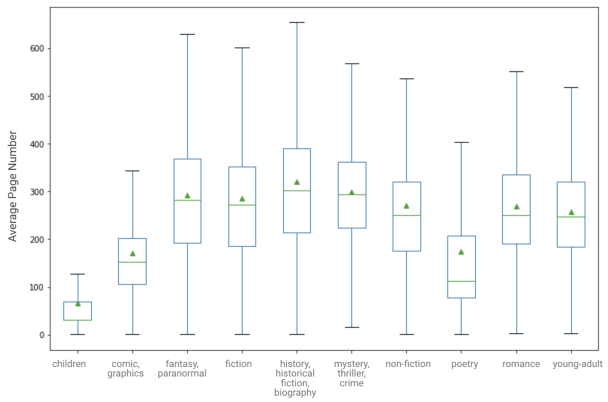

Figure 3: Average Page Number v/s Genre

We further explore the average number of text reviews (Figure 4) and ratings (Figure 5) in each genres. Interestingly, through in different scale, these two graphs shares a similar trend, where the genre "young_adult" has the most average review count and ratings, and "poetry" has the least. However, even though not shown in figure, there are also a noticeable standard deviation in two feature, possibly due to there is always some extremely popular books with a huge number of reviews and ratings in each genre.

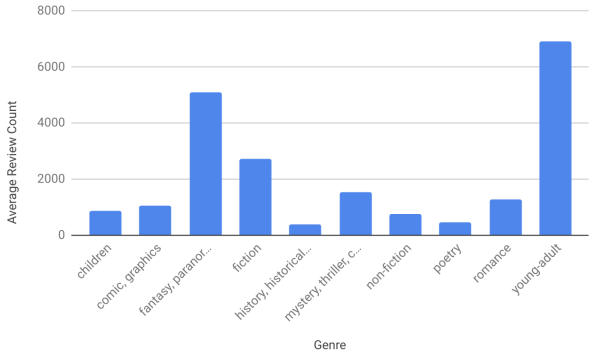

Figure 4: Average Review Count v/s Genre

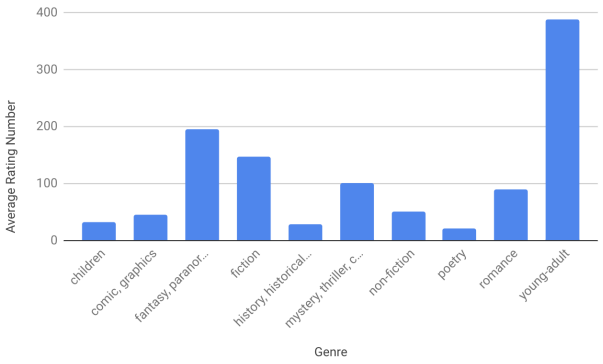

Figure 5: Average Rating Count v/s Genre

We also examine other interesting features like author and publisher to see if certain author or publisher tend to write/publish certain genres of book more compared to others. We found no obvious relationship between genre and publisher. Also, after filtering of work_id and series, the relationship between genre and author is weakened, and it is further weakened due to the fact that in the dataset, most author only write a limit number of books. No figure illustration for these two fields is presented here, for the huge amount of authors and publishers.
Based on the above data analysis, besides the text features like description and cover, we choose to also

apply the average_rating, num_pages (pages numbers), text_review_count (number of reviews) and ratings_count (number of ratings) as features when performing the genre prediction task.

3 Predictive Task

Our task here is, given the meta data of a book, such as description, cover, rating, review counts, etc., we use a model to predict the genre/genres of a book. We randomly choose 100,000 data from the big Goodreads dataset to perform our task. We split the chosen data into training data(50,000), validation data(20,000) and test data(30,000). As mentioned in section 2, this is a discrete multiclass classification problem. To compare different models, we split the task into two parts. First, we assume that each book can be assigned to only one genre. Second, we treat the task as multilabel classification problem.

i Multiclass Model Evaluation

We assume that each book can only be assigned to one genre, and there are 10 different genres in total, so this is a multiclass classification problem. We need classifier here instead of regressor or other models because the output labels are discrete and deterministic. First of all, we need to assign label(one genre) to each book. Feasible ways are, to randomly choose one genre from each book's genre list, or to pick the genre with the highest count in the list. What we did is the latter, to assign each book the genre with the highest count, i.e. the most popular genre. One possible baseline model is to find all similar books of a test book and assign the most frequent genre to it. The other possible models would be a machine learning classifier with designed feature vectors. When comparing and assessing the performance of our models, we compare the accuracy and training time of the model, i.e. how many data the classifier classifies correctly and how long it takes to train the model.

