Multilabel Movie Genre Classification with Power Transformations and Random Forest Classifier

Shashank Reddy Pulagam
School of Computer Science
San Diego State University
San Diego, California
Email: shashank.pulagam@gmail.com

***Abstract*—The following paper uses the movie dataset on kaggle to classify each movie into different genres. Specifically, each movie provided in the dataset can be classified into 19 labels: action, adventure and many others. To deal with missing values, the dataset is preprocessed to fill in missing values with median or remove certain rows. In addition, one-hot encoding and Tfidf encoding is used to obtain text features. Afterwards, several multilabel classification models such as Random Forest Classifier and Power Transformations are used to train the models for classification.**

I. Introduction

Multi-label classification problems are prevalent in literature in text domain as well as image domain. In text domain, for example, one problem that might be of interest is to classify each book into different genres. To do so, the corpus of various books may be analyzed and models may be trained on corpus features to classify books into different genres like Sci-Fi, Action etc. Whereas in the image domian, each image may have multiple objects like animals, tv etc. Multilabel classification problem is different from Multiclass classification problem in that each sample can have multiple labels whereas in Multiclass classification problem, each sample can only have one possible label. An example might be classifying an animal image to be either Dog, Cat or Lion.

II. Task Description

The paper focuses on applying multilabel classification methods to solve movie genre classification problem. For instance, a movie such as Harry Potter can be classified into Action and Adventure genres at the same time.

III. Major Challenges and Solutions

A. Feature Extraction and Data Cleaning

The major challenges that are encountered while working with the dataset are missing values, list of dictionary features and text features. For reading data files and doing computations numpy[1] and pandas[2] are used. To deal with list of dictionary of features, the structures of the dictionaries is closely inspected, and the dictionaries are iterated one after the other to create one-hot encoding labels for the text labels of interest. To deal with text features, stop words, most commonly occurring words that hold no meaning, are removed using nltk library[3]. In addition, punctuations are removed, sentences are lower cased, stemmed and lemmatized and Tfidf feature extractor is used to obtain word frequencies for 1-2 word ngrams. In addition, some features like the average word length, the number of words and the number of characters in a sentence are created to analyze the sentences further.

Tfidf, which stands for Term Frequency Inverse Document Frequency, is a technique which first computes the term frequency T_f, i.e. the number of times a word occurs in a sentence, and multiplies the term frequency by the logarithm of number of sentences S divided by the term frequency T_f [4].

$$Score = T_f * log(S/T_f)$$

The idea is that more frequently occurring words will have higher scores than those that do not occur as frequently. Then after, the missing values in the data are replaced with the median value of the column.

B. Dimensionality Reduction

Another challenge encountered while working with the dataset is the high dimensional feature space. While preprocessing text features like Tfidf, many features from text that may be useful for classification are chosen. However, the feature space eventually becomes so large that it is impossible to train machine learning models unless some redundant features are removed and summarized into few important features or components. To solve the problem, Scikit-learn's implementation of RandomizedSVD is used to reduce the data to 200 most significant components [5].

IV. Experiments

A. Dataset Description

The Movies Dataset has metadata about movies in a file called movie_metadata.csv that has approximately 46,400 movie samples and 24 columns such as the date of movie release, the status of the movie release, the genres of the movie, overview summary, the title, tagline, the production companies, the production countries, the languages the movie was released in, the homepage link, the imdb id, id, the original language of the movie, revenue, budget and others[6].

Figure 1 included below shows the genre labels plotted with seaborn[7] and matplotlib[8] that each movie is classifed into. As can be seen, in totality there are 19 possible genres:

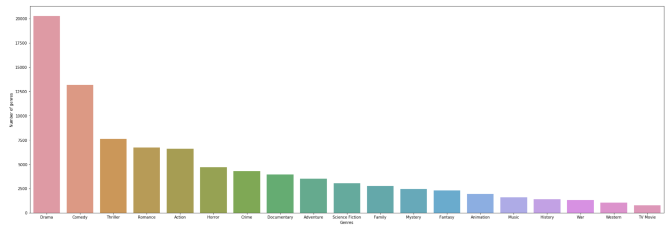

Fig. 1. Distribution of genre labels

Action, Adventure, Animation, Music, History, War, Western, TV Movie, Music, Fantasy, Family, Drama, Comedy, Thriller, Romance, Horror, Crime, Documentary and Mystery. Approximately 45% of the movies in the dataset are dramas, 33% are comedy and 16% are thrillers. The rest of the genres are little bit infrequent; however, the dataset is not entirely imbalanced for classification tasks.

