Using Recurrent Neural Networks in Movie Classifications

Katherine Fu jfu57@wisc.edu
Yuting Yan yan86@wisc.edu
Hangyu Kang hkang98@wisc.edu

Abstract

These days, with the development of the movie industries, it is extremely convenient and essential for people to select a specific genre to watch from the plot provided by the producers, and we wish to provide such an model. In our project, we performed supervised training since the plots and labels for each movie are provided in the data set. First, we employ the word2vec algorithm to create word embeddings as distributed representations of words in a vector space for the whole movie plots after data preprocessing. Different word embeddings of the movie plot generated by word2vec, TF-IDF weighted word2vec, and doc2vec are trained by KNN classifier. The word2vec embeddings are applied as the weights of embedding layer for the deep learning model. The RNN model architecture is consisted of bi-direction LSTM system and attention mechanism to process sequential text data. We also apply different algorithms including CNN, RNN-CNN, ensemble of the three deep learning model, and machine learning and compared their results with RNN using various evaluation means. The results show that RNN gives the best performance in accuracy, F1 score, and recall metric. The SGD optimizer and step learning rate decay are employed and the binary cross-entropy was used to calculate the loss between predicted values and real values. By inputting the sequences of text plots, our model is robust and capable enough of successfully classifying movies into different categories.

1. Introduction

In the modern movie selection system, due to the high volume of movies, movies recommendations based on their genres are extremely essential. Humans can potentially name the genre by simply watching the entire movie. To simplify the process, we proposed a model that simulates a human's watching process to predict a given movie's genre automatically with the help of movie plots. We obtained the frequencies of each word in the plots and utilized natural language process such as text classifications and different word embedding methods to get significant words and vectorize them. We fed the words into deep learning architectures including RNN, CNN, the hybrid RCNN, and the ensemble model to get genre of the movie and compared their performance. We also utilized the machine learning KNN method as our baseline model. The data set we employed is originally from the IMDb Datasets and has been roughly preprocessed by David S. Batista, a scholar in natural language processing and machine learning[1]. We simply downloaded the data matrix from his personal website and trained the mode using the first 33000 samples. The sequence of plot summaries of a given movie would be our source data and the movie genres are our output and will be used to test the model classification accuracy rate. The genres are already in the format of hot encoding, which gives me a lot more convenient in data processing step. By scanning the movie plots, the machines can categorize the movies according to their genre. The algorithm can make contributions to movie review applications and make people's life easier.

We make the following main contributions in this work: (i) We utilized the dataset with the movie plots with arbitrary lengths and the movie genres with unbalanced distributions. We mainly conducted data preparation in plot summaries. (ii) Being motivated by text classification methods that will be introduced in section 3[2], we applied different algorithms into classification. We finally found out the RNN along with bi-direction LSTM system and attention mechanism has the best performance. We know that bi-direction LSTM is traditionally utilized in speech recognition but we applied this technique here in text classification to improve the performance.[3] The main idea behind it is that each training sequence can capture both the backward and forward information at a certain time stamp. [4]

The rest of the paper is organized as follows. In Section 2, we will provide the related works as references. In Section 3, we will give a clear introduction of our proposed model. That includes all the architectures we have utilized. In Section 4, the entire experimenting processes will be given, which contains the data preparation and model conduction. In Section 5, we will present our final results and evaluations.

2. Related work

Numerous current researches have been done something related to the text classification. Jindal et. al.[5] wrote a review paper on techniques for text classification, where most of paper referred used machine learning methods. Kim[6] employed CNN in text classification firstly. CNN is good at extracting deeper information while setting the kernel size is still challenging. RNN is one of the most popular model to investigate text classification.

Recently studies have shown that a novel way of combining RNN with bi-directional LSTM and CNN together have given a greater result[7]. This hybrid method resolves the limitation of confined and undetermined kernel sizes in CNN as the bigger size will lead to more noise while the smaller one will loss information. It also helps equalize the emphasis on every word which could not be realized by simply applying RNN. Currently, ensemble model are widely used to improve performance no exception for text classification[8]. In these project, we were motivated to combine the above ideas with the attention mechanism and able to present a movie-classification model by extracting features only from the input sequence of text plots and provided the genres. We will present the experiment steps in applying RNN and CNN architectures separately as well as the RCNN architecture and the ensamble model for the three.

