Final Project Report

Prepared by,

1. G.Araf Ahmed (22101532) 2.

Md Shihab Sarker (22101516) 3.

Fahim Foysal Abit (22101543)

Course Code: CSE440

Section: 3

Group No: 1

Final Project Report

Abstract:

This project runs different Machine Learning and Neural Network models with different vectorization methods and does experiments to find the best models and techniques. First we loaded the datasets in the code environment then did extensive exploratory data analysis. The EDA shows us there are a total 280K rows and 2 columns in both the test and train dataset. There are 10 categories in the target feature “Class”. The “QA Text” column had three inputs in it, question title, question content and best answer. We found that merging all these types into a plain text give the best result so we merged them into a single text string and did data preprocessing on them. Then we did data preprocessing using lemmatization and then split the train data set in train and validation in a 80-20 ratio. Then we vectorized the data using different vectorization methods, TF-IDF, BOW, Glove and Skip gram. And we ran different machine learning and neural network models on each of them totalling 22 separate experiments. We found out that the best model for BOW vectorization is Naive Bayes, for TF-IDF Logistic Regression, for both Glove and Skip gram GRU.

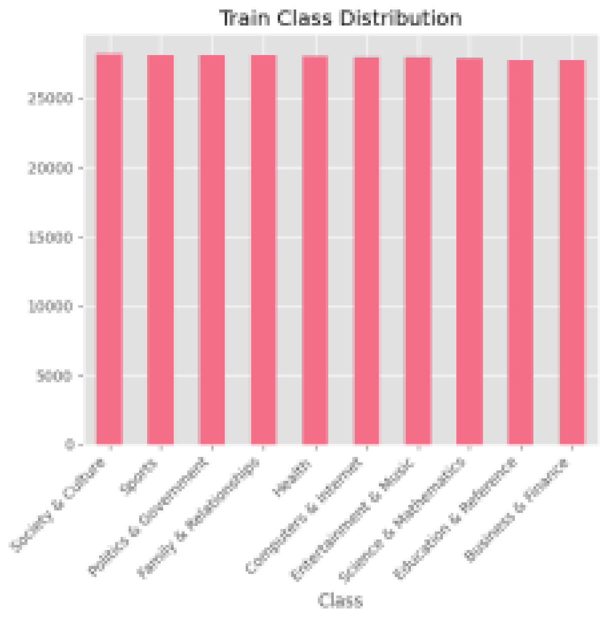
Introduction:

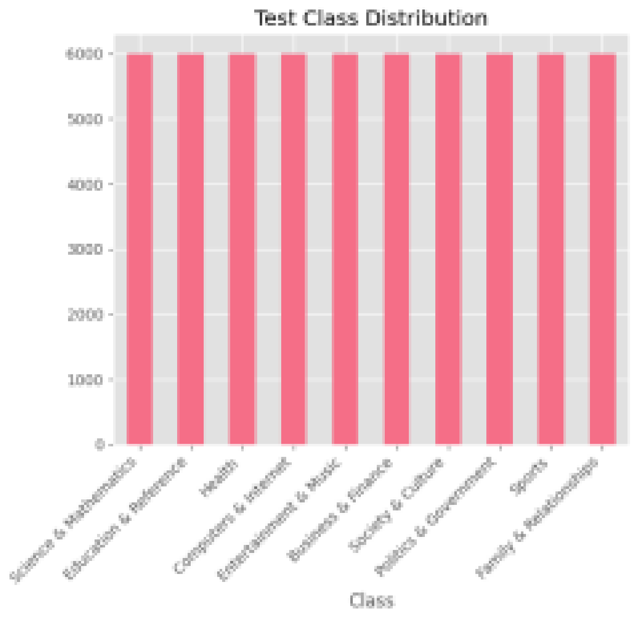
The task was given as a final project for the NLP course CSE440. We were given 2 datasets. One being the test dataset and the other train. Both of them had a total

280k rows and 2 columns each. The target feature “Class” had 10 unique categories. They were almost evenly distributed throughout the dataset. The “QA Text” column had three components. Question Title, Question Content and Best Answer. We found that the average text length pattern for title length is 10.7 words, average answer length: 59.2 words and content is often missing: 45.0% of cases.

