Book Genre Classification using Natural Language Processing

Harsimran Kaur, Lakshmi Samhitha Yadlapalli, Sravani Subraveti
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada

***Abstract*—The book genre helps in classifying the books into various categories and thus summarizing the book into a label. The extraction of information about the book genres through its cover can be useful in various domains. The project focuses on implementing a complete application that inputs book cover image and outputs the genre of the book. The task of classifying the book title into its respective genres is cumbersome owing to the ambiguity in the title and the overlapping of labels. However, the application utilizes several deep learning models, different embedding matrix implementations, Natural Language Processing(NLP), and Optical Character Recognition(OCR) to develop a complete end-to-end solution for book title extraction and classification of the book into its genre. The report also compares various models implemented based on their test accuracy and selects the best one.**

***Keywords*—Books, Recurrent Neural Network (RNN), Natural Language Processing (NLP), Optical Character Recognition (OCR), Global Vector (GloVe)**

I. INTRODUCTION

Books are the source of knowledge. There are hundreds and thousands of books in libraries. To reduce the time to search for a particular book in the aisles of the library, we can set up a system to scan the image of the book cover or enter the book title to give the genre and aisle number according to the title. Digitalization, in the world of books, is possible by using, Deep Learning and Machine Learning models. The learning capability of the designed model helps us to predict the genre of the books.

There are multiple ways to organize books in libraries. For instance, one can use the Authors and book title, or date published and book title. However, organizing books based on genres, makes life easier for both the reader and librarian. There are multiple type of genres. The report only focusses on 30 of them which are presented in the Book Genre Classification dataset extracted from the paper[2].

In the project, the genres of the books are identified based on the title of the book. The book cover image is used to extract the title using Optical Character Recognition (OCR). OCR is implemented using tesseract in OpenCv library in Python. To process the obtained text, the latest technique like Natural Language Processing (NLP) is used. The obtained features are passed to the Recurrent Neural Network (RNN) model which has Long-Short Term Memory (LSTM) layer.

Initially, the training of the model is done using the title of books from the dataset to obtain the book genre. Later, after the model is trained, a desktop application is developed to ease the genre identification method. A book cover image is captured and passed to OCR module to identify and recognize the title. The obtained title is preprocessed using NLP techniques and passed to the trained LSTM model to obtain the genre of the book. Accuracy is used as an evaluation metric to evaluate the performance of the model.

The project tries to identify the comparison between different ways of generating the word embeddings from the corpus of text dataset available. Moreover, multiple types of deep learning models like CNN, RNN, simple neural network are developed to compare their performances. The entire project is developed in Python 3.6 language and utilizes its various libraries like Keras, numpy, pandas, matplolib, gensim, scikit-learn, NLTK, Open-CV, pytesseract, Wordcloud.

This paper is organized in sections as following: Literature Review in section II, Background in section III, the Experiments in section V, Analysis on different models used in section VI, Discussions on the models implemented in sectionVII, future scope in section VIII & conclusion in section IX.

II. LITERATURE REVIEW

Classification of genre has been implemented by many researchers to classify music by its genre [9] [11]. They have used Natural language processing, text analytics and machine learning and deep learning models to perform their task. The two papers referred to design the approach for classification of genres are:

A. Classification Based on Book Cover

To understand the relationship between a book and its cover, a conference paper [2] was published by the German Research Centre for Artificial Intelligence. In this paper, they specified the possibilities to get the genre of the book based on the cover. They have done two types of approaches with two different variations in the dataset. The first one, the image data set, is used to the Convolution Neural Network (CNN), here the image recognition is performed. They have used AlexNet pre-trained network on ImageNet. By doing so, they have leveraged the learned features and pass it to different applications. This helps to improve the generalization of the large image dataset. As a comparison analysis, LeNet was also trained however, it showed similar results as AlexNet. In the Second Approach, a large dataset created had 32 classes, consisting of book cover images, title text, author text, and category membership. Here, AlexNet is also as pre-trained on ImageNet, though we have already obtained state-of-the-art document classification results, and we have had limited accuracy. Strong complexity level for the proposed dataset. The image dataset categorized into color, object, and text. In each type, one genre dominantly was recognized by the Alexnet model. For instance, in color, children's and hobby books were identified well, as the book cover is colorful. In object type, Law, Travel, Cooking books, were recognized well, because of the object detection. In-text type, Mystery books, were identified well. Out of all 30 classes, the highest recognition was 68%, for Test Preparation Genre books. Since the books have large font text on the cover page, it is easy for the model to recognize the text. After, the analysis, they have understood that it is a difficult task as several books have cover images with few visual features or uncertain aspects that cause so many incorrect estimates.[2]