David S. Batista, the author of the dataset, used Naive Bayes, SVM, and linear Logistic Regression accompanyed with TF-IDF weighted word embedding to train, and obtained 0.78, 0.87, 0.84 test F1-score respectively. However the results for word2vec and dac2vec are 0.48 and 0.45 with LogisticRegression classifier. We use different classifier and deep learning methods and expect to obtain equal even better performance of his best result.

3. Proposed Method

3.1. Word Embeddings

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers[9]. The modules word2vec and doc2vec in standard libraries Gensim are implemented to embed words or documents in a lower-dimensional vector space using a shallow neural network. Different methods investigated to represent the document(in our database is the plot of each movie) feature in our project are averaged word2vec, TF-IDF weighted word2vec, and doc2vec.

Averaged Word2vec. Input text corpus as a one-hot form, the Word2Vec algorithm outputs a distributed representations of words in a vector space[10], as shown in the Figure 1. By training this network, a 10000 length(represents the total number of vocabulary) vector can be transformed into a to 300 length vectors. The averaged

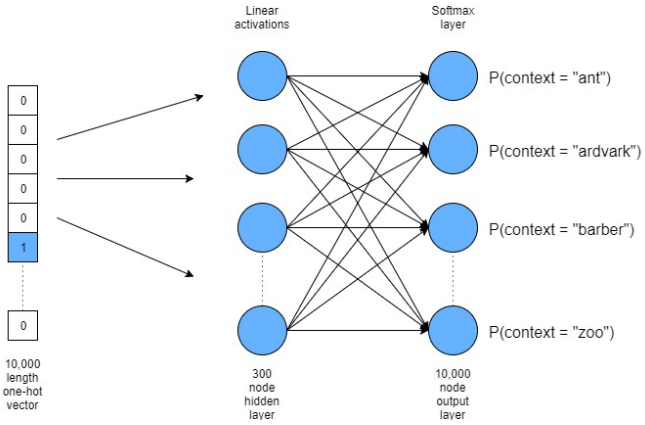

Figure 1. A trainer example for Word2Vec

word2vec which is a rather straightforward method, directly averages all word vectors occurred in the processed documents and outputs the embeddings matrix for all the plot items.

TF-IDF Weighted Word2vec. TF, Term Frequency, $TF_{ij} = \frac{n_{ij}}{sum_k n_{ij}}$, the frequency word i occurs in document j divided by the total number of the word in that document. IDF, Inverse Data Frequency, is the logarithmically scaled ratio of total number of documents in the corpus to the number of documents where the word appears. The tfidf value of each word is calculated by multiplying TF and IDF with the corresponding word. In our project, the vectors of each plot is obtained by the following: first multiply the tfidf value with the word vector, and then divide sum by sum of tfidf value for each document[11].

Doc2vec.Doc2vec, extension of Word2Vec, is the summary of all the word vectors in a document but fixed the problem of ignoring word ordering and semantics. [12]

3.2. Machine Learning Method

In this research, we also tried machine learning method as the comparative experiment. A k-nearest neighbor(KNN) is a supervised learning algorithm for classification. In the classifying situation we should choose k number of existing data point to refer. This is what k means in k-nearest neighbor. Hence, the tuning parameter for KNN can be the way to measure the distance between the new data point and existing data point; moreover, k is also can be a tuning parameter. However, when we use KNN algorithms for text based classification, there are many flaws such as calculation complexity and others. However, a suggestion by Zhou Yong, Li Youwen Xia Shixiong[13], using clustering method can improve original KNN method. In terms of our project, all of the words in each summary goes into categorized cluster. For example if one observations is catego-

rized as horror, then the whole words in summary goes into a horror cluster. The trained clusters can be used for the next classification.