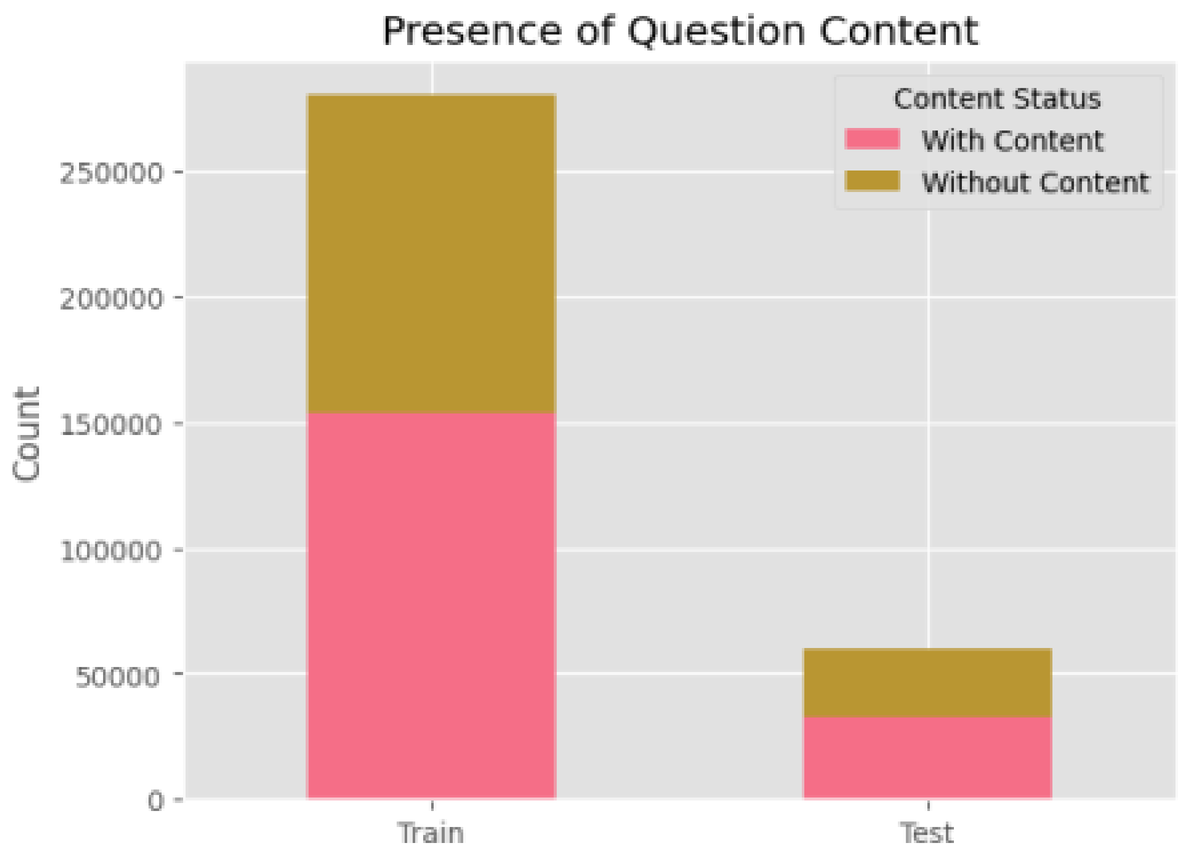
Methodology:

While doing exploratory data analysis we found that we had 10 unique categories in our target feature “Class”. We then ran code to see the class distributions. The class distribution for our both datasets among these 10 categories were almost similar. Below is the histogram for both of the Class columns from train dataset and test dataset.