B. Classification Based on Book Cover and Title

The paper [7] presented an approach to identify book genres using the book title and cover. The algorithm proposed in the paper uses Support Vector Machines to perform the classification. A color-based distribution in HSV space is used to identify the relation between the genre with the color of the book cover. Feature extraction is performed using transfer learning with Convolutional Neural Network(CNN) on the cover images and Natural language processing(NLP) on the book title text. The dataset for the problem was taken from OpenLibrary.org which contained 6000 book cover images classified into 5 genres: Business, Fantasy, History, Science-Fiction, and Romance. The dataset was pre-processed using different computer vision techniques like grayscale conversion of the image, blue filtering, and adaptive thresholding. Features were extracted using Transfer learning from the ImageNet model. The features extracted were normalized. These features and their labels combined with the NLP features were stored in a dictionary. Text Classification was performed using the classifiers, Stanford Classifier, and Word2Vec classifier. The features extracted from these classifiers were used to find the distance between each word in the book title and genre. The combined features were used for multi-class classification using Support Vector Machine(SVM). The proposed approach produced approximately 62.4% accuracy with SVM and 57.3% accuracy with transfer learning and NLP approach.

III. BACKGROUND

Natural Language Processing (NLP) is an artificial intelligence branch that uses natural language to communicate with computers and humans. NLP's overall goal is to view the human languages in a meaningful manner, translate them, understand, and make sense of them. [5] One of the most significant tasks of NLP is Text Classification. It is the process of categorizing text into structured categories. Text classifiers can automatically interpret text using Natural Language Processing (NLP), and assign a collection of pre-defined labels based on its content. It is a supervised machine learning problem where the model is trained with pre-defined labels and predictions are made. However, the test data available is highly unstructured and raw. Moreover, the comprehension and manipulation of text is an extremely complicated task. Thus various strategies and preprocessing are required to bring the data into its desired form. The classification is performed using various Deep Learning algorithms like Convolutional Neural Layer Networks(CNN), Recurrent Neural Networks(RNN), Long Short-Term Memory(LSTM), etc. These algorithms require data to be in numeric form. Therefore, the main objective is to convert the unstructured text data into an equivalent structured numeric data. The representation of text such that the words with similar meaning have identical representation is called Word Embedding [**?**]. Each word is represented as a real-valued vector in a predefined vector space. The real-value representation of the predefined vocabulary from a corpus of text is learned through various word embedding techniques: [3]

A. Embedding Layer

A word embedding layer is learned in combination with a neural network algorithm, on a common role for processing natural languages. The text needs to be cleaned and processed such as a one-hot representation of each word before it is passed to the embedding layer. The layer inputs the dimensions of the vector representation of each word and the vocabulary size of the text. The dimensions are initialized with small random values. The Embedding layer is used as the first layer of the neural network and is fitted using a supervised learning approach via the Back-propagation algorithm. The vectors are treated as trainable parameters of the model. The one-hot encoded representation of a word is mapped to the word vectors. The word vectors are joined together and fed as input to a Multi-Layer Perceptron model. In the case of Recurrent Neural Network, each word is passed as input in a sequence [3]. This technique involves a lot of training data and can be cumbersome, but can produce an embedding targeted to the specific data.

B. Word2Vec

Word2Vec is a computational method for learning a word embedding from a text corpus efficiently. Tomas Mikolov at Google developed this approach is 2013 to make neural network-based training of the word embeddings more efficient. Word2Vec represents different words with a vector chosen carefully to indicate the semantic similarity between the words represented by those vectors [12]. It uses two models to produce the vector representations: Continuous Bag of Words(CBOW) and Continuous Skip Grams. CBOW predicts the current word using nearby context words. The prediction of the model is not affected by the order of the context words. The continuous skip-gram model utilizes the current word to predict the surrounding context words [3]. The context is defined by a window of neighboring words which is hyperparameter of the model and needs to be configured. Large window size produces more relevant similarities while the small window size produces more syntactic similarities. The surrounding words are given more weights than the other words. CBOW is relatively faster than skip-grams. However, skip-grams works better for less frequent words [3]. Word2Vec approach is more efficient than the Embedding approach as it has low space and time complexity and generates high-quality embeddings.