The KNeighborsClassifier in sklearn liberary is used to train the movie plots classification. Different word embeddings including averaged word2vec, TF-IDF weighted word2vec, and doc2vec as input data are compared. Moreover, fine-tuning on number of neighbors finds the best number of neighbors is 3.

3.3. Embedding layer

The first layer of all our deep learning models is embedding layer, having the same function to Word2Vec. What is the difference with Word2Vec is the input of embedding with the index of all words while that for Word2Vec is one-hot matrix.

3.4. RNN Architecture

In our project, we applied Recurring neural network(RNN) as our main method. RNN is shown to possess high performance compared to other architectures in that it takes the sequencing patterns of the input into considerations. However, basic RNN has vanishing gradient issue, so we incorporated LSTM. In their paper, Hochreiter and Schmidhuber[14] proposed the long short-term memory network (LSTM) which is used largely in the deep field to help learn the long-term dependencies while avoiding the vanishing gradient problems. The LSTM architecture is performed as another hidden layer that contains a separate cell including an input gate, an output gate, and a forget gate. It could determine and control the things, in this case, the text vectors, to be remembered or forgotten.

The basic attention layer is applied in our model. First introduced in 2015 as a way for automatic English-French translation[15], this mechanism achieves high performance by selecting certain words that are essential for predicting the result output. Concerning the deficiencies in LSTM, such as gradient explosion or vanishing, shallow layers, etc, we add attention mechanism to LSTM.

We integrated the above architecture and the example networks are shown in Figure 2. Given the all the word indices of each movie plot, and set the weighted of embedding layer as our pre-trained word vectors. Since the Word2Vec is also a short trained neural network, we freeze the weight in the embedding layer. The input dimensions is the vocabulary size and embedding dimension with 300. In this model, two Bi-LSTM layers involving dropout are applied with a hidden size 256, and then concatenate the outputs of two layers as the input of next layer. The function is added to accurate running speed and reduce computing error. In order to improve the performance of LSTM, the attention mechanism is used. A two full connected layers with input dimension 2*256 and output 27 classes. After the dense layers, Sigmoid active function is connected to calculate the probabilities of all classes.

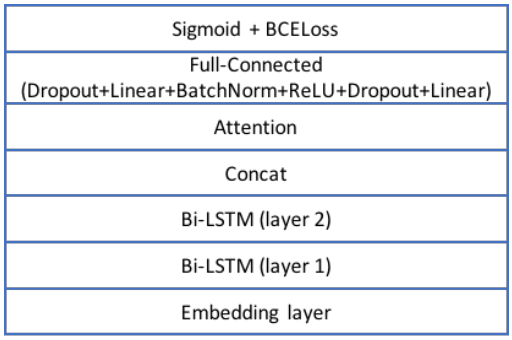

Figure 2. RNN model architecture

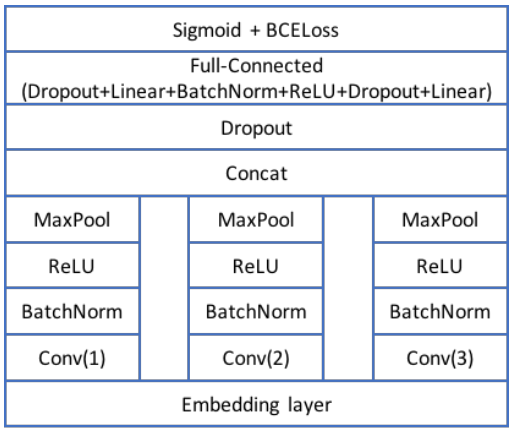

Figure 3. CNN model architecture

3.5. CNN Architecture

As more researcher are also using CNN to investigate text classification problems, we proposed the CNN architecture. In this project, a parallel CNN are implemented by a three dimensional kernel, show in Figure 3 where each dimension is used to conduct Conv1d once. In this model, the BathchNorm method is employed instead of the usual dropout. The outputs of each Conv1d are concatenated as the input of the following layers, which is the same to RNN.