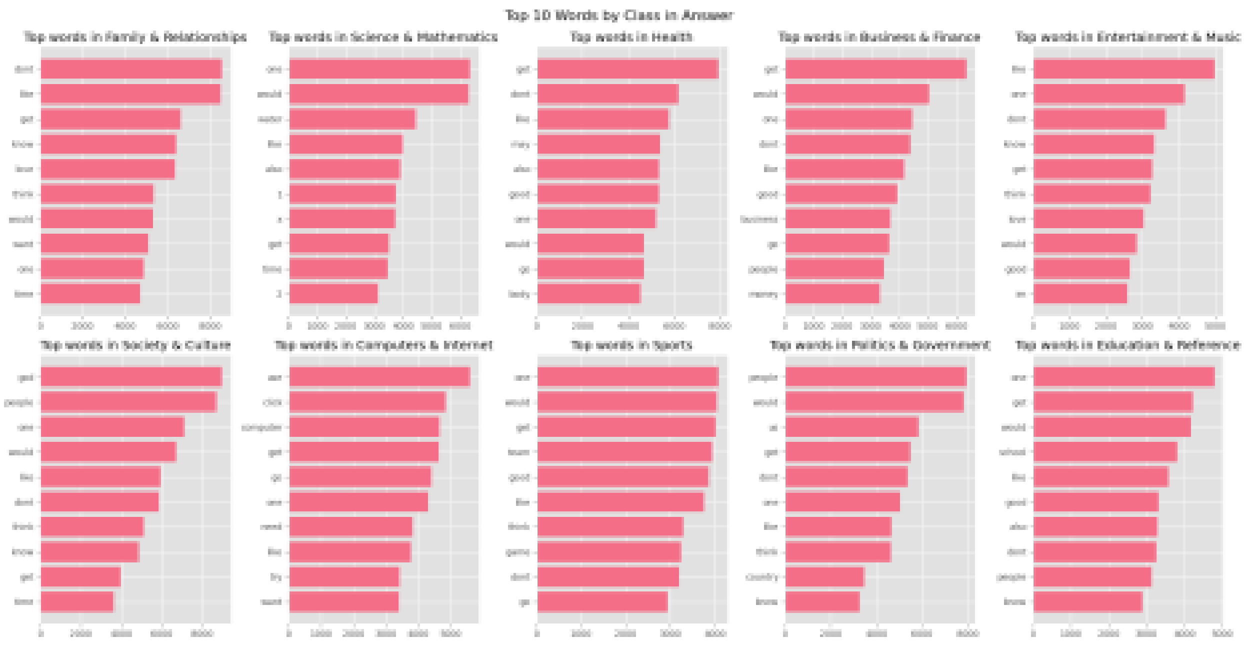




We found that the Question Content component in the “QA Test” column has a lot of missing data. We just replaced the Nan values with empty strings.



We also ran codes to see the top 10 words in Question title, Question Component, Best Answer and the target feature “Class”. Below is the graph for top 10 words in the Class feature.

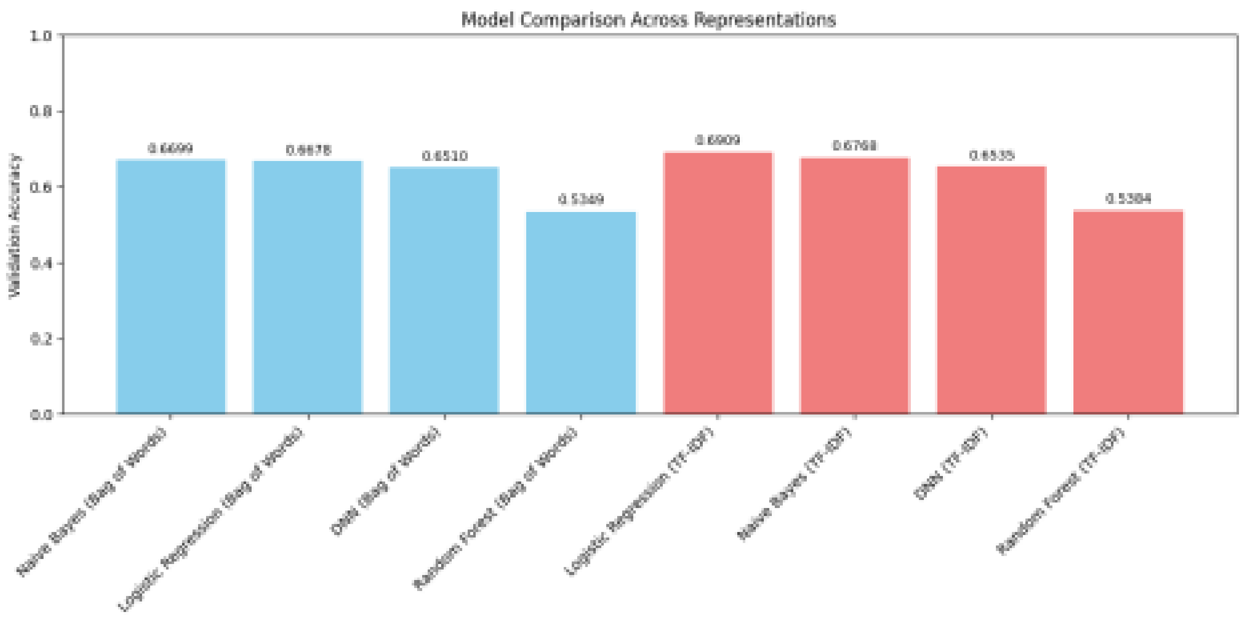


We ran the data on a Logistic regression model using TF-IDF vectorization to extract information on how we should create the vectors. We found out concatenating all those components into a single text string gives better output than separately vectorizing those components and using hstack on them later. That's why we concatenated the components.Then we removed all the punctuation marks, turned the text into lower case handled Nan values with empty strings and then removed stopwords. Then we ran lemmatization techniques on the data. Now we have clean data ready for vectorization. First we used 2 vectorization methods, BOW and TF-IDF and after vectorizing the data for each of them we ran the vectors on the Logistic Regression model, Naive Bayes model, Random forest model and Deep Neural Network model. We used different parameter sets to see which parameters gives the best results. With this we were done doing 2x4=8 experiments. Next we used Glove vectorization with 7 Neural Network models. We used Deep Neural Network, SimpleRNN, GRU, LSTM, Bidirectional SimpleRNN, Bidirectional GRU and Bidirectional LSTM. Then we used Skip Gram vectorization to run the same 7 models. We used a 3 hidden layer deep neural network for both BOW and TF-IDF vectorization. For Glove vectorization we used different neural network models with different hyper parameters. For the Skipgram vectorization we also use all those 7 Neural Network models. The Deep Neural Network model had only one hidden layers due to it taking too much time to run if we increased the layers. We used 512 batch size for it and 15 epochs. And for other models we used different batch sizes and epochs based mainly on saving while running the models smoothly.