C. Global Vector

Pennington and his fellow researchers at Stanford developed the Global Vector (GloVe) word representation is an extension to the Word2Vec approach to further enhance the efficiency of learning the vector representations of the word. The original approaches of vector representations of words used matrix factorization techniques like Latent Semantic Analysis(LSA) which used global text statistics to produce the vector spaces. However, it was less efficient in identifying the meaning of the words and fails to recognize word analogies. GloVe used both global statistics for matrix factorization similar to LSA along with local context-based learning like Word2Vec [3]. Thus uses the advantages of both the approaches to perform the task. GloVe constructs an explicit word-context matrix using statistics [3] from the entire text corpus as an alternative to the local context window of the Word2Vec technique. Thus produces a more efficient word embedding. It is a highly recommended model for identifying word analogy, similarity, tasks.

IV. Proposed Approach

The training procedure is define in the flowchart in figure 1.

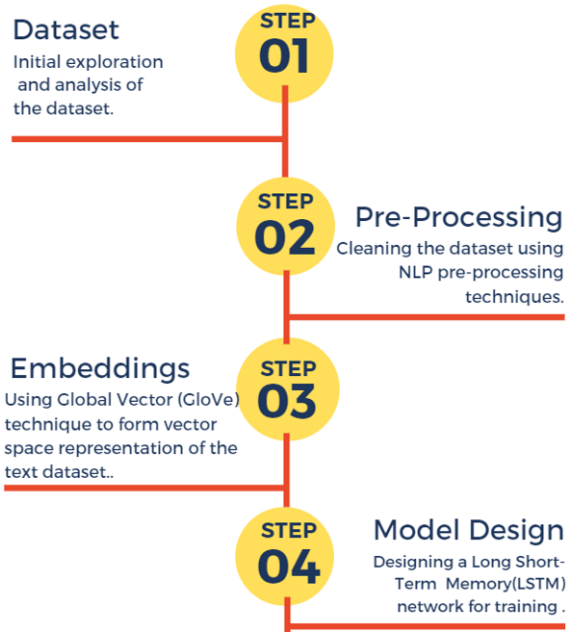

Fig. 1. Training Process

A. Dataset

Multiple datasets from sources like Kaggle and previous papers[2][7] were explored to classify the books

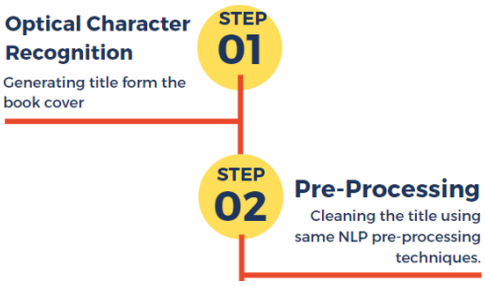

Fig. 2. Testing Process

based on the genre. The dataset from the paper [2] was chosen to perform the task. The dataset has a well classified and unique genre categories. It contained an ample amount of data to analyze and evaluate the approach. Moreover, the initial exploration of the dataset revealed a clear balance in terms of the number of samples for each category of book genre as shown in figure 3. No null values and duplicates were found in the dataset.

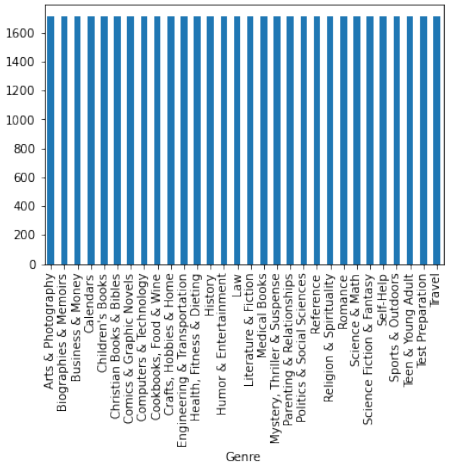

Fig. 3. Dataset Exploration

The dataset has 57000 samples, containing book titles, genres, numeric labels for genres, URLs to the book cover images, and author names. The dataset was filtered by selecting only the title, the genre, and its respective numeric label using the pandas library. The filtered dataset was split into a 90-10% train-test ratio using "sci-kit learn" library function. The training dataset contains 51,300 samples which were further split into a 80-20% train-validation split using the same function. Therefore, the training dataset has 41,040 samples and the validation set has 10,260 samples. The dataset was classified into 30 unique labels as shown in figure 3.