3.6. RCNN Architecture

Some scholars also proposed Recurrent Convolutional Neural Network (RCNN) to overcome the deficient of CNN and RNN, sunch as difficulty in determining the kernel size and dominant later words than earlier words in RNN. Based on the CNN and RNN architecture about, we combine them to be a RCNN architecture in Figure 4. The RNN part is used to capture the contextual information to the greatest extent possible when learning word representations. and the

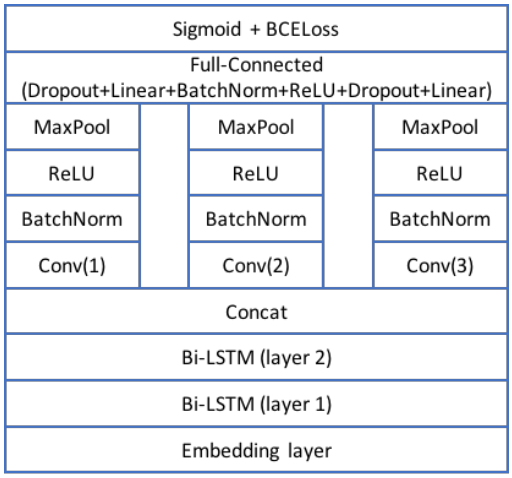

Figure 4. RCNN model architecture

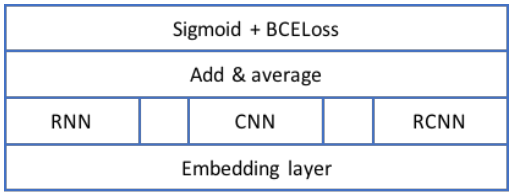

Figure 5. Ensemble model architecture

CNN part is applied to extract more features of text.

3.7. Ensemble Model

Ensemble learning is all about best combining the predictions from multiple existing models in some specific method. Generally, this technique is more suitable for those neural networks with high variance. In the project, we add and average the probabilities of three pre-trained model and obtain the accuracy and F1-feature. The model architecture is shown in Figure 5.

4. Experiments

This section describes the steps we performed through the entire training. We will first introduce the data preprocessing step and then the training process and the final results will be presented.

4.1. Dataset

As mentioned above, the data set we employed is from the IMDb Datasets. This data set originally contains 117194 data samples with features of ”title”, “plot”, and the hot encoding for 30 labels. “Plot” contains the summaries of a given movie and it would be the input of our model. Genre labels are the output and will be used to train the RNN model. Figure 6 gives some examples of dataset items.

	title	plot	Action	Adult	Adventure
0	"Ancient Impossible" (2014) {Greatest Ships (#...	Some of the greatest ships in history were ac...	0	0	0
1	"Buffy the Vampire Slayer" (1997) {First Date ...	Buffy accepts an invitation to dinner from Pr...	1	0	0
2	"Aviation Storytellers" (2013) {The Tail Gunne...	Take a journey on the B-17 "Flying Fortress" ...	0	0	0
3	"A Year In: Outings and Alternate Perspectives...	Comprised of footage taken from real life, do...	0	0	1

Figure 6. Examples for dataset items

In our pre-processing processes, we conducted data preparation in the summaries to obtain a clean input matrix for our RNN model. Figure 7 describes data preparation for summaries and the subsection 5 is for labels.

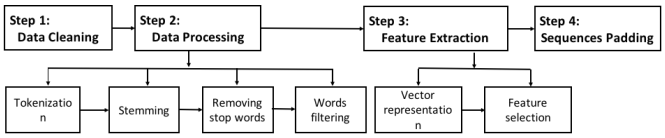

Figure 7. Data preparing process

4.1.1 Data Cleaning

In data cleaning processes, we simply removed the data samples with empty labels and those with the number of words in summary less than 9.

4.1.2 Data Preprocessing

In this section, we performed four steps: (i) Tokenization basically means separating each word from the long summary sentences. (ii) Stemming refers to the removal of unnecessary symbols such as ”$” or ”{}”, etc. (iii) removing stopwords involves deleting ”a”, ”an”, ”the”, etc. (iv) Word filtering applies “GoogleNews-vectors-negative300” as the pre-trained model to select words that have appeared in this dataset.