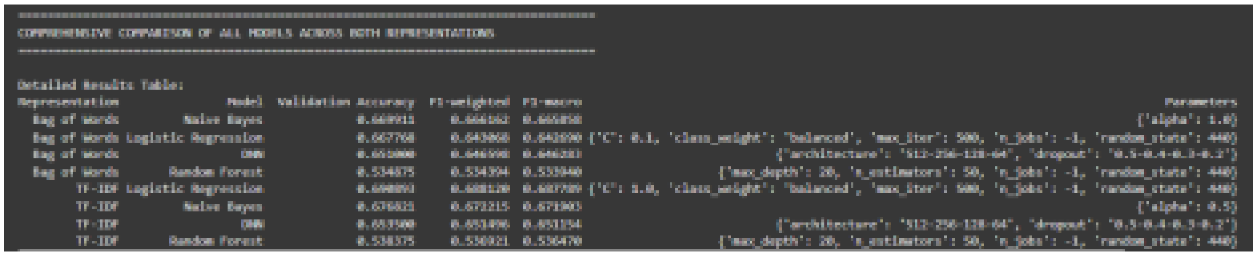
Results:

For the BOW vectorization among Logistic Regression, Naive Bayes, Random Forest and Deep Neural Network, Naive Bayes did the best with a F1-macro score of 0.6622.

In the TF-IDF vectorization part among the same models Logistic Regression produced the best result. It had a F1-macro score of 0.6878.