B. Pre-Processing

1) Feature Data: The title column of the dataset contained the corpus of text required for the model to train. The initial view at the word corpus of the dataset is shown using "WordCloud" in figure 4.
The word cloud highlights the top 1000 words in the dataset, the bigger the word, the more frequent it is in the dataset. It can be observed that words like 'to', 'of', 'vol', 'and', 'for', etc have no or little significance in classifying the books according to its genre. Therefore, the text data needs to be processed with different NLP techniques to convert it into a suitable form for the deep learning model. The different text processing techniques of NLP include cleaning, stemming, lemmatization, tokenization.

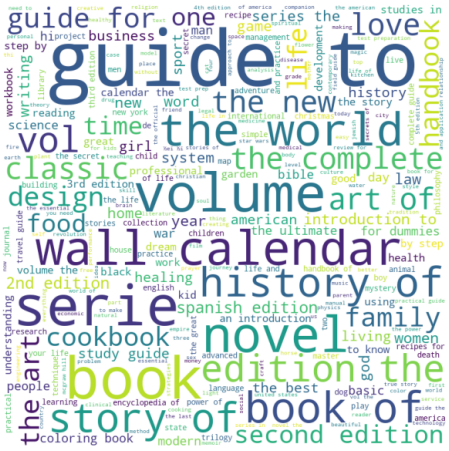

Fig. 4. Before Pre-processing

- Cleaning:
  "NLTK" and "Regex" libraries of python are used to clean the dataset. The punctuation is removed from the dataset using regex string. Everything except alphabets, both lower and upper case, is removed from each title. Each title is split into words. All the English language stopwords like and, the, for, etc are downloaded from the 'NLTK' package and stored in a set. Any word which is present in the set is removed from the title. Therefore, stopwords are removed from the dataset.
  "
- Stemming:
  Stemming is a kind of normalization technique. The words with similar meaning but with little variation as per the context or the title are normalized. For example word like guide, guides, guiding is converted to one single word like guide which

represents all the 3 variations. Therefore, stemming helps in removing the different variations that create ambiguity while training the model and during prediction. "NLTK" python library is used to implement stemming.

- Lemmatization:
  Depending on the meaning of the words, the lemma of the word is found through Lemmatization techniques. "NLTK" library is used to implement lemmatization. Morphological analysis of the words is done to remove the inflectional endings [6]. This results in the base words called lemma. WorldNet's built-in morph function is used by NLTK lemmatization method.
  Lemmatization is a better approach than stemming as it considers the morphological analysis of the word. Stemming results in the removal of the suffix from the word [4]. However, lemmatization results in generating dictionary words for different variations of the word. Therefore, the meaning of the word is retained by using lemmatization. It is a more intelligent operation and thus yields better features. The comparison of the vocabulary generated by Lemmatization and Stemming can be seen in figure 5

