Book Genre Classification

Shreyash Bali and **Pramod Anantha**
University of Illinois at Chicago
sbali3@uic.edu and panant4@uic.edu

Abstract

This paper discusses the classification of book genres based purely on the plot summaries. Several methods were implemented to assess the ability to distinguish books and predict the correct genres. We implemented various multi-label feature extraction and classification techniques. The results showed that the multi-label models implemented performed relatively better as compared to the neural network and the recurrent neural network models. Also, the precision, recall and f1 scores of the best model are quite comparable with the human book genre classification showing that the task in itself is hard because of the overlapping sense of the genres

1 Introduction

One of the most important applications in Machine Learning, and particularly supervised learning is classification, where each item from the dataset belongs to an attribute vector, which represents the range of values from the dataset. Labels refer to the values that the items can belong to. When each item is associated with a single class(label) then it is a single-label classification problem whereas if an item can be associated with multiple labels then it is a multi-label classification. Genre Classification is the technique of classification based on the type/ genre. Genre classification can be done on movies, books , sports etc. Since an item being considered can belong to multiple genres hence Genre classification is in general a multi-label classification problem. Book genre classification is an especially hard problem as it involves a huge number of genres. Our goal is to create a model that can predict the genre of a book based on its summary. If the genre of the book can be know based on its summary, this can help in classifying the books based on the blurb printed on the back cover of the book. Surprisingly it is difficult even for users to identify the exact genre of the book based on its summary. We fed our algorithm the cleaned summaries and tried to teach it to guess the genre of the book . The motivation for solving this problem is for automatic classification of books based on their genres in libraries where they can just scan the last page to get the synopsis which can be fed into the model to obtain the required genre and also online websites like Amazon.

2 Related Work

One of the first works at tackling our particular problem of book genre classification was attempted as Classification of Book Genres by Cover and Title[1] by Holly Chiang, Yifan Ge and Connie Wu. However they used just the cover and the title. They implemented a model which uses transfer learning convolutional neural networks and Stanford NLP classifier. Another work that inspired us was Movies' genre classification based on synopsis (Ka-Wing Ho)[2]. The dataset used was IMDB movie summaries dataset. The number of genres considered was limited to 10. Combining the objectives to the two stated previous works and with the availability of the dataset the goal to classify book genres based on summaries was set. The problem in consideration was a multi-label classification problem. Multilabel Classification: An Overview (Grigorios Tsoumakas)[3] provided quite some insights about the various methods than can be used to solve the multilabel classification. ML-KNN: A Lazy approach to Multi-label Learning (Min-Ling Zhang et.al)[4] tried implementing the ML-KNN model that has been tried out for the given problem.

3 Dataset

The data that is used in this project is obtained from the CMU Book Summaries dataset[7]. The dataset contains plot summaries for 16559 books along with other features like title, author and genre. A single movie can have multiple genres. There are 221 genres listed in the dataset.

4 Preprocessing

The dataset had a lot of inconsistencies. A lot of variables had null values. Hence the dataset was cleaned of all the inconsistent instances. Finally the dataset size reduced to 12505 book summaries. Use of Regular Expressions(RegEx) provided a much cleaner dataset. Although the problem is a multi label classification problem; training the model on 221 different genres in quite a hard task. Hence, all the sub genres were replaced by the parent genre. The replacement reduced the number of genres from 221 to 50. The Google Writing List of Genres[5] was used as a reference for the concatenation. The genre distribution for the dataset is shown below.

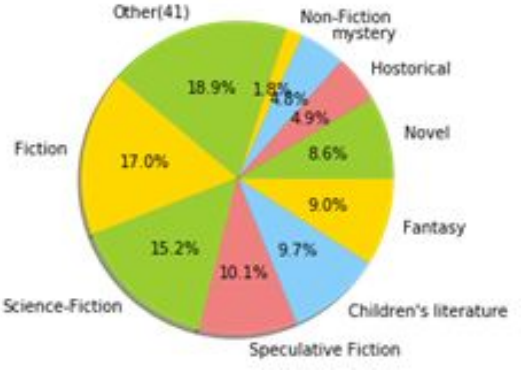

Fig1. Genre Distribution

We can see that Fiction, Science-Fiction and Speculative-Fiction occur the highest number of times.

5 Feature Extraction

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then further passed onto the next stage. Tweet-Tokenizer module provided by NLTK is used for tokenization. Stemming is the process of reducing inflected or derived words to their word stem, base or root form. PorterStemmer and SnowballStemmer provided by NLTK are used.