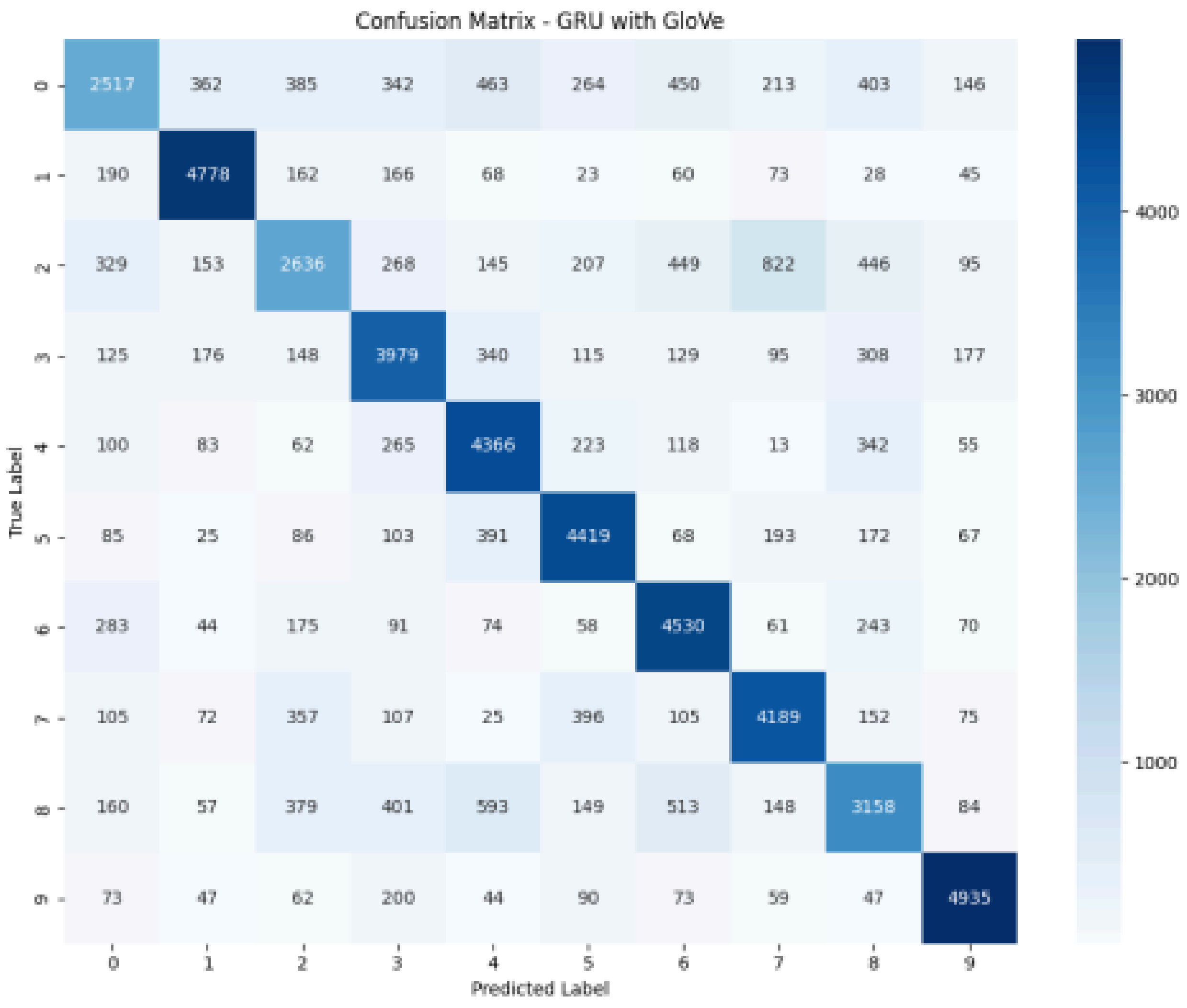


And here is a screenshot of the performance of other models on their best parameter set:

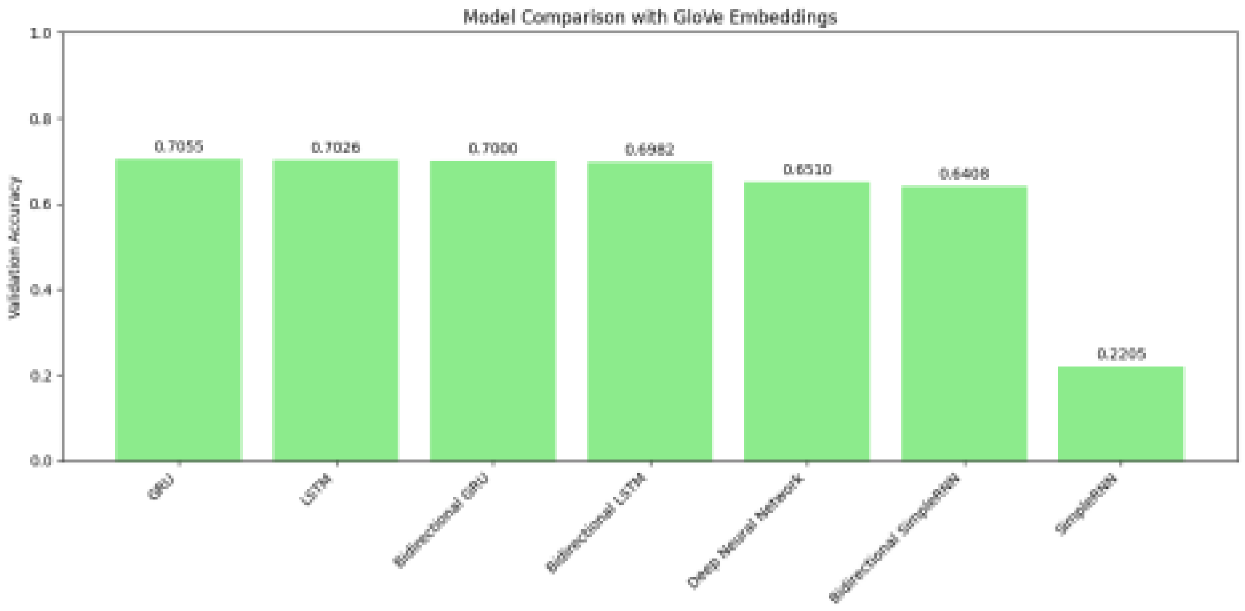


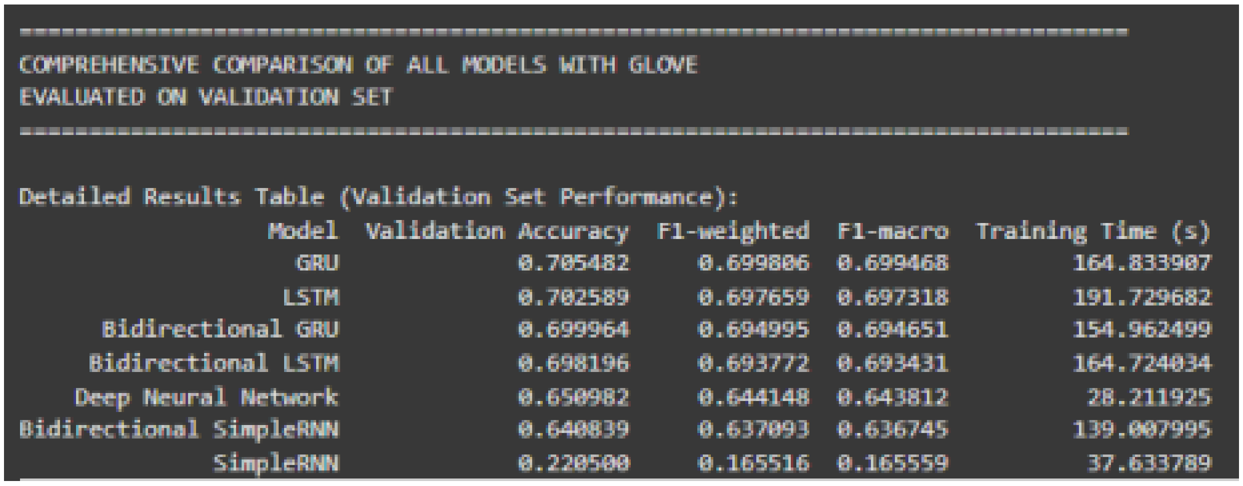
```
================================================================================
COMPREHENSIVE COMPARISON OF ALL MODELS ACROSS BOTH REPRESENTATIONS
================================================================================

Detailed Results Table:
Representation               Model  Validation Accuracy  F1-weighted  F1-macro                                                                                Parameters
  Bag of Words         Naive Bayes             0.669911     0.666162  0.665858                                                                            {'alpha': 1.0}
  Bag of Words Logistic Regression             0.667768     0.643068  0.642030 {'C': 0.1, 'class_weight': 'balanced', 'max_iter': 500, 'n_jobs': -1, 'random_state': 440}
  Bag of Words                 DNN             0.651000     0.646598  0.646283                      {'architecture': '512-256-128-64', 'dropout': '0.5-0.4-0.3-0.2'}
  Bag of Words       Random Forest             0.534875     0.534394  0.533940              {'max_depth': 20, 'n_estimators': 50, 'n_jobs': -1, 'random_state': 440}
        TF-IDF Logistic Regression             0.690893     0.688120  0.687709 {'C': 1.0, 'class_weight': 'balanced', 'max_iter': 500, 'n_jobs': -1, 'random_state': 440}
        TF-IDF         Naive Bayes             0.676821     0.672215  0.671903                                                                            {'alpha': 0.5}
        TF-IDF                 DNN             0.653500     0.653496  0.651254                      {'architecture': '512-256-128-64', 'dropout': '0.5-0.4-0.3-0.2'}
        TF-IDF       Random Forest             0.538375     0.530921  0.536470              {'max_depth': 20, 'n_estimators': 50, 'n_jobs': -1, 'random_state': 440}
```

For Glove vectorization among the 7 Neural Network model GRU produced the best score, an F1-macro of ~0.70. Below is the confusion matrix of GRU ran on Glove:

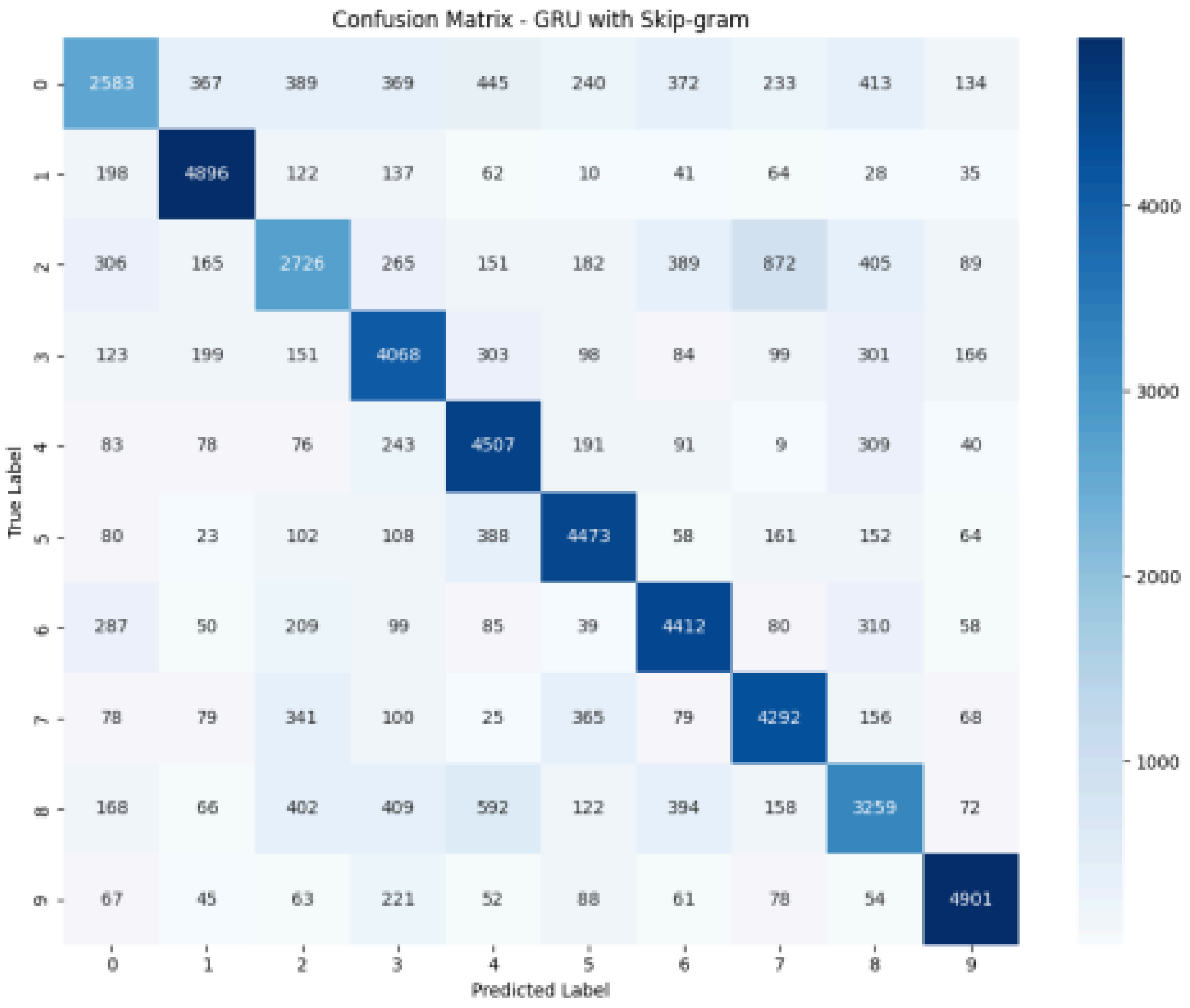


And the performance of other models on Glove is also below:



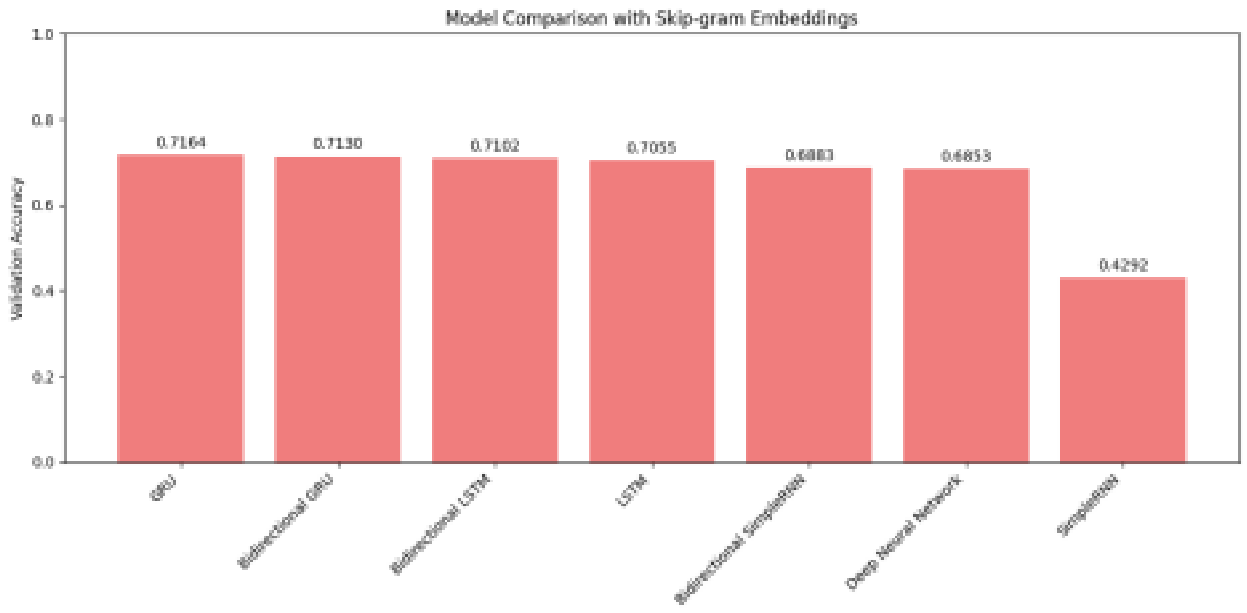

```
================================================================================
COMPREHENSIVE COMPARISON OF ALL MODELS WITH GLOVE
EVALUATED ON VALIDATION SET
================================================================================

Detailed Results Table (Validation Set Performance):
                  Model  Validation Accuracy  F1-weighted  F1-macro  Training Time (s)
                    GRU             0.705482     0.699806  0.699468         164.833907
                   LSTM             0.702589     0.697659  0.697318         191.729682
      Bidirectional GRU             0.699964     0.694995  0.694651         154.962499
     Bidirectional LSTM             0.698196     0.693772  0.693431         164.724034
    Deep Neural Network             0.650982     0.644148  0.643812          28.211925
Bidirectional SimpleRNN             0.640839     0.637093  0.636745         139.007995
              SimpleRNN             0.220500     0.165516  0.165559          37.633789
```