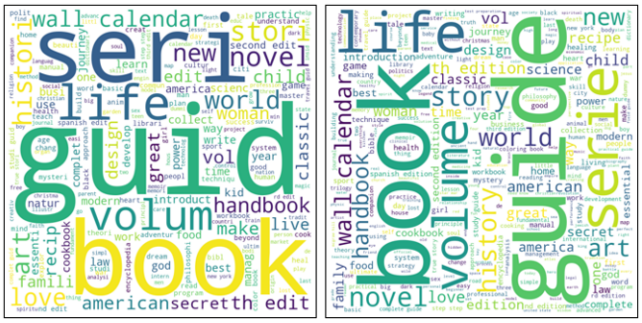

Fig. 5. After Pre-processing

- Tokenization
  The deep learning model accepts input in the form of numeric numbers. However, the dataset is in the form of text. Therefore, it needs to be converted into a numeric format. This procedure is called Tokenization. "Keras" library is used to perform tokenization. The tokenizer class of Keras is used to generate the vocabulary or dictionary of all unique words of the dataset. The words that occur more frequently are given less number than the words which occur less frequently in the dictionary. A parameter 'num_words' is specified inside the Tokenizer object. It is equivalent to the vocabulary size required and it needs to be configured. Initially, None is assigned to the 'num_words' parameter to estimate the exact number of unique words in vocabulary of the dataset. The top-most words less than the length of the word_index (the vocabulary of the dataset) are selected by the hit and trial method. An out of vocabulary token 'oov' is also used to represent all the words that are in the titles but not in the vocabulary of the dataset. The dataset is fitted with the tokenizer object for the top words using 'fit_on_text' and 'text_to_sequences' functions. This results in generating sequences of integers for each title. Each word in the title is represented by an integer. Same words in different titles are represented with a single value.
  The deep learning model is initiated with an Embedding layer that accepts the input in equal length. The 'pad_sequences' function of Keras is used to pad the sequences generated. The maximum length of a title in the dataset is 44. After analyzing the dataset the 'max_length' parameter of the function is given a value between [8-10] as this length would contain all the significant information of the title. 'Post' padding is selected as after exploring the dataset, some titles contain author, volume, and repetitions of words after the main title which needs to be removed. The padded sequences generated, are passed to the model to be trained.

C. Labels

The labels are converted to one-hot encodings using the 'to_categorical' function of the 'Keras' library in python.

D. Model Design

Different model architectures are created each with embedding layer in the beginning. The approach discussed in the paper compares the performance different embedding techniques in performing text classification. The different architectures tried are Convolutional Neural Network, Long Short-Term memory, simple neural network. The detailed architectures are mentioned in the V section.

E. Model Testing

The deep learning model testing is done through the test dataset which contains 5700 testing samples. The

dataset is processed in the exact same way as the training samples are processed. The processed test samples are evaluated using the Keras functionalities and the testing accuracy is determined.

F. Application Testing

The testing procedure of the application is depicted in the flowchart in figure 2. A desktop application is developed that allows users to capture a book cover image and reveal its genre.

1) Optical Character Recognition: The Optical Character Recognition(OCR) module is implemented using the "Pytesseract" library in python. The book cover is pre-processed first through the following steps.
1. The image is converted to 224*224 shape.
2. Grayscale conversion is used to reduce the depth of the image.
3. Thresholding is used to identify and isolate the features.

The processed cover image is passed to the pytesseract function configured with english alphabets to extract the title of the image.

2) Pre-Processing: Techniques similar to the preprocessing of training samples are used on the extracted title as well. The processed title is passed to the trained model to predict its genre. Therefore, the application accepts a single image at a time and predicts the genre that best suits the book title among the 30 different genres on which the model is trained.

G. Loss function and Performance Evaluation Metrics

Since the dataset has multiple categories, categorical cross-entropy loss function along with softmax activation in the dense layer is used to evaluate the loss required for the Backpropagation process. 'Adam' optimizer is used as it adaptively updates the learning rate throughout the training process and contains the goodness of both the root mean square propagation and stochastic gradient descend optimizers.
To evaluate the performance, accuracy as a metric is used. The higher the accuracy the better the model.

V. EXPERIMENTS

Several models were designed to perform the task of classification. Each model was initialized with its first layer as the Embedding layer to generate the Embedding matrix that holds the real-value representation of the input text data. Different ways of obtaining the embedding matrix are implemented:
1) Through trainable embedding layer,
2) Pre-trained GloVe Embeddings for 3 different dimensions [100,200,300],
3) Training word embeddings using Word2Vec using Gensim Python library.
The models are developed using Keras library in python. The figure 6 represents the different model designs used to perform the classification. The accuracy obtained with each model along with the model's configurations is mentioned in figure 7. Figure 7 contains multiple rows for models labelled in the figure 6. For example, 5 rows of the 2nd model containing a single LSTM layer with different embedding configurations can be seen. The figure 7 also reflects the test accuracy of the model on the testing data.

Model	Layers
1	Embedding, LSTM(50), Flatten, Dense(30)
2	Embedding, LSTM(50), Dropout(0.2), Flatten, Dense(30)
3	Embedding, LSTM(32), Dropout(0.2), Flatten, Dense(30)
4	Embedding, LSTM(32), Dropout(0.2), LSTM(32), Dropout(0.2), Flatten, Dense(30)
5	Embedding, LSTM(32), Dropout(0.2), LSTM(16), Dropout(0.2), Flatten, Dense(30)
6	Embedding, Conv(128,5,relu), MaxPooling(2), Flatten(), Dropout(0.2), Dense(30)
7	Embedding, Conv(64,5,relu), MaxPooling(2), Flatten(), Dropout(0.2), Dense(30)
8	Embedding, Conv(128,5,relu), GlobalMaxPooling(), Dense(30)
9	Embedding, GRU(128), Dropout(0.2), Dense(30)

