IMDB Genre Classification

Tran Hieu Minh, Bui Huy Dong, Dong Van Manh,
Le Song Vu, Vu Quang Thai
University of Engineering and Technology
Vietnam National University
Ha Noi, Viet Nam

***Abstract*—This paper outlines the development of a specialized neural network designed for genre-based movies categorizing by using data extracted from theirs titles and cover images. Through different approaches, we demonstrate a methodology in initializing and improving the models accuracy. The results leverages and elaborates our knowledge on the respective field and hopefully contribute to further usages.**

***Index Terms*—neural network, machine learning, movies, multi-label, classification**

I. Introduction

In the contemporary landscape of technological advancements, the integration of Natural Language Processing (NLP) and image processing stands out as a prominent trend, which are both addressed in this multi-factor problem. By combining languages and images features, our approach seeks to make use of those data for designing a movie genre classification model.

The value of our research lies in its dual significance: meeting the demands of practical applications while contributing to the academic discourse surrounding these technologies. As progressing, we faced not only short-term challenge but also build the base for future advancements at the the field of NLP and image processing.

II. Methodology

As the data is already collected and structurally parsed, the following subsections detail the key steps undertaken in our research:

A. Data preprocessing

The dataset processing for our movie genre classification task involves a series of steps aimed at preparing and organizing the textual and visual information. The comprehensive processing pipeline ensures the integrity and relevance of the data for subsequent model training and evaluation. Utilizing the Pandas library, we load the movie data from the provided files, incorporating relevant columns such as movie ID, title, and genre. The 'genre' column is processed to represent each movie's genre as a list of genres, enabling a more granular analysis of multiple genres associated with a single movie.

Corresponding image paths are assigned to each movie based on the provided source image folder (ml1m-images), ensuring a direct link between the movie data and its associated cover image. The 'title' column undergoes preprocessing, removing the last six characters, which typically represent the release year. This ensures consistency in textual representation across the dataset. Unique indices are assigned to each movie, facilitating referencing and retrieval. The indices are used to construct image paths and are stored in new 'id' columns. The genres are loaded from a text file to create a list of unique genres. This list is used as the basis for genre labeling in subsequent steps.

The distribution of genres in the training set is visualized through a bar plot, offering insights into the diversity and prevalence of genres within the dataset. Movies without available cover images are identified and dropped from both the training and testing sets. The number of dropped instances is recorded for further analysis.

B. Model architecture

The architecture of our neural network is designed to seamlessly integrate both textual and visual features for effective movie genre classification. The model consists of distinct pathways for processing textual and image data, followed by a merging layer to combine the extracted features. The architecture is outlined below:

Leveraging the widely-used ResNet18 as our base model for image processing, we extract meaningful visual features from movie cover images. The pretrained ResNet18 is truncated to retain its convolutional layers and batch normalization layer is applied to enhance the stability and convergence of the visual features.

The movie titles are processed through a linear layer, transforming the text into a feature representation. This step captures semantic information from the textual input.A rectified linear unit (ReLU) activation function is applied to introduce non-linearity and enhance the expressive power of the model. The concatenated features undergo an additional linear transformation, providing the model with the capacity to learn intricate relationships between textual and visual representations.

The final layer of the model is a linear transformation that maps the combined features to the number of output classes, corresponding to the different movie genres.

We also try a self-build CNN approach on image processing inside the architecture, employs a convolutional neural network (CNN), comprising convolutional and pooling layers to process movie cover images. Batch normalization enhances the stability of visual features. The outcome will be discussed later.

C. Training and evaluating

The training process involves iterating through the training dataset for a specified number of epochs, updating the model's weights, and monitoring the loss. Simultaneously, we evaluate the model on a separate validation set to gauge its generalization capabilities.

The model is initialized (in this case, an instance of the BaseModel class) and moved to the available device for efficient computation. Worth to note that, learning rate, dropout probability, and early stopping patience are hyperparameters that can be tuned to optimize the model's performance. In this case, the learning rate is set to 0.001, and dropout is not utilized.

The model is trained using the training dataset, iterating through batches of movie samples. The loss is computed using the BCEWithLogitsLoss function, which is suitable for multi-label classification problems. The optimizer (Adam in this case) updates the model parameters based on the computed gradients during backpropagation. Training progress is logged, and the average training loss is displayed periodically for monitoring.

After each epoch, the model is evaluated on the validation dataset. The BCEWithLogitsLoss is computed, and additional metrics such as F1 score are calculated for a more comprehensive assessment. [8] The validation loss and F1 score are reported, providing insights into the model's performance on unseen data.

To prevent over-fitting, we implement early stopping. If the validation loss fails to decrease for a specified number of epochs, training is halted, and the best-performing model is retained. This mechanism ensures that the model generalizes well to new data and prevents unnecessary computational expense.