ii Multilabel Model Evaluation

For this part, we assume that the genre list is the genres that a book belongs to. So this is a multiclass and multilabel classification problem. We not only need a model that can sample an output from a multinomial distribution data, but we also need a model to yield multiple labels from the data. We use multilabel machine learning classifiers with designed feature vectors in this part. When it comes to multilabel prediction, our model yields much more negative predictions than positive predictions, so we concentrate more on the positive predictions. We use a few metrics from sklearn library [9]. The four main metrics are precision, recall, F1-score and accuracy [8].

$$Precision = \frac{tp}{tp+fp}$$

$$Recall = \frac{tp}{tp+fn}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

$$accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ predictions}$$

We need Precision to see how many correct positive predictions our model yields. We need Recall to see how many positive predictions are made over all positive labels, to see whether our model is naively predicting negative or not. F_1 score is an average evaluation between precision and recall. Accuracy is an overall evaluation for our model.

4 Models and Methods

In this section, we approach the task with multiple models from `sklearn` library [9]. We've tried a simple popularity-based or similarity-based model, multiple linear classification models, neural network classification model and different features. As mentioned in section 3, there are 10 categories in total, so all the models we use are for multi-class classification. Then, we also address multi-label classification at the end of this section.

4.1 Naive popularity-based model

Our first thought of how to solve this predictive task is to find clues in the "similar book" entry. Suppose we don't know genre of a book in test set, we can look into its similar book in training set(if any) and see what the dominant genre among them is.This book, we conjectured, has a good change of having

such dominant genre. Therefore, our naive model has the following steps to perform prediction: Firstly, iterate through each book in the similar book entry, find out the ones that are in the training set(if not such book exists, predict the genre that occurs most often in training set). Secondly, map those books to its genre and predict the genre that occurs mostly. If there is a tie, predict the one that occurs mostly among them in training set.

4.2 Classification models

After getting a bad accuracy at our Naive model, we try to use classifiers to fit some features from the data. Inspired by the fields in dataset and the paper "Classifica-tion of Books' Genres By Cover and Title" [4], we design a feature vector that contains the encoded description and cover of a book. The feature vector for each book looks like,

$$[description\ words \mid cover\ words] \tag{1}$$

The description and cover for a book are strings, so we use TfidfVectorizer to convert the strings to tfidf features. First collect the corpus of words in all data under the field 'description' as well as 'cover'. Then fit the corpus in the vectorizer to get numerical representation for each word. For each word in the corpus, its tfidf value is gathered based on the below equation[10].

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

$$tf(t,d) = frequency\ of\ t\ in\ d$$

$$idf(t,D) = \log \frac{N}{|d \in D : t \in d|}$$

Need to notice, there are many frequent words in the text like 'and', 'a', etc. So we decide to remove all the stop words in English. By doing so, we can decrease the dimensionality of our tfidf feature. After fit the corpus, we have numerical features for description and cover. For each book in the data, concatenate the description and cover and transform it to the tfidf feature vector. Then under this subsection, we can use this feature vector to do classifications. The evaluation metric we use for different classifiers is the accuracy, i.e. how many data the classifier classifies correctly.

i Support Vector Machine

We use the `LinearSVC` from sklearn.svm. With the library function and the feature vector described above, we train the data on our model, and choose the hyperparameters based on the performance of he validation data. The hyperparameter we attempt to experiment is the penalty parameter C of the error term. This parameter denotes how much we want to penalize for the misclassifications. We've experimented C = 0.01, 0.1, 1. As C goes larger, it takes longer to train the classifier. And the best accuracy we obtain on the validation data is 68.75%, when C = 0.1. Also, for the multicalss parameter in `LinearSVC`, we use 'OvA' metric, i.e. 'One class versus Other classes'.

ii Logistic Regression

We use the `LogisticRegression` from the library sklearn.linear_model. With the same designed feature vector above, we use the same metrics when choosing the best model hyperparameters. LogisticRegression is a linear classifier with a little non-linearity on the output, which may performs better on non-linear data. We've also tried different regularization parameters C = 0.01, 0.1, 1. And the best accuracy we can get on the validation data is 67.41%. Our data is multinomial, and we update the weights of our classifier based on cross-entropy-loss [6].

iii Multi-Layer Perceptron Classifier

We use the `MLPClassifier` from the library sklearn.neural_network. MLPClassifier is a simple neural network that has multiple layers with relu activation function and perceptron output on the last layer. We set up a MLPClassifier with two hidden layers, each of which contains 100 neurons and has relu as activation function. We use Adam optimizer and the adaptive learning rate to initialize our classifier, which is considered as an empirically better settings [7]. The model inherently uses a log loss function to optimize the weights of the classifier, and it uses One versus All method to classify multinomial data. The best accuracy we can obtain is 69.24%.