B. Data Visualization

In addition to the features provided in the movie_metadata file, some keywords provided in keywords.csv such as jealously, friendship are included for each movie. As can be seen above in Figure 2, most frequently occurring keywords are murder, based on novel, musical and several others. Analyzing keywords, in addition to the features already provided in movie_metadata might help to predict some genres better.

Figure 3 included above shows some of the most common words occurring in the movie overview descriptions visualized with WordCloud visualization library[9]. As can be seen, the big words are love, man, friend, work, family, wife, live, and discover all of which are contextual to better predicting romance, drama and comedy movie labels.

Figure 4 shows that overall between years 2000 and 2018, users found movies under genre Adventure popular. Year 2020 is also included because some movies that are yet to be released are included in this dataset as well. After adventure, users find Western and Fantasy movies more popular. The popularity ratings based on different years might help to predict certain movie labels accurately as some genres are more popular than others in certain years.

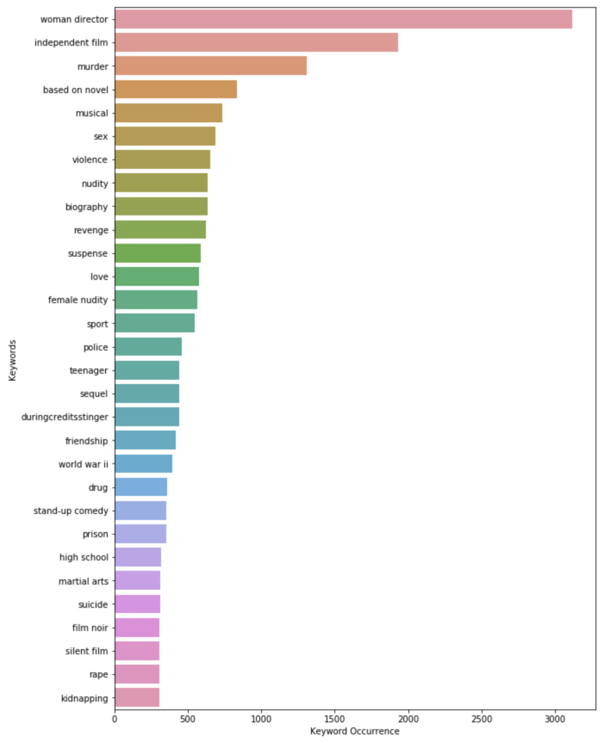

Fig. 2. Top 30 frequent keywords in the dataset

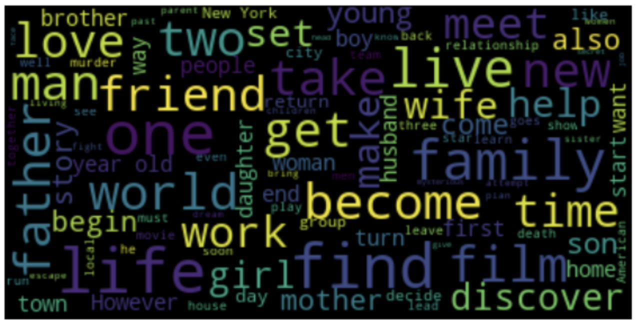

Fig. 3. Hundred most common words in movie overview

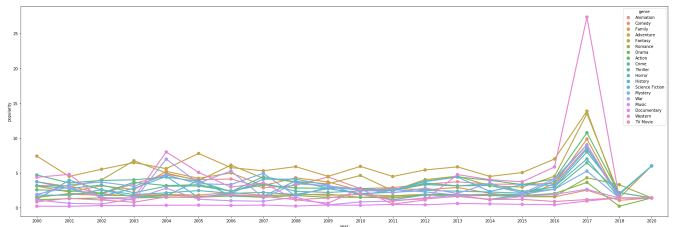

Fig. 4. Movie popularity for different genres over the years

C. Association Rules

X	Y
alien	Science Fiction
Canal+	Drama
friendship	Drama
based on novel	Drama
biography	Drama
independent film	Drama
love	Drama
world war ii	Drama
Romance	Drama
History	Drama
War	Drama
martial arts	Action
suspense	Thriller
dystopia	Science Fiction
stand-up comedy	Comedy
History,War	Drama
love,Drama	Romance
love,Romance	Drama
suspense,Drama	Thriller
Romance,independent film	Drama
Foreign,Romance	Drama
Romance,Crime	Drama
Thriller,Romance	Drama
Romance,woman director	Drama
Adventure,Thriller	Action
Mystery,Horror	Thriller
Mystery,Drama,Crime	Thriller

Table 1: Association Rules based on keywords and genres

As can be seen in table 1, keywords can be used to predict certain labels well. The above association rules are generated using a minimum support of 0.005 and confidence of 0.65. For instance, movies containing alien keyword most certainly are Science Fiction related and movies containing friendship keyword are Drama related. Furthermore, some genres also cooccur together. For instance, a movie that is a Mystery and a Crime and is labeled with Drama keyword is most likely a Thriller. Another rule that is discovered is that a movie that is a Mystery and Horror is more likely to be a Thriller too.

D. Evaluation Metrics