Vectorization

Bag-of-words model is a simplifying representation of text as a multiset of its own words, disregarding grammar and even word order but keeping the multiplicity. The output instances of the summaries of the stemmer were further fed to a Bag of Words (BoW) model.
Term frequency inverse document frequency (TF-IDF) evaluates how important a word is to a collection or a corpus. TF-IDF of words from summaries were computed.

$$tfidf(w_k, d_i) = w_{ki} * log \frac{m}{\sum_{d=1}^{m} 1\{w_{kd} > 0\}}$$

where W_{ki} is the frequency of the k-th word in the i-th book's synopsis (d_i), and m is the number of training/test sets. The intuition behind calculating the TF-IDF vectors for the words was that certain words in the summaries could be closely linked to the genres and thus further help in the prediction. CountVectorizer method was tried too although TF-IDF Vectorizer performed significantly better.

Multi-Label Binarizer

One-hot Encoding is a process by which categorical variables are converted into a form that can be provided to ML algorithms to do a better job in prediction. The LabelBinarizer is responsible for converting the categorical labels into unique class label integers. The One-hot Encoding function converts the class label integers into a so-called one-hot array, where each unique label is represented as a column in the new array. Multi-Label Binarizer is a modification of one-hot encoding designed for multi-label classification. Here instead of assigning just a single class variable to 1, all labels occurring in the row are set to 1.

	Genre 1	Genre 2	Genre 3	Genre 4
Book 1	0	1	1	0
Book 2	1	0	1	1
Book 3	1	0	0	1

Fig 2. Multi-Label Binarizer Example

6 Feature Selection

Various Feature Selection methods like Chi-square and SelectKbest methods to choose the best features were tried. However, since there was no significant difference in the feature values selecting the best features did not affect the results at all. Infact ignoring the remaining features reduced the metric results and hence were not included in the final mode.

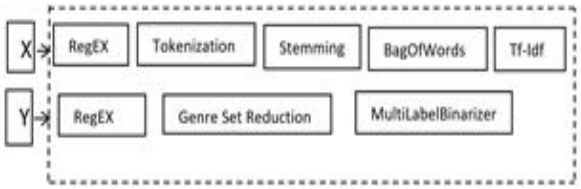

Fig 3. Preprocessing

7 Classification

Traditional Classification techniques don't work with multi-label targets as they permit only multi-class and binary type of problems. The type of problems where the model has to predict a list of relevant labels as the target set for an insatnce are known as multi-label classification problems. Basically, there are three methods to solve a multi-label classification problem, namely:

7.1 Problem Transformation

In this method, the multi-label problem is transformed into one or more single-label classification problems. This method can be carried out in three different ways as:

Binary Relevance

In binary relevance, the multi-label problem is transformed into n different single-label classification problems. Each binary classifier is then responsible for predicting the association of the label. For example, If a book summary had 4 target labels, then it would be trained on each class separately. This is called binary relevance.

Classifier Chains

In Classifier Chains, the first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain. This is quite similar to binary relevance with the only difference that it forms chains in order to preserve label correlation.

Label Powerset

Here, the problem is transformed into a multi-class problem where one multi-class classifier is trained on all unique label combinations found in the training data. So, Label Powerset gives a unique class to every possible label combination that is present in the training set. Linear Support Vector Machine is used for the multi-class classification. The only disadvantage of this is that as the training data increases, number of classes is huge, thus, increasing the model complexity and resulting in a lower accuracy.

7.2 Algorithm adaptation

Algorithm adaptation, involves modifying the existing algorithm to directly perform multi-label classification, rather than transforming the problem into different subsets of problems or performing any preprocessing. For example, multi-label version of k-Nearest Neighbor is represented by Multi-Label k-Nearest Neighbor (ML-kNN). The first step involves retrieving the k-nearest examples. It uses the maximum a posteriori principle to determine the label set for the instance based on the prior and the posterior

probabilities for the frequencies of each label within the k nearest neighbours.

7.3 Neural Networks

We tried different types of neural networks just to see how well multi-label classification performs on them.

Embedding(EMB)

This layer is used to set the input dimension and the output dimension of the features. It is also used to set the limit on the number of words that are allowed from the vocabulary.

Dropout(DROP)

This layer is used to regularize the output so that the model does not overfit on the training data. For example, if the dropout rate is set at 30 percent which means that each time features pass through that layer, 30 percent of the neurons are set to 0.

Convolution1D(CONV)

This layer consists of filters of a specified size passed over to a 1-dimensional feature vector and corresponding feature maps are generated . These feature maps contain important information extracted from the features.

MaxPooling1D(POOL)

This layer is used to generalize the features better so that only the important ones are selected. In this type of pooling, given the pool size, only the maximum value is selected. This layer is usually used in combination with the Convolution1D layer

Long Short Term Memory(LSTM)

This layer is used to combat the vanishing gradient problem that usually occurs in deep learning. During the backpropagation of neural networks , the global optimum is reached by making changes to the gradient of the error function . The problem is that in some cases, the gradient will be so small that it won't cause any change to the error function. In the worst case, it may completely stop the neural network from training. This is call the Vanishing Gradient problem . LSTM layer combats this problem by using its gates which are used to remember the values of the gradient.