Fig. 6. Model Designs

The last layer of each model is a Dense Layer with 30 neurons to predict 30 different genres. Softmax activation was used in the last layer with a combination of Categorical Crossentropy Loss (Softmax Loss). Vocabulary size of the top 16,000 words out of 29,811 unique words identified by the Tokenizer was selected as it resulted in the sequences of titles with fewer occurrences of out of vocabulary token. Lemmatization was preferred than Stemming as it results in increasing the overall accuracy by approximately 10%. The best model according to figure 7 was the 2nd model with a single LSTM Layer of 50 units. The training process of the best model over 10 epochs is shown in figure 8.

For testing the application, the OCR module experimented with different threshold values shown in figures 9 and 10.

A threshold value of 110 was chosen as it resulted in obtaining efficient titles. However, a major drawback of this approach is that it is light sensitive. The threshold

Model	Embedding Layer	Embedding dimension	Input Sequence Length	Training Parameters	Testing Accuracy
1	Simple	100	8	1,631,730	40.27%
1	GloVe	100	10	71,730	49.93%
2	Simple	100	8	1,631,730	43.12%
2	Simple	200	8	3,251,730	44.17%
2	GloVe	100	8	31,730	48.24%
2	GloVe	200	10	51,730	50.11%
2	**GloVe**	**300**	**10**	**71,730**	**52.53%**
2	Word2Vec	300	10	225,030	47.52%
3	GloVe	100	10	31,730	48.74%
3	GloVe	200	10	30,815	48.53%
3	GloVe	300	10	43,614	49.71%
3	Word2Vec	300	10	43,614	42.24%
4	GloVe	300	10	51,934	41.21%
4	Word2Vec	300	10	51,934	40.35%
5	GloVe	300	10	46,276	33.28%
5	Word2Vec	300	10	46,270	40.88%
6	GloVe	300	10	130,718	38.24%
6	Word2Vec	300	10	157,598	42.14%
7	GloVe	300	10	65,374	45.55%
7	Word2Vec	300	10	78,814	43.44%
8	GloVe	300	10	67,998	43.88%
8	Word2Vec	300	10	195,998	47.32%
9	Word2Vec	300	10	168,990	43.44%

Fig. 7. Model Results

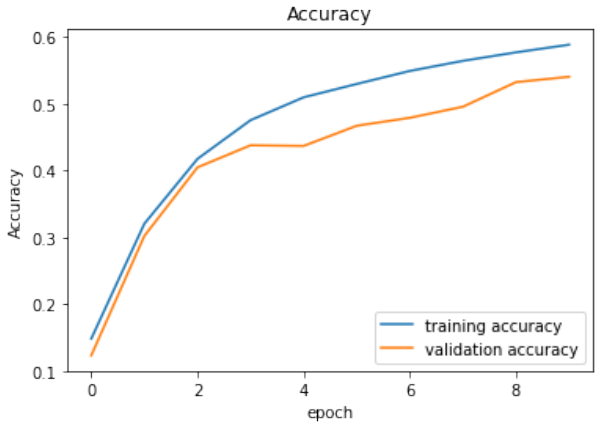

Fig. 8. Training Process of the best model

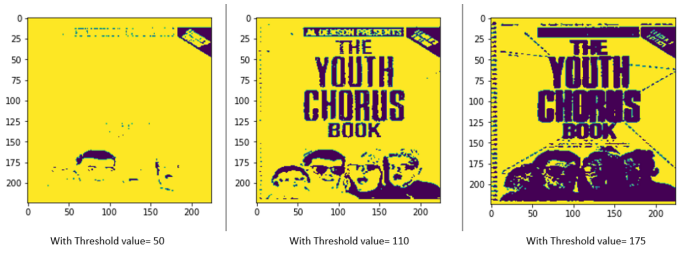

Fig. 9. Cover Image Processing

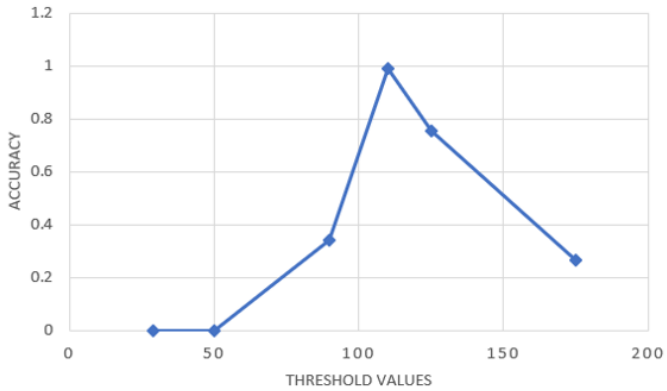

Fig. 10. Accuracy of obtaining the exact title w.r.t different threshold values.