4.3 Experiment with different features

Apart from using different models, another way to improve the accuracy is to use more informative features. Our first experiment is to add the genres of similar book to the feature vector. In the naive model, we simply predict the dominant genre of similar books. Here, we creates a vector of size 10(the number of all distinct genres) and each entry is the count of how many times the corresponding genre occurs in similar books. We then normalize it by dividing the max value in the vector. This ensures that such vector is determined by the relative proportion of occurrence among genres instead of the absolute size.

The second additional feature we considered to be useful is the rating count. Our supposition is that some genres of books are more popular and some are not. So we simply use the rating count, which is a good representation of popularity, as the extra feature. However, this extra feature has much larger value than the original features since a book might have thousands of rating counts. It turned out that such highly imbalanced feature vector slowed down the SVM model significantly. To solve this problem, we performed normalization on rating count. Since the distribution for rating count is highly skewed(the range of rating count of low popularity books is much narrower than that of popular books), we apply log function on the rating count and then compute z-score. In this way we obtained a normalized feature with reletively small values.

We then append these two extra features to our original feature vector and perform prediction using SVM and logistic regressor. It turned out the the accuracy improved by roughly 3%

Other than these two extra features, we also tried features such as page number, review count and genres of what the author writes. They all turned out to be not very useful though.

4.4 Multilabel classification

It's too naive to consider that a book has only one genre. So this motivates us to predict multiple possible genres when given the meta data of a book. First of all, we need to encode the labels since the output of our classifier may be more than one. We propose a method similar to one-hot encoding: first find the number of genres n, then for each training book's label, design a vector with length n, set an index to be one if the book belongs to the genre at that index. For example, if we have a genre corpus

$$[children\ ,\ fantasy\ ,\ science]$$

and a book's genre list is

$$[fantasy\ ,\ science]$$

then its label is encoded as

$$[0\ 1\ 1]$$

For the feature vector of each book's meta data, we use the same designed vector as in section 4.3. For the multiclass and multilabel problem, we use two classifiers from the sklearn library, which inherently can be used for generating multilabel output. Different from the single label prediction, we use more metrics when evaluating the performance of our model as mentioned in section 3.

i Random Forest Classifier

We use `RandomForestClassifier` from sklearn. It's a tree-based classifier, which consists of a 'forest' of Decision Trees. It fits sub-samples of the dataset to multiple decision trees. By averaging the results, it can improve the prediction accuracy and prevent over-fitting [9]. We set the max depth of the decision trees to be 6, which is generally considered as a decent choice.

ii Multi-Layer Perceptron Classifier

Similar to the implementation in section 4.2, we also set up a MLPClassifier with two hidden layers, each of which has 100 nerons and relu as activation function. We still use Adam optimizer and adaptive learning rate. Moreover, we enable the classifier to early stop if the performance on validation set doesn't go up.

5 Literature Survey

The dataset we use here is from the paper "Item Recommendation on Monotonic Behavior Chains",

where the dataset is used for book recommendation modeling based on user's ratings, books read and books in their public shelves. Some similar dataset are included in the research work of following related literatures:

- Classification of Books' Genres By Cover and Title, 2015 [4]
- Judging a Book by its Cover, 2017 [5]
- Amazon Product Data (Subset: Book Reviews) [3]

We explore the other predictive model used in these literature and compare it with our own model.
The paper "Classification of Books' Genres By Cover and Title" [4] obtains its dataset used in this paper from Google Books API, which provides both the book title and the image of the book's cover. We only focus the process of title in this paper, for we lack the feature of cover image. In this paper, two different natural language processor (NLP) are used to extract features from books' title. A probabilistic soft-max classifier is then used to measure the possibility of the title belonging to each genres and perform the final classification. This paper shows accuracy around 60% when trying to predict the genre of books using merely title in NLP model. Our model achieves 69% accuracy, which is a significant improvement
This paper also suggests other approaches such as Naive Bayes, K-nearest neighbors, neural network and decision tree in genre prediction. Another literature "Judge a Book by its Cover" [5] also integrated the cover feature of a book into the features. The authors of this paper apply deep Convolutional Neural Network (CNN) in building the predictive model. In the field of text recognition and feature extraction, such CNN is consisted of eight layers. No performance data is provided in this paper, but the conclusion reveals that the text of a book's title has some correlation with the genre of the book.
From the information above, we could see a trend of the application of neural network and image recognition in the current book genre prediction models. However, by designing more representative feature, we achieve better performance using relatively simple model such as SVM and logistic regressor.