D. Testing

The testing phase of our movie genre classification model involves evaluating its performance on a separate test set to assess its ability to generalize to unseen data, which is critical on real world applicability.

The testing process is performed batch-wise, allowing for incremental evaluation and reporting. This enables the monitoring of the model's performance throughout the entire test set.

The trained model is applied to the test dataset, generating predictions for movie genres based on the movie titles and cover images. The top-k predicted classes and their corresponding probabilities are extracted for each movie in the test set.

The top-k predicted classes are compared to the actual genres in the test set. The average precision at k (AP@k) is computed for each movie and then averaged across the entire test set, return the mean average precision at k (MAP@k). This metric accounts for the position of correct predictions within the list of top-k predictions, providing an over-look on model's performance.

III. Result

In the preceding sections of this paper, we introduced and implemented a myriad of components and methodologies, each playing a crucial role in the pursuit of our overarching objective – the accurate classification of movie genres based on their titles and cover images. Throughout the experimentation phase, we delved into the utilization of diverse components, ranging from neural network architectures and data processing techniques to novel evaluation metrics. Each of these elements contributes uniquely to the fabric of our research, and it is now imperative to illuminate the technical underpinnings and theoretical rationale behind their incorporation.

In this section, our focus shifts from the comprehensive overview provided earlier to a more detailed exploration on the technical intricacies and theoretical foundations of our research.

A. On facing preprocessed data

Recognizing the redundancy and lack of contextual significance in certain elements such as release years, we strategically eliminated this information to streamline the input features. Additionally, the extraction of meaningful information from movie titles necessitated the removal of common stop words like "A" and "The," ensuring that the remaining text contributes substantially to the model's understanding of genre-related content. However, a noteworthy consideration arose regarding movies with relatively sparse titles, where text volume was insufficient for robust model training. This limitation led to thoughtful decisions during data processing, which we elaborate on in subsequent sections.

We consider image as a rich source of characteristic can be extracted for our task. Unexpectedly, a significant proportion of movies, exceeding 500 rows in the training set alone, lacked their corresponding cover images. Recognizing creating an automated download and cropping script is lackluster, we went with a more effective strategy on this problem, prioritizing data quality over quantity. This strategic omission aimed to enhance the overall outcome quality instead of traditional approach to create random noises.

B. On imbalanced data

During the iterative training process, we observed a phenomenom: the domination of certain classes within the dataset make the model harder to generalize effectively. This led to the inflation of accuracy metrics, primarily driven by the overwhelming representation of majority classes while overshadowing the classification of minority classes.

To address this issue of imbalanced data, a crucial step was taken to visually inspect the distribution of classes within the dataset which give us better insight of what actually going on in the dataset.

An used strategy is weighted classes. By assigning differential weights to individual classes, we reduce the dominance of majority classes, allowing the model focus more on minority

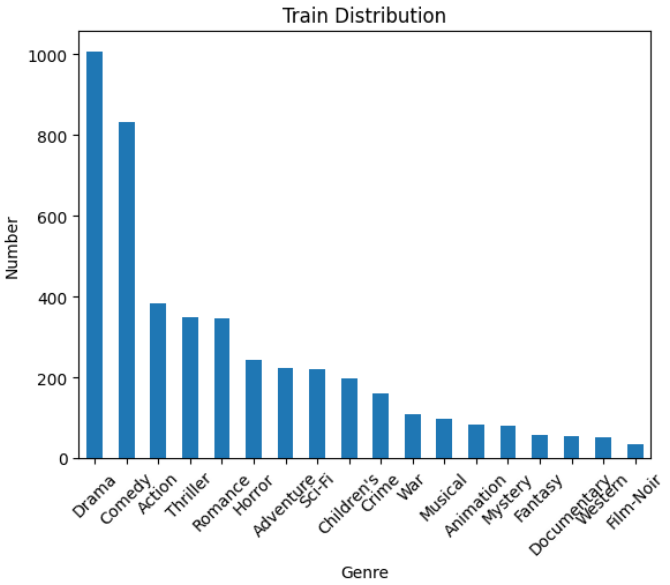

Fig. 1. Visualize of classes distribution in train set.

