CPSC 585 Project 2 Reviews

Group 9
Devansh Sharma
Bike Qian
Uma Maddukuri
Michael Collins
Brijesh Prajapati
Parth Nilkanthrai Barot

Introduction

This review aims to compare and highlight various similarities and differences observed when comparing different approaches by groups 9, 12, and 13. The task for all groups was to predict the quality and difficulty scores a student would assign based on the text of the student's comments using a recurrent neural network model.

Approach Implemented by our Group

We started off with importing the necessary libraries. We loaded the data and extracted columns namely, ‘diff_index’, ‘comments', 'star_rating', 'student_star', 'student_difficult’ into the dataframe ‘data’. Next, to avoid converting any null values into features, we dropped them from our dataframe. Then, we grouped the data into input variables and target labels. The ‘comments’ column was taken as input features and the columns ‘student_star’ and ‘student_difficult’ were taken as the target labels ‘*y_ss’* and ‘*y_sd’*. As a part of text preprocessing, we first converted the ‘comments’ into string and then performed two-step preprocessing that included lowercasing and removing punctuations. Although TextVectorization implicitly performs both the above mentioned preprocessing methods, but since we used Tokenizer library, we had to explicitly perform data preprocessing at the beginning itself. Moreover, we did not implement any other pre-processing methods because they were hurting the model performance afterwards while we were carrying out our experiments. Moving forward, we implemented Tokenization using the Keras’s ‘Tokenizer’ library, followed by padding the sequence generated by the tokenizer object. As far as embedding is concerned, we first implemented our own embedding layer but since it wasn’t performing well enough, we had to switch to the pre-trained Glove embedding model. Therefore, we first loaded the Glove text file having a dimension of 100d. Then we created the embedding matrix using the vocabulary generated by the tokenizer object. Using this embedding matrix, we developed the embedding layer. This was followed by performing train and test split, wherein we split the dataset into training and validation data in the ratio of 80:20. We performed the train-validation split separately for both the ‘student_star’ and ‘student_difficult’ target labels. For the purpose of passing the data into our LSTM model, we converted them into tensors first and made them suitable for training. The targets were additionally one-hot encoded because

they were creating shape errors during the experimentation. Also, number of classes were set to 9 because initially we were considering the target variable ‘diff_index’ and ‘star_rating’ as the target labels for the problem which had 9 classes, but later on we realized that we were supposed to perform sentiment analysis on ‘student_star’ and ‘student_difficult’ ratings and fortunately setting 9 as the number of classes worked out for these target labels as well, and provided the best accuracy. Then we started with building our models. Our team came up with four possible approaches:

1. Glove embedding model with regression approach
2. Glove embedding model with classification approach
3. Word2Vec embedding model with regression approach
4. SimpleRNN model with classification approach.

Things that worked out and did not work out in our experiments:

1. We used the same model that was available under Tensorflow link in the documentation provided by the professor’s assignment document for building our model named ‘Glove architecture with regression and classification approach’, along with making slight changes to it (like using different regularization techniques such L1, L2, L1_L2, BatchNormalisation, early stopping) and found that it worked out the best for us in all the scenarios. For example, we tried the following techniques that did not work out: We tried to add dense layers, We tried adding more LSTM layers with the number of nodes in the order of 64, 32, 16, and a few other numbers of node combinations. We tried using L2 regularization with 0.00001 instead of our current L1_L2 with 0.001. We tried to decrease the Adam learning rate to 0.00001, and some more things, but no major improvements happened. Some methods decreased the accuracy, and some of these hyperparameter tuning approaches started overfitting the model. Therefore, we chose to continue working on the model that was available in the documentation for all models that we built.
2. In the second approach, we used Word2Vec embedding to see the probability of words appearing. We have imported the API from Gensim to load the (‘glove-wiki-gigaward-100’) vector with 100 dimensions. Later we converted the words of our data into vectors and added an embedded layer to the network architecture. In this model, we added 2 GRU layers with a dropout of 20%. The output layer is a linear dense layer with a single neuron to predict the star rating and difficulty rating values. We trained the same network model for both star rating and difficulty rating prediction separately. Using this embedding layer and GRU layers, we have achieved a validation accuracy of 88.8% for both star rating and difficulty rating, and validation MAE and validation loss are 0.1952 and 0.098 respectively for star rating and 0.199 and 0.098 respectively for difficulty rating. So in this model, we were able to achieve better performance without any overfitting.