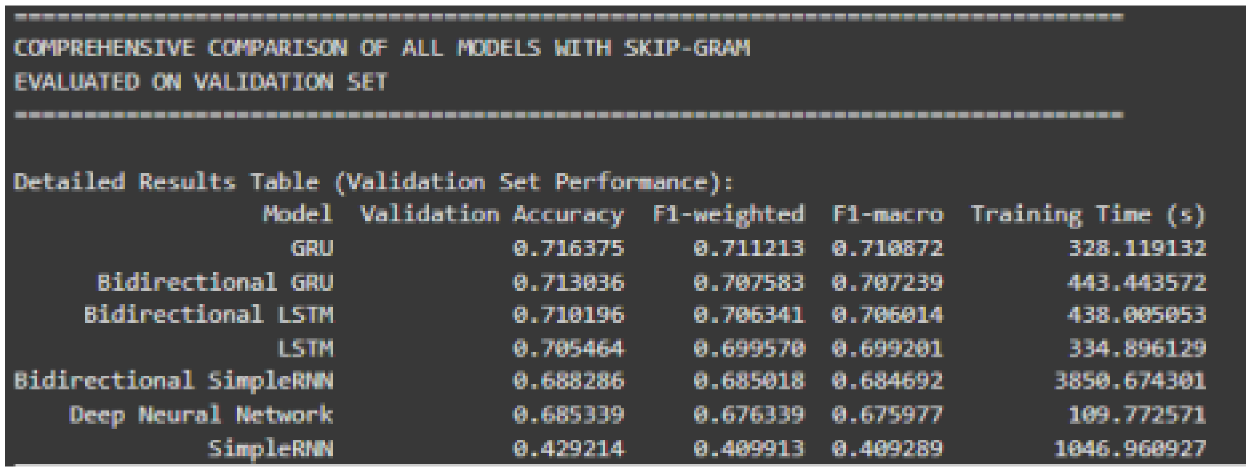
For the Skipgram model GRU again produced the best result. An F1-macro of 0.71. Below is the confusion matrix GRU ran on embedding.



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	2583	367	389	369	445	240	372	233	413	134
1	198	4896	122	137	62	10	41	64	28	35
2	306	165	2726	265	151	182	389	872	405	89
3	123	199	151	4068	303	98	84	99	301	166
4	83	78	76	243	4507	191	91	9	309	40
5	80	23	102	108	388	4473	58	161	152	64
6	287	50	209	99	85	39	4412	80	310	58
7	78	79	341	100	25	365	79	4292	156	68
8	168	66	402	409	592	122	394	158	3259	72
9	67	45	63	221	52	88	61	78	54	4901

The performance of other models that ran on GRU is also below:





```
================================================================================
COMPREHENSIVE COMPARISON OF ALL MODELS WITH SKIP-GRAM
EVALUATED ON VALIDATION SET
================================================================================

Detailed Results Table (Validation Set Performance):
                    Model  Validation Accuracy  F1-weighted  F1-macro  Training Time (s)
                      GRU             0.716375     0.711213  0.710872         328.119132
        Bidirectional GRU             0.713036     0.707583  0.707239         443.443572
       Bidirectional LSTM             0.710196     0.706341  0.706014         438.005053
                     LSTM             0.705464     0.699570  0.699201         334.896129
  Bidirectional SimpleRNN             0.688286     0.685018  0.684692        3850.674301
      Deep Neural Network             0.685339     0.676339  0.675977         109.772571
                SimpleRNN             0.429214     0.409913  0.409289        1046.960927
```

Among all the 22 experiments we ran we got the best performance from GRU that ran on Skipgram embedding. And the Simple RNN model ran on Glove produced the worst result with an F1-macro score of 0.165.

Conclusion:

To sum up, for BOW we had the best performance from Naive Bayes, for TF-IDF Logistic Regression, for both Glove and Skipgram we had GRU. GRU is clearly the winner here with the highest validation, F1-weighted and F1-macro scores. The Neural Network models on average produced the best results. Only the simple RNN model produced very poor results. The simple RNN model produced the poorest result overall. Our main limitation in this project was time and resources. We had very little time and the models were taking a lot of time to train. And the low GPU usage time made things harder for us. That's why we had to make the models very simple, with few hidden layers and a big batch size. In the future we could produce better results if we have enough time and enough resources. We could make the Neural Network models a bit more complex, have a bit more epochs and achieve better results.

References

[1] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation,"
EMNLP, 2014.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector
space," ICLR, 2013.

[3] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR

2011. [4] F. Chollet et al., "Keras," 2015. [Online]. Available:

https://keras.io

[5] TensorFlow, Google Brain Team. [Online]. Available:
https://www.tensorflow.org

[6] NLTK: Natural Language Toolkit. [Online]. Available: https://www.nltk.org