6 Results and Conclusions

We compare the accuracy and training time for each model. Also, we visualize the popularity of words in description that symbolizes a genre in order to prove that the encoded description feature is useful for predicting. Moreover, we compare different metrics for the multilabel part. Below are the glimpse of some of the results.

i Words Visualization

Figure 6: Fiction

Figure 7: romance

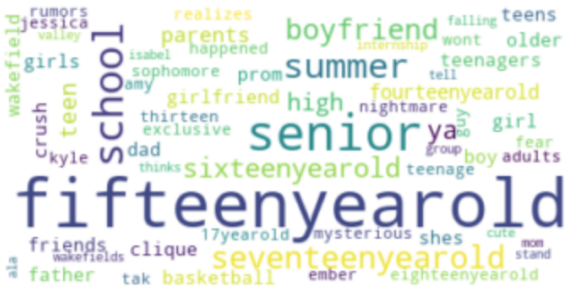

Figure 8: Young-adult

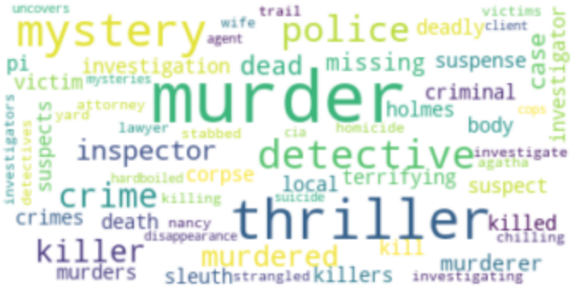

Figure 9: mystery, thriller, crime

As we can see from Figure 6 to Figure 9, the description words are highly related to the genre of a book. By using tfidf value for the words, each word has its own weight when classifying a genre from the other. For example, our LogisticRegression classifier uses description as one of the features, and each word in the description corpus has a weight. If the input is a book with description containing lots of romantic words such as "attraction", "sexy", etc., then the weight/swtich for those words will be "turned on", resulting in the output of "romance".

ii Multiclass Model

In this section we compare the performance of our Baseline model, LinearSVC model, LogisticRegression model and MLPClassifier model. Moreover, we compare their performances on two different feature vectors. Below are the results.

Table 1: Accuracy of models with feature 1
The feature vector here is encoded [*description* | *cover*]

Model	Accuracy
Baseline	0.375
LinearSVC	0.6811
LogisticRegression	0.6741
MLPClassifier	0.6924

Table 2: Training time of models with feature 1
The feature vector here is encoded [*description* | *cover*]

Model	Training time(s)
Baseline	minimum over all models
LinearSVC	5.47
LogisticRegression	94.346
MLPClassifier	4635.402

Table 3: Accuracy of models with improved features
The feature vector here is encoded
[*description* | *cover* | *similar book* | *rating counts*]

Model	Accuracy
LinearSVC	0.7105
LogisticRegression	0.7081
MLPClassifier	0.7166

Table 4: Training time of models with improved features
The feature vector here is encoded
[*description* | *cover* | *similar book* | *rating counts*]

Model	Training time(s)
LinearSVC	5.05
LogisticRegression	90.34
MLPClassifier	4796.207

As we can see from table 1 to table 4, the baseline model has a bad accuracy when predicting the genre of a book. We also try randomly predict the genre of a book, and the accuracy is about 0.1. The reason why the baseline performs so bad may because it only uses the overall and similar books information to predict, which are not personalize for each specific genre. It's reasonable to make such assumption that a book's genre can be determined by its similar books' genre at first. However, since we only use a

portion of the big dataset, and split the chosen data into 70% training and 30% testing, many books may not have similar books list. Therefore, their genre is determined by the overall most popular genre, which can't be very precise. Then we introduce machine learning models. From the perspective of accuracy, the simple neural network MLPClassifier performs best among all models(highest accuracy), but it's too costly since it takes a much longer time to train the classifier compared to the other two. In general, LinearSVC is an appropriate choice due to its decent accuracy and less expensiveness. For LogisticRegression and LinearSVC model, we try different regularization parameter to prevent over-fitting. We choose the regularization term based on their performance on the validation data, and the best C we pick for LinearSVC is 0.1 and 1 for LogisticRegression. Except comparing the different models, we also try multiple features besides description and cover. Below is the improvement on accuracy.