4.1.3 Feature Extraction

We performed the vector representation which maps vectors with numeric values to the textual data, using the index of the word in the whole text as representations. Then, we did feature selection process where we obtained the encoded words with vectorized representations that capture some of relation and similarity between words within the text corpus, using pretrained model GoogleNew described in the above Word Embeddings section as the benchmark of the each word.

4.1.4 Sequence Padding

Since our input text sequences vary in their lengths, we added padding to the vector matrices with lengths less than 174. This is the maximum length of plots among all the data samples. To perform this step, we simply conducted the right-pad with 0s.

4.1.5 Labels Preprocessing

In this dataset, the hot encoding for labels are provided so we did not need to do preparation for the labels. The number of movies per genre are shown in Figure 8

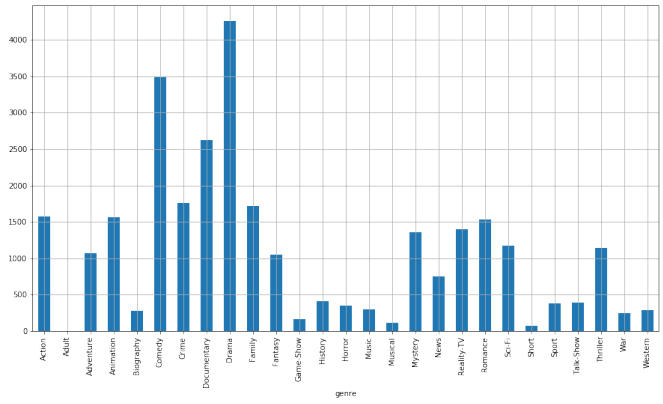

Figure 8. Number of movies per genre

4.1.6 Data Splitting

We first shuffled our data set and split it into the training, validation, and test data set with the percentage 70%, 15%, and 15% respectively. The purpose of this step is to reduce the potential of over-fitting that usually happens when exclusively applying the training data set.

4.2. Software and Hardware

We mainly utilized PyTorch, Numpy, and Spacy from Python to write all the algorithms and Jupyter notebook and terminal to run the scripts. Google cloud and Paperspace also helped to support running our large dataset. All the scripts can be found on our project GitHub.

All the running processes were done either on our personal laptops or the GPU cloud listed above. We have also utilized Google scholars, Analytics Vidhya, Mediums, and other related deep learning online sources to learn more about various models and methods.

4.3. Deep Learning Training Parameters

Hyperparameters in our project are given in Table1. We choose SGD as optimizer because some programmer found that although Adam is more advanced and higher speed training, SGD is more stable and convenient to control overfitting. The parameters of learning rate scheduler are determined by numerous trials from 0.0001 to 1. Weight decay in optimizer is similar to regularization, which provide an approach to reduce the model overfitting. For the input of the full connected, it depends the output of the former layers.

Table 1. Hyperparameters setting

	Parameters	Value
Optimizer	optimizer	SGD
	weight decay scheduler	$7 \times 10^{(-4)}$
	momentum	0.9
Learning rate	initial value	0.1
	decay scheduler	StepLR
	stepsize	15
	gamma	0.5
Embedding layer	vocabulary size	45350
	embedding dimension	300
CNN	in channels	300
	out channels	100
	kernel sizes	[3, 4, 5]
	MaxPool kernel sizes	[608, 607, 606]
LSTM	input size	300
	hidden size	256
	num layers	2
Dense layer	hidden size	1000
Batch size		0.5
Dropout rate		128
Number of epoch		40

4.4. Loss Function

To measure the quality of our approximation, we have choose a loss function for the model, binary cross entropy(BCELoss). It is one of the most common loss function in classification problem even it can deal with multi-label classification problem. However, for our project, each observation possibly has more than two labels, so it is impossible to use multi-label cross entropy function. For this reason, we need to binarize our outcome. The way to binarize our outcome is simple. We just need to do one-hot encoding that shows predicted value and the original value is the same. If they are the same it returns 1; otherwise, it returns 0.

$$L_{binary} = -\sum_{i=1}^{n}(y^{[i]}\log(a^{[i]}) + (1-y^{[i]})\log(1-a^{[i]})) \quad (1)$$

where $a^{[i]}$ is a logistic sigmoid active function outputs, the possibility that model predicted correctly.