Dense(DEN)

The dense or Fully Connected layer is used to perform classification on the features extracted by the previous layers. In a dense layer, every node in the layer is connected to every node in the preceding layer. Usually, the final layer in a neural network is a dense layer.

Pre-Conditions

As this is multi-label classification certain parameters had to be set. The loss function had to be set to binary cross entropy. Binary cross entropy works similar to binary relevance in terms of classification. The activation function on the final layer must be set to sigmoid as that is optimum for Multi-label Classification. The activation functions in the hidden layer were set to ReLU(Rectified Linear Unit) .

Architecture for CNN and RNN

For the Convolutional neural network which usually involves many layers, the architecture is:

EMB → DROP → CONV → POOL → CONV → POOL → DENSE.

For the Recurrent neural network which usually involves many layers, the architecture is:

EMB → LSTM(200) → DROP → DENSE

8 Evaluation measures

Precision:
It can be defined as fraction of the retrieved documents that are relevant to the query.

$$\text{Precision(h)} = \frac{1}{N} \sum_{i=1}^{N} \frac{|h(xi) \cap yi|}{|yi|}$$

Recall:
It can be defined as fraction of relevant documents that are successfully retrieved.

$$\text{Recall(h)} = \frac{1}{N} \sum_{i=1}^{N} \frac{|h(xi) \cap yi|}{|h(xi)|}$$

F1 Score:
It is defined as the harmonic mean of precision and recall.

$$F_1 = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \times |h(xi) \cap yi|}{|h(xi)| + |yi|}$$

Hamming loss:
It evaluates the number of times an instance-label pair is misclassified i.e a label belonging to the instance is not predicted or a label not belonging to the instance is predicted. The smaller the value of hamming loss the better the performance.

$$\text{Hamming\_loss(h)} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{Q} |h(xi) \Delta yi|$$

where h(xi) : relevant documents
y(i) : retrieved documents

9 Results

The evaluation measures include precision, recall, f1-score and hamming loss. The results are tabulated below. The results are evaluated in three sets. First with just the first 1000 samples, then with the first 5000 samples and finally with the whole dataset. Packages used include scikit learn, NLTK, Keras and Tensorflow. Environment is Anaconda and Python 3.5.
In every set of evaluation three further cases are considered. The first is considering only those genres which occur more than 25 times, the second being only the genres occuring more than 100 times and lastly genres which occur more than 200 times.
It can be observed from the results that the adapted models perform significantly better than the convolution neural networks and the recurrent neural networks in all the evaluations. Also there is not much difference in the results for the OneVsRest classifier and the LabelPowerset classifier both performing comparably well. We have achieved results comparable to human standards.
We can also observe that the F1 scores increase as the number of genres decrease thus indicating that the higher the number of labels the more difficult the problem. Also because of the overlapping essence of the various genres with each other it is very hard for the model to predict the exact set of labels as expected.
Some instances and predictions are shown below.
Instance:
y_true: [‘Fiction’,’Fantasy’]
y_pred: [‘Fiction’,’Fantasy’,’Science-Fiction’]

Instance:
y_true: [‘Mystery’,’horror’]
y_pred: [‘Fantasy’,’Mystery’]

Instance:
y_true: ["Children's literature", 'Fantasy', 'Fiction', 'Speculative fiction']
y_pred:["Children's literature", 'Fantasy', 'Speculative fiction']

Instance:
y_true: ['Fantasy', 'Fiction', 'Mystery', 'Science Fiction', 'Speculative fiction', 'Suspense']
y_pred: [‘Speculative Fiction’]

For Genres with	Count > 25 remaining = 19			Count > 100 remaining = 13			Count > 200 remaining = 6		
	P	R	F1	P	R	F1	P	R	F1
OneVsRest (LinearSVC)	**0.68**	0.45	0.48	**0.70**	0.48	**0.50**	**0.73**	0.57	0.59
Label Powerset (LinearSVC())	0.55	0.48	**0.51**	0.59	0.50	0.54	0.69	0.53	**0.59**
MLkNN(k=100)	0.58	0.46	0.47	0.60	0.49	0.50	0.67	**0.59**	0.59
CNN-7 layers	0.45	0.48	0.45	0.44	0.47	0.45	0.58	0.55	0.56
LSTM(100)	**0.68**	0.45	0.48	**0.70**	0.48	**0.50**	**0.73**	0.57	0.59

Table 1. Precision, Recall and F1 Scores for 1000 samples.