Among the evaluation metrics for multilabel classification problem, I chose to use F1-macro averaging. F1-macro averaging computes F1 score for each label by using F1-score formula of multiplying recall, precision by 2 and dividing by the sum of recall and precision [10].

$$F1 = 2 * recall * precision/(recall + precision)$$

Once F1 score for each label i is obtained, an average F1 score is obtained by taking sum of F1 scores for each label and dividing by the total number of labels n.

$$F1_{macro} = \Sigma_i F1_i/n$$

F1 macro score is used even though some genre labels occur less frequently because the labels are not highly infrequent for the dataset to be considered imbalanced.

E. Major Results

All the multi-label classification methods are evaluated using 5-fold cross validation, which is created with KFold in scikit-learn library. The dataset is split into 5 folds, where 4 folds are used each time to train the model and one left is left out each time in turns to evaluate the model upon and obtain an average and robust metric score.

Model	$F1_{Macro}$
Binary Relevance + GaussianNB	0.223
Label Powerset + GaussianNB	0.222
Classifier Chain + GaussianNB	0.217

Table 2: Multilabel classification models + Naive Bayes

1) Binary Relevance: Binary Relevance is a power-transformation technique that treats each label to be classified independently from the other labels. One classifier is trained on one label class; hence, in total n classifiers will be used to train to classify n different label classes. It may fail to capture associations between the labels as it considers each label independently [11].

Gaussian Naive Bayes model is fed as the classifier to the Binary Relevance model in scikit-multilearn library[12]. As can be seen in table 2 above, the model has performed relatively well when compared with its counterpart multilabel classification models with an 5-fold cross validation F1 macro score of 0.223.

2) Label Powerset: Laber Powerset, unlike Binary Relevance, treats a subset of labels as one label and then fits a classifier to each of those subset of labels. The problem is that it could be computationally expensive if labels occur in many combinations as many classifiers will be trained to fit each of those label combinations[5].

Gaussian Naive Bayes model is used as the classifier to label powerset. As can be seen from the table, label powerset performs a little poor than Binary Relevance and better than Classifier Chain with 5-fold cross validation F1 macro score of 0.222.

3) Classifier Chain: Classifier chain is another technique that is used to solve multilabel classification problems. Just like binary relevance, classifier chain also trains n models as there are n labels; however, each classifier at label l learns how to classify a label based on the classifications made by models for labels 1..l-1 [13].