4.5. Evaluation

Here, we first give definitions for the true positive(TP), false positive (FP), true negative (TN), or false negative (FN) in Figure 9[16], as the indicators for our model.

accuracy, and F-measure are applied to evaluate the performance of model: ,

Accuracy. Accuracy is defined ad the ratio of movies that are correctly classified compared to the whole set. Below is the euqation:

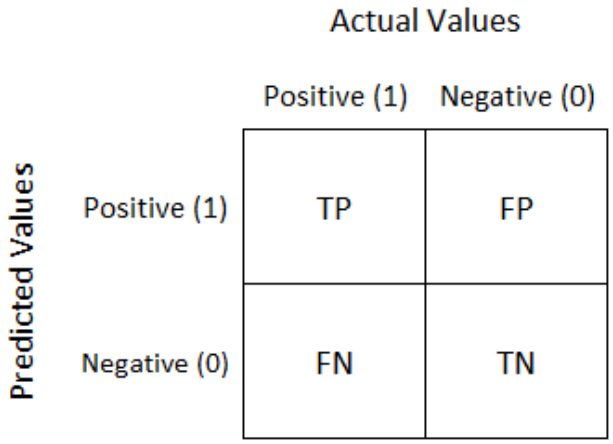

Figure 9. Confusion Matrix

$$\text{Accuracy} = \frac{TN + TP}{TP + FP + TN + FN} \tag{2}$$

F-measure. F1 score is widely used for measuring evaluation in information retrieval and machine learning. It can be used to evaluate the unbalanced data. To calculate the F1 score we should know Precision and Recall. The Precision is the ratio between the number of actual true value and the number of predicted true value. The Recall is the ratio between the number of predicted true value and the number of actual true value. It is the harmonic average of Precision and Recall. Precision, recall, and corresponding F-measure can be computed as follow[17]:

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \\ \text{F-measure} &= \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{3}$$

5. Results and Discussion

Considering that the number whole dataset is too large so it will consume several hours to run the whole code. Thus, we only chose 33000 items.

5.1. Results of RNN model

First, we will present the result by conducting the RNN model. To identify the influence of word embeddings, we set the different weights for embedding layer given by "GoogleNews-vectors-negative300" and results of custom-trained word2vec model to train the neural networks. The results in Table 2 shows little difference between them but custom-trained word2vec has slight higher F1-score. Therefore, we used the self-trained Word2Vec for further training.

As we can see in the table, all the accuracy are much higher than the precision. That is because our targets matrix is a sparse one, the accuracy defined is based on the whole set while the precision is based on the true labels.

Table 2. Comparison different weights setting in embedding layer

Model	Accuracy	Precision	Recall	F1-score
GoogleNews	94%	0.756	0.463	0.613
Word2Vec	95%	0.775	0.586	0.667

Figure 10 gives the F1-score of all classes testing F1-score. Compared the Figure 10 and Figure 8, we found out an inverse relationship between the number of movies and the testing F1-score for each category. The "Drama" has the lowest F1-score values, with detail information in the confusion matrix presented in Figure 11.