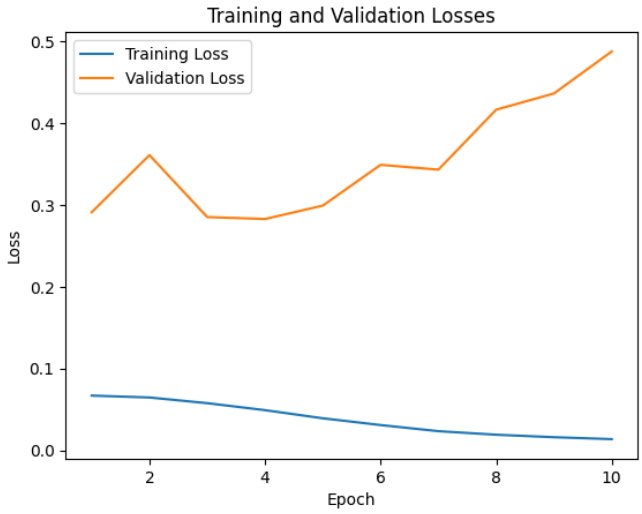

Fig. 2. Visualize of training and validation losses in 10 epochs.

classes. This adjustment directly targets the bias introduced by the imbalanced distribution, creating a better learning process.

Another approach is the implementation of early stopping. We suspect that over-fitting could make worse impact on the model in addition to imbalanced dataset. It prevents the model from memorizing the imbalanced training data and allowed training process to halt at the optimal epoch. As the result, early stopping reduce fake accuracy as the model will not be over-fitting, especially on dominant classes.

C. Model Architecture

As the results is not as good as we expected, we tried different sequences of model architecture.

One significant modification involved the Rectified Linear Unit (ReLU) activation functions in the model. We recognized that connecting multiple linear layers without non-linear activation functions will not allow the model effectively learn the characteristic as features cannot be extracted from outcome linear layer. By using ReLU activation, we added non-linearity into the model, enabling it to learn more complex and representative features.

We also made an attempt of structuralized image processing layers, combining of convolution layer and maxpooling, which yielded suboptimal results. So we decided to use pretrained model in our project, which is expected to extract more representations of characteristic on movie cover images. The decision to integrate a pretrained image model served two primary purposes: first, to capitalize on the learned features and hierarchical representations embedded in the pre-existing model, and second, to enhance the overall model on image features.

As the graph show, the model does not have a good result, but start to resemble the learning curve of classification problem.

D. Outcome

We evaluated the model by using map@k (Mean Average Precision at k) metric, along with human evaluating.

As the map@k metric struggle to pass the goal of 0.6 precision [6], our team tried with text predict output and come to conclusion that the model have acceptable result. There are plenty of room for improvement and modification, we will discuss that in later section.

The result refers to our IPynb notebook script. Please note that the hyper parameters used in the notebook does not represent the best model as we are still testing them.

IV. Discussion

Acknowledge that we could have done better to improve the quality of this project, our team have some notes while polishing the model, hoping for further modification.

Data parsing: We could improve our vocabulary list to have better quality on that set. Either we drop the meaningless words (for example: ”The”, ”A”, ”’s”,...), or sorting out the special character exists in the titles that we missed. The lackluster lead to lower quality data and we are trying to improve on that.

Parameters optimization: Hyper parameters play an important role on training process. We do not have enough experienced nor time to test for wider range of parameters. Strategies such as grid search, random search, or more sophisticated methods like Bayesian optimization could have been applied.

Model structure: The complexity of the current model structure poses another avenue for enhancement. Consideration should be given to reducing model complexity through techniques such as feature selection, dropout, or even exploring alternative architectures. As our model is over-fitted, we are researching on alternative model structure that suit the task.

Imbalanced data: The imbalances have not been handle properly. There are more strategies to resolve the problem, for example, over-sampling and under-sampling. Another interesting method we are trying to implement is to increase weight of majority classes along with down-sampling their data, which helps those specific classes converge faster. [7]

V. Conclusion

In conclusion, while our current model is not good enough, there are areas where we can make it even better. We can do this by modify some settings, changing the way the model is built, and handling imbalanced data more carefully. We believe these changes will greatly improve how well the model works. Exploring different methods, like using multiple models together and learning from existing models, can contribute to creating a stronger and more effective movie genre classification system.

Our commitment to getting better matches the changes happening in machine learning. The improvements we've suggested are steps towards making a more advanced and precise model. Future research can explore these ideas more deeply, leading to a new phase of improved models that can better understand the details of movie genres with outstanding accuracy.

References

[1] S. Kolassa, "Are unbalanced datasets problematic, and does oversampling help¿'
[2] J. Brownleee, "Multi-label Classification with Deep Learning"
[3] Z. A. Daniels and D. N. Metaxas, "Addressing Imbalance in Multi-label Classification Using Structured Hellinger Forest"
[4] P. Sharma, "Build Multi-Label Image Classification Model in Python"
[5] P. Joshi, "Predicting Movie Genres using NLP"
[6] N. Gupta, "Mean Average Precision MAP@k metric" on Kaggle
[7] Google Developer Foundation Class
[8] Pytorch Documentation