For Genres with	Count > 25 remaining = 31			Count > 100 remaining = 18			Count > 200 remaining = 13		
	P	R	F1	P	R	F1	P	R	F1
OneVsRest (LinearSVC)	**0.69**	**0.46**	**0.52**	**0.69**	**0.48**	**0.54**	**0.71**	**0.51**	**0.56**
Label Powerset (LinearSVC())	**0.56**	0.49	**0.52**	0.58	0.50	0.53	0.61	**0.52**	**0.57**
MLkNN(k=100)	**0.62**	0.41	0.46	0.63	0.43	0.48	0.65	**0.46**	0.51
CNN-7 layers	**0.41**	0.48	0.44	0.46	0.48	0.46	0.47	0.49	0.48

Table 2.Precision Recall and F1 Scores for 5000 samples

For Genres with	Count > 25 remaining = 34			Count > 100 remaining = 23			Count > 200 remaining = 16		
	P	R	F1	P	R	F1	P	R	F1
OneVsRest (LinearSVC)	**0.69**	0.41	0.49	**0.69**	0.42	**0.49**	**0.68**	0.43	0.51
Label Powerset (LinearSVC())	0.57	0.46	**0.50**	0.58	0.46	0.51	0.59	0.47	**052**
MLkNN(k=100)	0.59	0.33	0.40	0.60	0.34	0.41	0.61	**0.35**	0.43
CNN-7 layers	0.41	0.49	0.44	0.42	0.48	0.44	0.44	0.49	0.46
LSTM(100)	0.27	**0.5**	0.35	0.30	**0.53**	0.38	0.35	0.57	0.43

Table 3. Presicion Recall and F1 Scores for Full Dataset

Classifier	1000 Samples			Full Dataset		
	C >25	C >50	C >100	C >25	C >100	C >200
OneVsRest	0.1038	**0.138**	**0.228**	**0.053**	0**.077**	**0.105**
LabelPoweset	0.1241	0.145	0.246	0.059	0.085	0.115
MLkNN	**0.1095**	0.159	0.248	0.058	0.084	0.115
CNN	0.1371	0.199	0.289	0.078	0.107	0.15
LSTM	0.1994	0.3	0.39	0.132	0.213	0.289

Table 4. Hamming Loss

Since fantasy and science fiction subgenres share so much in common it is not surprising to see that the model predicts science fiction too in the above example.

The results from the table showing the hamming loss of all the classification tasks when analysed, it is clear that the adaptive multi-label algorithms perform better than the neural network models. It can also be observed that as the dataset grows the hamming loss values decrease. Thus it can be inferred that increasing the size of the dataset along with restriction of the multi-labels improves F1 scores and thus the overall performance of the model.

10 Conclusion and Future work

The adaptive multi-label classification algorithms performed significantly better than the convolution neural networks and the recurrent neural networks. Also out of the three multi-label classifiers OneVsRest and LabelPowerset performed relatively better than ML-kNN. The problem of book genre

classification based on summaries is hard to solve because of the large number of genres a nd also the similarities in the meaning of these genres. It can be found in the dataset that a few of the genres occur in majority as compared to the other genres thus creating an imbalance. Also with limited data it is hard for the model to learn the features leading to inaccurate predictions.

In our future work we would like to implement different neural network and recurrent neural network models by fine tuning the hyper parameters. We would also like to combine the ventors created for the authors and the summaries separately and train on the combined model. Implementing the model on a similar multi-label classification problem and comparison of the results with this problem to check for the correctness of the model.

Acknowledgement

We thank Prof. Barbara Di Eugenio for advising us on this project and encouraging us to take up a difficult problem . This project would not have been possible without her guidance.

Appendix

This project was completed with the equal involvement of both the students. We met every other day and worked on the project . The initial period involved understanding the genres themselves and extensive reading about the various techniques that can be used for multi-label classification. As one of us lacked the high computing systems required for the implementation writing and testing of the code was collaborative.

REFERENCES

[1] Holy Chiang, Yifan Ge, Connie Wu: Classification of Book Genres by Cover and Title

[2] Ka-Wing Ho: Movies' Genres Classification by Synopsis

[3] M. ling Zhang and Z. hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition, 40:2038–3048,2007.

[4] Tsoumakas, Grigorios & Katakis, Ioannis. (2007). Multi-label classification: An overview. IJDWM. 3. 1-13.

[5] A. K. McCallum. Multi-label text classification with a mixture model trained by em. In AAAI 99 Workshop on Text Learning, 1999

[6] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[7] CMU Book Summaries Dataset URL:http://www.cs.cmu.edu/~dbamman/booksummaries.html

[8] Keras Implementation URL:https://keras.io/

[9] Natural Language Toolkit URL: http://www. nltk.org

[10] Wikipedia List of Writing Genres URL:https://en.wikipedia.org/wiki/List_of_writing_genres