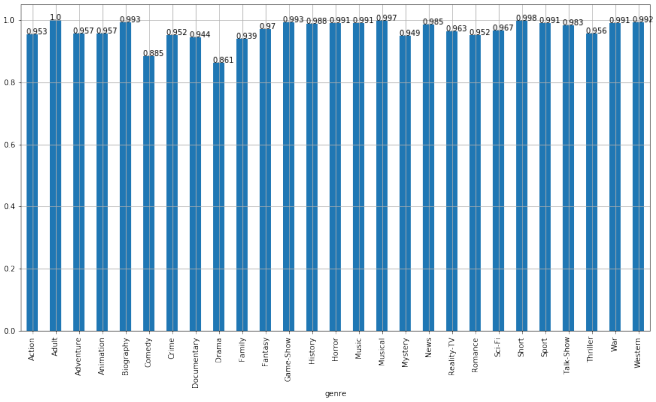

Figure 10. Testing F1-score for all classes

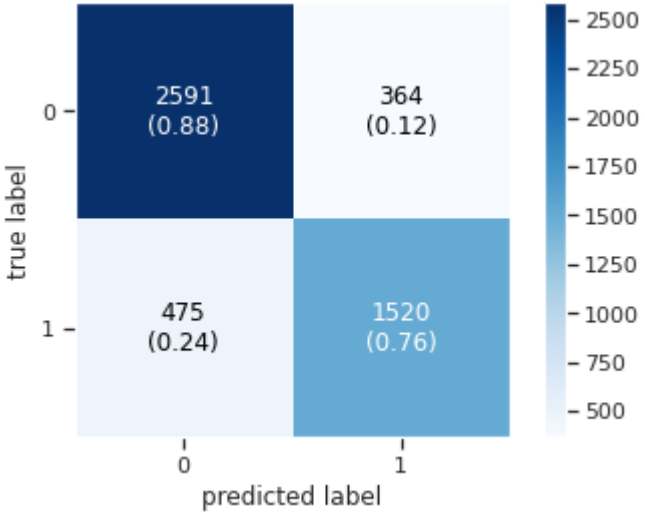

Figure 11. Confusion matrix for class "Drama"

During the trails on RNN model construction, we found out that both the batch normalization and dropout mecha-

nism can tackle the overfitting problem but at the same time, they both have an inverse impact on the accuracy. Adding the batch normalization increases accuracy without other observable effects, while adding dropout decreases. However, the models with bath normalization are more time-consuming than dropout. Moreover, adding attention mechanism also decreases the difference between train accuracy and valid accuracy as well as increasing test accuracy slightly.

5.2. Comparison of Different Models

By Applying the baseline and deep learning methods, we presented the results with accuracy and F1-score in Table 3 and 4. All the results are testing evaluation outputs.

Table 3. Results of different doc embeddings for KNN

Model	Accuracy	Precision	Recall	F1-score
Average	93%	0.580	0.570	0.575
TF-IDF	94%	0.709	0.581	0.638
Doc2Vec	92%	0.510	0.315	0.390

Table 4. Results of deep learning methods

Model	Accuracy	Precision	Recall	F1-score
CNN	94%	0.646	0.548	0.593
RNN	95%	0.775	0.586	0.667
RCNN	95%	0.788	0.527	0.632
Ensemble	94%	0.804	0.438	0.567

The KNN classifier with different embedding methods gives comparable results with that of deep learning. The results turn out that TF-IDF weighted custom-trained word embedding has the best performance, while on the contrary, custom-trained Doc2vec perform much worse. The doc2vec vectors were generated from gensim library, which is like a black box to us. A possible way to improve the performance is the parameter tunning on vector generation. Also, TF-IDF vectors with up to 5000 dimension compared to 300 for word2vec and doc2vec may be achieve with non-linear kernel.

Although many researchers stated the RCNN and model ensemble place higher performance than CNN or RNN, the results in our project can't get such conclusion. Compared with F1-scores of all methods, the performance of RNN is slightly superior to others. But for precision, the stacked ensemble is the best. Since we spend most time in tunning the RNN model, this maybe one of the reason for best performance in all the models. The train time order is Ensemble>RCNN>RNN>CNN. One of reason we think the added attention layer might play a significant role in the RNN model. We also compared the CNN to the RCNN and found out that the RCNN outperforms the CNN architecture. That's because the recurrent structure in the RCNN captures contextual information better than window-based structure in CNNs.