Comparing Group 12’s approach to ours

One of the data formatting techniques utilized by Group 12 includes removing the most commonly occurring words in the English language. A stop words list removes all the commonly occurring words in the comments dataset. By removing stop words, the size of the dataset is reduced, which aids in training speed while also reducing storage requirements and memory usage. Moreover, group 12 also removes the commonly occurring words associated with "difficulty," which further reduces the size of the training data. Data formatting techniques by group 9 include converting the text to lowercase and removing all the special characters, which makes the text more concise and removes noise.

Both groups 9 and 12 use the string tokenization technique. String tokenization is breaking a string into smaller units called tokens. These tokens can be anything ranging from words to phrases. Tokenization helps to standardize the input data by converting it into a uniform format that machine learning models can process. Furthermore, both groups padded the sequence, which ensures that all the sequences are of a fixed length since many machine learning algorithms and deep learning models require that the training data be of a fixed size.

Group 12 has applied four different types of approaches in this project. While group 9 used four other techniques, two of which were regression and two methods are classification techniques. For the first approach, they applied topic modeling and then used Sentiment Analysis, Rescaling, and word embedding. Group 9 has used GloVe embedding and added bidirectional RNN and LSTM in architecture with 0.5 dropouts and L2 regularization to avoid overfitting. And with that, we achieved a loss of 0.09 on the Validation dataset for the student star rating and student difficulty rating with the regression approach achieved an MAE score of 0.15 and a loss of 0.09 for the student difficulty rating for the classification approach.

For the second approach, Group 12 used Sentiment analysis on unlabelled text data and got an MSE of 1.16 on the training dataset and 63% accuracy on the testing dataset for Star rating and MSE of 1.48 and 65% accuracy for test data in difficulty rating. They have used a sequential model with two LSTM layers by adding dropout in between the layers and also added a Dense layer.

In Approach 3 they cleaned test data to improve accuracy for difficulty and star rating, and as a result, they got a Training MSE of 1.29, and Testing MSE of 1.20, and testing accuracy of 65% for Star rating, and a Training MSE of 1.51 and Testing MSE of 1.5 and testing accuracy of 73% for difficulty rating. Group 9 has applied Word2Vec Embedding by embedding layer to convert the sequence of words into a vector representation. Also, we have multiple dropout layers to prevent overfitting and added multiple bi-directional GRU along with a dense layer. By using early stopping, we improved the generalization performance of the model and achieved 88.8% validation accuracy and 0.098 validation MSE for the student star rating and for the student difficulty rating.

Apart from that, we have also experimented with another approach wherein we built a model using simpleRNN and created our own embedding layer instead of using the Glove pre-trained embedding model. We used dropout and l2 regularization. And by doing so, we received a

validation accuracy of 42% and a validation loss of 1.2 for the star rating, whereas the validation accuracy was 31% and validation loss was 1.4 for the difficulty rating. At the other end, Group 12, in their 4th approach applied rescaling and word embedding. They have also used the text-processing.com API which is a simple JSON over HTTP web service for text mining and natural language processing. In that analysis, they got 51% testing accuracy for both star and difficulty ratings.
Overall, from their 4 approaches, they got the best result in their 3rd approach with text cleaning for Sentiment Intensity Analysis version 2.

Group 12 performance			
	Method	Star rating evaluation	Star difficulty evaluation
1	Topic modeling	accuracy 62%	accuracy 54%
2	Sentiment Intensity Analysis, Version 1	Convert MSE to accuracy 63%	Convert MSE to accuracy 65%
3	Sentiment Intensity Analysis, Version 2	Convert MSE to accuracy 65%	Convert MSE to accuracy 73%
4	Rescaling	accuracy 51%	accuracy 51%

Group 9 performance			
	Method	Star rating evaluation	Star difficulty evaluation
1	Glove Model with Regression Approach	accuracy 88.9% MAE 0.19 loss 0.09	Accuracy 88.9% MAE 0.19 loss 0.09
2	Glove Architecture with Classification approach	accuracy 35%, MAE 0.17. loss 0.09	accuracy 37.71%, MAE 0.15 loss 0.09
3	Word2Vec Embedding	accuracy of 88.8%, and validation MSE of 0.098.	accuracy of 88.8% and validation MSE of 0.098.
4	Simple RNN	accuracy 42%	accuracy 31%

Comparison		
	Group 12	Group 9
Highest accuracy	73%	88.9%

Number of models	4	4
Conclusion	Both groups used multiple methods and gained high accuracy. The key difference is Group 12 manipulated the topic modeling method, Group 9 manipulated both the regression and classification methods.	