Just as for Binary Relevance and Label Powerset, Gaussian Naive Bayes model is fed as the classifier to Classifier Chain. As can be seen in table 2, the model has performed the worst with a 5-fold cross-validation F1 macro score of 0.218.

n-estimators	max-features	max-depth	$F1_{Macro}$
50	25	10	0.052
100	20	15	0.0724
20	25	20	0.101
20	20	20	0.098
50	25	20	0.093
100	15	20	0.083

Table 3: Scores for different Random Forest models

4) Random Forest: Random Forests is an ensemble method which combines results from many decision trees trained on different random subsets of features. It is robust as it considers results of multiple decision trees to make its final prediction [14].

Randomized Search technique in scikit-learn is used to iterate through hyperparameters for Random Forest, with n_estimators of [10,20,50], max_depth of [10,15,20] and max_features of [15,20,25]. n_estimators hyperparameter decides upon how many decision trees to use, max_depth decides upon what the maximum depth of decision tree should be and maximum features decides how many features in totality should be considered while splitting in decision tree. Based on the table above, the best model found uses 20 estimators, 25 maximum features and 20 maximum depth and scores 0.101 on the 5-fold cross validation.

F. Analysis

Despite some hyperparameter tuning for Random Forest classifier, the F1 score has not received a significant boost with maximum being 0.1012. All the three power-transformation methods outperform Random Forest classifier with F1_scores above 0.217. Since classifier chain and label powerset each performed worse than binary relevance, it indicates that perhaps the genres are not strongly associated amongst themselves, i.e. they are independent to the most extent. The performance has not been so great as the training sample is a little small to obtain proper coverage for each genre label.

V. Conclusion and Future Works

BinaryRelevance worked better than other models maybe because the genres are independent of each other. However, a comprehensive model has not been discovered to properly classify the movies into genres.

Perhaps a larger dataset would be needed to capture enough information to classify each genre label accurately. In addition, some text feature extraction techniques like word embeddings and some other variables involved in a movie like the cast, directors and producers can be used to obtain a better representation of the data. Also, some time-series deep learning models like Recurrent Neural Networks can be used to predict genres making use of time information of popularity of genres every year etc.

Other venues to explore might be some ensembling multilabel classification models like MajorityVotingClassifier or some unsupervised clustering methods to obtain labels and then classify based on those cluster labels.

References

[1] Stfan van der Walt, S. Chris Colbert and Gal Varoquaux *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science Engineering, 13, 22-30 (2011)

[2] Wes McKinney. *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

[3] Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. OReilly Media Inc.

[4] J. Ramos, *Using TF-IDF to Determine Word Relevance in Document Queries.*

[5] Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, douard Duchesnay. *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, 2825-2830 (2011)

[6] Rounak Banik, *The Movies Dataset*, Version 7. Kaggle, 2018.

[7] Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, Adel Qalieh. (2017, September 3). mwaskom/seaborn: v0.8.1 (September 2017) (Version v0.8.1). Zenodo. http://doi.org/10.5281/zenodo.883859

[8] John D. Hunter. *Matplotlib: A 2D Graphics Environment*, Computing in Science Engineering, 9, 90-95 (2007)

[9] A. Muller, *WordCloud*, WordCloud for python documentation. Sphinx.

[10] M. Sokolova and G. Lapalme, *A systematic analysis of performance measures for classification tasks*, Information Processing and Management, vol. 45, no. 4, pp. 427437, 2009.

[11] O. Luaces, J. Dez, J. Barranquero, J. J. D. Coz, and A. Bahamonde, *Binary relevance efficacy for multilabel classification*, Progress in Artificial Intelligence, vol. 1, no. 4, pp. 303313, 2012.

[12] S. P. and K. T. *A scikit-based Python environment for performing multi-label classification*, ArXiv e-prints, Feb. 2017.

[13] J. Shan, C. Hou, H. Tao, W. Zhuge, and D. Yi, *Co-learning binary classifiers for LP-based multi-label classification*, Cognitive Systems Research, vol. 55, pp. 146152, 2019.

[14] L. Breiman, *Random Forests*, 2001.