Unfortunately, we don't achieve the highest F1-score values of Document Classification. After discussion, we speculate some reasons: 1)TF-DIF weighted words can be better in representing the word embeddings and have much higher dimension than word2vec; 2)our model hyperparameters haven't be tunned to reach the best performance; 3)only part of the dataset rather than the whole data are used in our model; 4) David chose 1/3 of the dataset as test data but we only have 15%, and there is overfitting in our model.

6. Conclusion

In conclusion, RNN has the best performance among all the algorithms. It has 95% accuracy, 0.77 precision, 0.58 recall, and 0.667 in F1 test score. In terms of multiple embedding with different weights set up when applying RNN, we also found out the difference in the results. The self-trained Word2Vec shows good performance and is used in further training.

Mentioned above in section 5, we are aware that the good performance of RNN might be due to the tuning variables we chose. We can find the performance results among all methods do not show wide difference. Thus, if we are given more time, it is possible for us to give totally different results. However, since we can still achieve around 95% accuracy in our testing data, we could conclude that our model is robust enough in the classification task.

7. Future Direction

Our deep learning model have only provided a basis for classifying the movies into different categories based on the unfixed length of input sequences. More works can be tried such as date argumentation by dropping some word and randomly shuffling the order of words, especially for those classes with less movies. Fine-tuning on the embedding layer and bootstrap for loss maybe valid to improve performance. In the future, We need to fine tune our model hyper parameters to improve the test accuracy and potentially apply the model into classifying more given genres. More importantly, we want to generalize the model into classifying the book genres as well as other potential applications related to classifications. We also want to explore different methods and compare each performance and select the best one. Furthermore, we wish to provide open-source software for the deep learning researchers.

8. Acknowledgement

We want to show our greatest thanks to our professor Sebastian Raschka for his feedback and understandings. At the beginning, we had hard time training our model using

book summary dataset provided by Carnegie Mellon University since the book labels were extremely unbalanced and the number was too large. The results we got from training this dataset were unpromising. Professor Raschka allowed us to switch to movie dataset which we are able to present a decent model that classifies given movies.

9. Contributions

The current situation prevented our group from meeting face to face but we were able to meet remotely through zoom bi-weekly and have close contacts on group chats. Our researching process was not easy and had met some obstacles but all of the group members were able to work together and contribute to the final project.

In the project, Katherine was mainly responsible for text pre-processing, text transformation, and word extractions. Yuting was mainly in charge of models construction and the parameter validation. HanGyu was responsible for utilizing testing and validation dataset to calculate classification accuracy rate illustrated above.

References

[1] David S. Batista. Movie dataset.

[2] Liu Gang and Jiabao Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. 01 2019.

[3] M. Wöllmer, Z. Zhang, F. Weninger, B. Schuller, and G. Rigoll. Feature enhancement by bidirectional lstm networks for conversational speech recognition in highly non-stationary noise. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6822–6826, 2013.

[4] Marco Basaldella, Elisa Antolli, Giuseppe Serra, and Carlo Tasso. Bidirectional lstm recurrent neural network for keyphrase extraction. In Giuseppe Serra and Carlo Tasso, editors, *Digital Libraries and Multimedia Archives*, pages 180–187, Cham, 2018. Springer International Publishing.

[5] Rajni Jindal, Ruchika Malhotra, and Abha Jain. Techniques for text classification: Literature review and current trends. *webology*, 12(2), 2015.

[6] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[7] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification.

[8] Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, 57:232–247, 2016.

[9] Wikipedia contributors. Word embedding, 2020. [Online; accessed 1-May-2020].

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[11] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE, 2015.

[12] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[13] Yong Zhou, Youwen Li, and Shixiong Xia. An improved knn text classification algorithm based on clustering, 03 2009.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Yoshua Bengio Dzmitry Bahdanau, Kyunghyun Cho. Neural machine translation by jointly learning to align and translate. 2015.

[16] M Ikonomakis, Sotiris Kotsiantis, and V Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

[17] Mostafa Sayed, Rashed Salem, and Ayman E Khedr. Accuracy evaluation of arabic text classification. In *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, pages 365–370. IEEE, 2017.