values need to be updated according to the HSV (hue-saturation-value) value of the image. Out of 140 cover images selected randomly from the test dataset, the OCR approach was able to detect 64 titles correctly giving an accuracy of approximately 46%. More accurate extraction of title results in better genre classification accuracy than a less accurate extraction of title.

VI. ANALYSIS OF RESULTS

The CPU time required by the Gensim's Word2Vec model to generate the word embeddings was significantly less than the time needed to obtain the embedding matrix using the Embedding layer. A lower dimension embedding matrix was not able to generate significant results as titles are small sentences with unique words and require more features to represent each word and thus, classify the titles into 30 unique genres.

GloVe allows parallel implementation and thus, it is easier to train over more data as compared to Word2Vec. Unlike Word2Vec, GloVe uses global statistics i.e. word co-occurrences to obtain the vector representations of each word and does not depend only on local context information of the words [10]. GloVe embeddings focus on capturing the sub-linear relationship in the vector space and eventually performs better than the Word2Vec model in tasks that involve word analogy like classifying the book titles. The generated model using GloVe embedding requires more storage space than the Word2Vec model.

Recurrent Neural Networks(RNN), is another neural network where connections between the nodes form a directed graph along the sequences [8]. Therefore, it

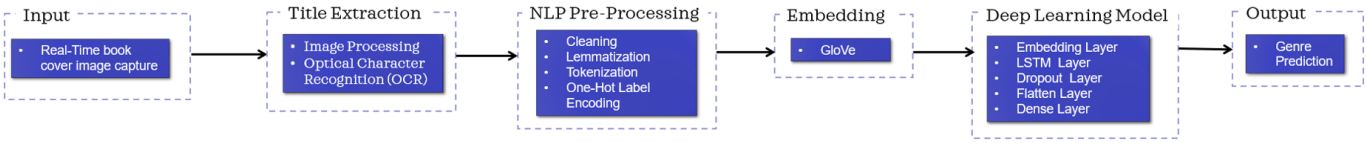

Fig. 11. Complete Application Flow

is a preferred model for text processing. Long Short term Memory(LSTM) and Gated Recurrent Unit(GRU) mitigates short term memory using gates that regulate the flow of information flowing through the sequence chains [8]. The forward LSTM layers in the model encode the features extracted from the last output of the RNN before passing it to the feed-forward network for classification. The GRU extracts the contextual semantics between the embeddings of words from the previous layer and use it for classification. GRU models are faster to train than the LSTM model. However, the accuracy obtained with LSTM is more than the GRU architecture for this dataset. As LSTM contains more gates than GRU, the LSTM networks are more stable and thus give high accuracy.

Convolutional Neural Network (CNN) is a type of deep feed-forward network. The usage of CNN on text data has yielded promising results in NLP. The neurons of each convolution layer fire when a special pattern is detected. Variable sizes of kernels and the concatenation of their results generates patterns/sequences of variable sizes [8]. Thus, CNN is able to identify sequences irrespective of the position.

Although CNN gives a more stable result, however, it fails to predict the genres accurately as compared to RNN. Since the dataset has a large amount of data, RNN is a preferred approach to work. CNN requires less training time as compared to RNN. Moreover, the CNN models start to overfit after 3-4 epochs of the training process. Therefore, the RNN model is selected as it gave the highest accuracy with less overfitting.

The model hyper-parameters were altered with hit and trial method to obtain better accuracy. A batch size of 128 and 10 epochs was considered as the best configurations.

A major issue observed throughout the experiments was the Overfitting of the model. The model was very likely to get over-fit after few 10-20 epochs. The validation loss exceeded the training loss for various models. Regularization with the use of Dropout layers was used. However, it failed to regulate the model after some extent. The variation between the trainable parameters of each model and the number of input features was observed. The high variation between both resulted in poor results. Therefore, the layers were modeled to produce less trainable parameters. This is another reason why GloVe implementations gave better accuracy.