- similar books: roughly $+2.5\%$
- rating counts: roughly $+0.3\%$
- review counts: no improvement
- average rate: no improvement
- page number: no improvement
- author: No improvement

iii Multilabbel Model

Table 5: Metrics of models
The feature vector here is encoded
[*description* | *cover* | *similar book* | *rating counts*]

Model	Accuracy	Precision	Recall	F_1 score
RandomForestClassifier	0.8289	0.8099	0.4074	0.5421
MLPClassifier	0.88476	0.8159	0.6928	0.7494

Table 6: Training time of models
The feature vector here is encoded
[*description* | *cover* | *similar book* | *rating counts*]

Model	Training time(s)
RandomForestClassifier	402.3986
MLPClassifier	5300.544

As displayed in table 5 and table 6, MLPClassifier has a better results when predicting multiple genres for a book. In the case of multilabel problem, accuracy is an initial overview of the performance of the models, which is not as reliable as the other three metrics. We focus more on the value of precision, recall and F_1 score. Both RandomForestClassifier and MLPClassifier have a nice accuracy, but it can only tell us that most of the predictions are correct. In this task, as mentioned in section 3, we care more about the positive predictions/labels. When looking at the precision and recall value in table 5, we can detect a huge difference between these two models. From the value of precision and recall, we can see that RandomForestClassifier doesn't do well to yield positive predictions, only 40% of which are yield and only 80% of them are correct. This partially explains why it has a high accuracy but low recall. This reveals a potential problem with most multilabel classifiers, that the class is somewhat imbalance but they yield mostly negative predictions. We have tried to assign additional weight to the class that accounts for a small portion in the dataset, but it doesn't improve the scores. On the contrary, the MLPClassifier performs better when looking at the precision and recall. It can make a lot more positive predictions and have a higher accuracy when predicting the positive label. Also, the F_1 score exposes the average performance of both model. However, considering the training time and cost, MLPClassifier is worse than RandomForestClassifier. In general, if we consider the positive labels in the book dataset much more important than the negative labels, then we should pick MLPClassifier to perform the task. If the cost are more important, then a classifier with decent accuracy, precision and recall should be considered to perform the task.

iv Summary

After our experiment, the most related features we find in the dataset are

- description
- cover

- similar books
- rating counts

When a book is considered belonging to one genre, the Support vector Machine is a decent choice with accuracy 71.05%. When a book can have multiple genres, the MLPClassifier is a decent choice with a good precision and recall. In the future, we can try using a deep neural network, exploring more time related features or including more data to approach the task.

References

[1] Wikipedia, Goodreads
https://en.wikipedia.org/wiki/Goodread

[2] Mengting Wan, Julian McAuley, "Item Recommendation on Monotonic Behavior Chains", in Proc. of 2018 ACM Conference on Recommender Systems (RecSys'18), Vancouver, Canada, Oct. 2018.,
https://cseweb.ucsd.edu/~m5wan/paper/recsys18_mwan.pdf

[3] Amazon Product Data, Julian, McAuley, UCSD
http://jmcauley.ucsd.edu/data/amazon/

[4] Holly Chang, Yifan Ge, Connie Wu, "Classification of Books' Genres By Cover and Title", 2015
http://cs229.stanford.edu/proj2015/127_report.pdf

[5] Brian Kenji Iwana, "Judging a Book by its Cover", 2017
https://arxiv.org/pdf/1610.09204.pdf

[6] Logistic Regression Cross-Entropy-Loss
https://datascience.stackexchange.com/questions/20296/cross-entropy-loss-explanation

[7] Mathieu Ravaut, Satya Krishna Gorti, "Faster gradient descent via an adaptive learning rate", 2018
http://www.cs.toronto.edu/~mravox/p4.pdf

[8] StackExchange: What are the measure for accuracy of multilabel data?
https://stats.stackexchange.com/questions/12702/what-are-the-measure-for-accuracy-of-multilabel-data

[9] scikit-learn library, classifiers reference
https://scikit-learn.org/stable/modules/multiclass.html

[10] Tfidf value formula
http://cseweb.ucsd.edu/classes/fa18/cse158-a/slides/lecture10_annotated.pdf