Comparing Group 13's approach to ours

In this section, we are going to discuss the different architectures and approaches used by Group 13 and Group 9.

Group 13 used 6 different architectures to predict the student rating and student difficulty level, among which 3 are regression techniques and the remaining 3 are classification techniques. Group 9 used 4 different architectures, among which 2 are regression techniques and the other 2 are classification techniques. Both of our groups have used the same **RateMyProfessor_Sample** dataset for training, but Group 13 has divided the dataset into **training** (**80%**), **testing** (**10%**), and **validation** (**10%**) sets, but we have just split the dataset into **training** (**80%**) and **testing** (**20%**) sets and used **test set as validation set**. They used Tensorboard to depict the loss curves whereas we did not use any visualization tools for tracking the progress of the training.

Group 13 used the Dense model as the baseline and compared the remaining models with that. Details of different architectural networks for Group 13 are provided below in the table

Models	LSTM Regression	LSTM Classification	GRU Regression	GRU Classification	GRU Regression w/corrected words	GRU Classification w/corrected words
Trainable parameters	425,762	426,224	327,202	327,664	327,202	327,664
Layers	11	11	11	11	11	11
Activation function	ReLU, Linear	ReLU, Softmax	ReLU, Linear	ReLU, Softmax	ReLU, Linear	ReLU, Softmax
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Loss Function	MSE	Sparse Categorical CrossEntropy	MSE	Sparse Categorical CrossEntropy	MSE	Sparse Categorical CrossEntropy
Dropout Rate	50%	50%	50%	50%	50%	50%

Regularizer	-	-	-	-	-	-
Bidirectional RNN Layers	2	2	2	2	2	2
Epochs	15	15	15	15	15	15
Early Stopping	-	-	-	-	-	-
Embedding Layer	Glove	Glove	Glove	Glove	Glove	Glove

Group 13 used the same architecture for all the models but with different RNNs in LSTM and GRU models. They have also used multiple output models for predicting the star rating and difficulty rating while our group trained all the architectures separately for star rating and difficulty rating.

Group 13 has experimented with another technique by correcting the misspelled words to see whether LSTM gives higher accuracy or GRU. They installed an autocorrecting package and applied it to the dataset before converting it into vectors. Later they added a glove embedding layer and trained the model with GRU architecture to see if the accuracy was improved or not.

Group 13 performance			
	Method	Star rating evaluation(validation set values)	Star difficulty evaluation(validation set values)
	Simple RNN as baseline model	Loss 2.1532 MAE 1.3142	Loss 1.6728 MAE 1.0642
1	LSTM Classification	Accuracy 46.45% loss 1.5432	Accuracy 34.3% loss 1.4624
2	LSTM Regression	MAE 0.8419 loss 1.0959	MAE 0.9569 loss 1.3375
3	GRU Classification	Accuracy 45.9% loss 1.5678	Accuracy 33.97% loss 1.4586
4	GRU Regression	MAE 0.7855 loss 0.9912	MAE 0.9795 loss 1.3966

5	GRU Classification with corrected words	Accuracy 47% loss 1.1404	Accuracy 33.85% loss 1.4498
6	GRU Regression with corrected words	MAE 0.8584 loss 1.1293	MAE 0.9665 Loss 1.3638

The **accuracies and MAE loss** or the performance of the various models for **Group 13** are listed in the above table. According to the performance values, GRU models are giving better prediction results for star rating, and LSTM models are best suited to predict the difficulty rating of the comments.

Comparison		
	Group 13	Group 9
Highest accuracy	47% for difficulty and least loss is 1.337 for star rating	88.9% for both
Number of models	6	4
Conclusion	Through Group 13's experimentation, ultimately they derived the following conclusions. From their results they determined that the LSTM models were better at predicting the difficulty rating. However, the GRU models were better at predicting the star ratings.	

Conclusion

In conclusion, this review allowed us to get an insight into other groups' approaches and compare our approach with theirs. Group 12 tried multiple approaches ranging from topic modeling to sentiment analysis. Of their four methods Sentiment Intensity Analysis, Version 2 achieved the highest accuracy for both star rating and difficulty evaluation. On the other hand, group 13 also utilizes multiple methods ranging from simple RNNs to Gated Recurrent Units (GRU) for classification and regression tasks. These approaches show that multiple approaches can be incorporated to achieve the best accuracy while minimizing overfitting.