The complete application procedure is presented in the figure 11

VII. DISCUSSION

The human accuracy to predict the genres of the book from its titles is around 75% as specified in paper [7]. Our implementation using GloVe embeddings and LSTM network gave an accuracy of 52.53%. The main reason behind the failure to beat human accuracy was the overfitting of the model. Irrespective of regularization the validation loss remained higher than the training loss. The overlapping of the 30 unique genres in the dataset may be a reason for this overfitting. For example, 'Religion & Spirituality' books title are quite similar to the title of 'Bibles'. Similarly, 'Comic' book titles are quite similar to 'Children Book' titles. Thus, this may lead to confusion in predicting the genres due to the ambiguity in the categories. This problem could be resolved by implementing a Multi-class classification where each title could be labeled under multiple genres. Although the paper focuses on the implementation of only a deep learning model, several machine learning models like Support Vector Machine, Decision tree, Random forest, etc could be tried out to compare the results. The analysis highlighted the significance of using GloVe embeddings to achieve higher accuracy as compared to the other approaches discussed in the paper. The light dependency to extract the title through the OCR module could be reduced by using Adaptive thresholding and retraining the Tesseract model available [1] as open-source with variable fonts, styles of English alphabets. A deep learning model could also be implemented to identify and detect the titles from the cover images to improve the detection accuracy.

VIII. FUTURE WORK

The application of extracting the title from the cover image and predicting the genre of the book could be improved by incorporating Reinforcement Learning. Taking valuable feed-backs or reinforcements from the user when the model fails to predict the correct genres would be useful in improving the model's precision and accuracy. Additional functionality of text-to-speech can be incorporated in the application to make it handy for physically challenged people as well.

IX. CONCLUSION

The paper demonstrated an approach to develop a complete application that can capture real-time book cover images, extract the book title, and predict its genre. Although the results were not overwhelming in comparison with the human accuracy of 75% to predict the genres, the best model selected gave an accuracy of 52.23%. Overfitting due to ambiguous genre categories was concluded as the main reason for obtaining the low accuracy. Moreover, as per the analysis of the experiments performed, GloVe embeddings outperformed in generating an efficient word embedding to represent the titles of the book in comparison to Word2Vec and the trainable Embedding layer. According to the accuracies achieved, Recurrent Neural Networks performed well in comparison to the Convolutional Neural networks in classifying the book titles.

REFERENCES

[1] Tesseract. https://tesseract-ocr.github.io/. [Online; accessed 18-June-2020].

[2] S. Ahmedz A. Dengelzx & S. Uchida B. Kenji Iwana, S. T. R. Rizvizx. Judging a book by its cover. 2016. [Online; accessed 10-June-2020].

[3] J. Brownlee. Word Embeddings for Text. https://machinelearningmastery.com/what-are-word-embeddings/, 2017. [Online; accessed 14-August-2020].

[4] Stanford Education. Stemming and lemmatization. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html, 2009. [Online; accessed 18-June-2020].

[5] M. J. Garbade. A Simple Introduction to Natural Language Processing. https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32, 2018. [Online; accessed 13-August-2020].

[6] Guru99. Stemming and Lemmatization with Python NLTK. https://www.guru99.com/stemming-lemmatization-python-nltk.html, 2020. [Online; accessed 12-August-2020].

[7] Y. Ge & C. Wu H. Chiang. Classification of book genres by cover and title. 2019. [Online; accessed 10-June-2020].

[8] A. Maheshwari. Report on Text Classification using CNN, RNN & HAN. https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f, 2018. [Online; accessed 12-August-2020].

[9] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. *Int. Soc. of Music Inform. Retrieval,*, vol.Citeseer, 2004, pp. 525–530, 2004.

[10] Sciforce. A Simple Introduction to Natural Language Processing. https://medium.com/sciforce/word-vectors-in-natural-language-processing-global-vectors-glove-51339db8~:text=Compared%20to%20word2vec%2C%20GloVe%20allows,methods%20exploiting%20global%20statistical%20information, 2018. [Online; accessed 11-August-2020].

[11] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech Audio Process.,*, vol. 10, no. 5, pp. 293– 302, 2002.

[12] Wikipedia. Word Embedding. https://en.wikipedia.org/wiki/Word2vec#:~:text=Word2vec%20is%20a%20technique%20for,words%20for%20a%20partial%20sentence., 2020. [Online; accessed 13-August-2020].