NLP - Topic Classification

S Anjana Shankar
CS24MTECH14015

June 5, 2025

1 Introduction

The task was to identify the topic of a multi class problem which has 2 features: title and abstract of a paper. The problem was treated as a multi class single label problem. Multiple ML as well as DL models were tried on the dataset provided and their evaluation metrics noted. The model with the best scores were identified.

2 Data Analysis and Preprocessing

- The data was cleaned before giving as input to the model.
- The data was checked to see if any NULL values are present in the dataset.
- The 'abstract' and 'title' columns were combined into a single 'text' column for processing.
- The output columns were converted into a single 'label' column for multiclass classification.
- The number of records for each class was identified and plotted.

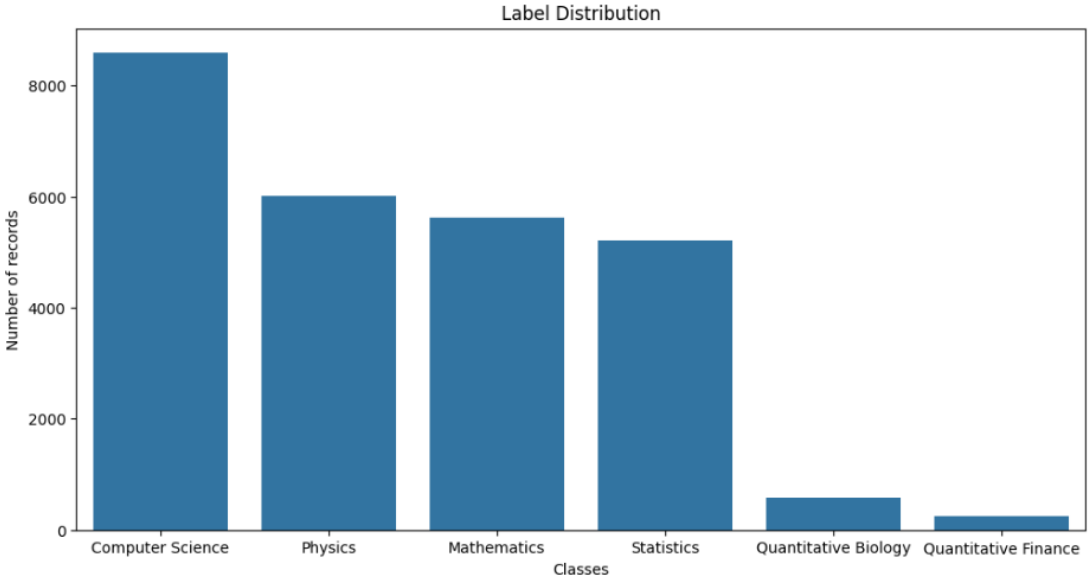

Figure 1: Class Imbalance

- The plot shows significant imbalance in the classes.

- The data was checked and removed any duplicates if present.
- The data was converted into lower case characters.
- Special characters were removed from the data keeping only alphabets.
- The data was tokenized and lemmatization was performed on the tokens and returned after joining back.
- All of the stop words were also removed in this process.
- After the data was cleaned, synonym replacement using wordnet was done to augment data so that class imbalance can be handled better.
- After augmenting data, the data was converted into embeddings using tfidf vectorizer.
- Label Encoding was performed on the labels.
- The dimension of the embeddings produced were also reduced using truncated SVD.
- Finally, after all the processing, the data was split into test and train with test size set to 20% and shuffle set to True.

3 Training

- After dataset preprocessing, the dataset was trained and tried on various Machine Learning and Deep Learning models.
- The number of iterations were set to 300 for most models and run to lesser iterations if it reaches convergence earlier.
- GridSearch was used to find the best parameters based on the macro F1 score of the model.
- The various Machine Learning models tried were:
    1. Logistic Regression
    2. Support Vector Machine
    3. Random Forest
    4. XG Boost
- The various Deep Learning models tried were:
    1. Multilayer Perceptron
    2. RNN
    3. Bidrectional RNN
    4. LSTM
    5. Bidrectional LSTM
    6. GRU
    7. Bidrectional GRU
    8. CNN

4 Machine Learning Models

Various different machine learning models were tried on the dataset and the evaluation metrics were noted. The loss curves were also plotted for most of the models to ensure convergence.

4.1 Logistic Regression

Logistic regression is a statistical model that predicts the probability of a binary outcome by applying the sigmoid function to a linear combination of input features.

The hyperparameters for the models were set as:

- L2 regularization set to be used.
- Inverse of the regularization strength set to 1
- Tolerance for stopping criteria set to 1e-4
- To track the loss, the model was run with one iteration at a time.

The model was run for 200 epochs and the losses plotted.

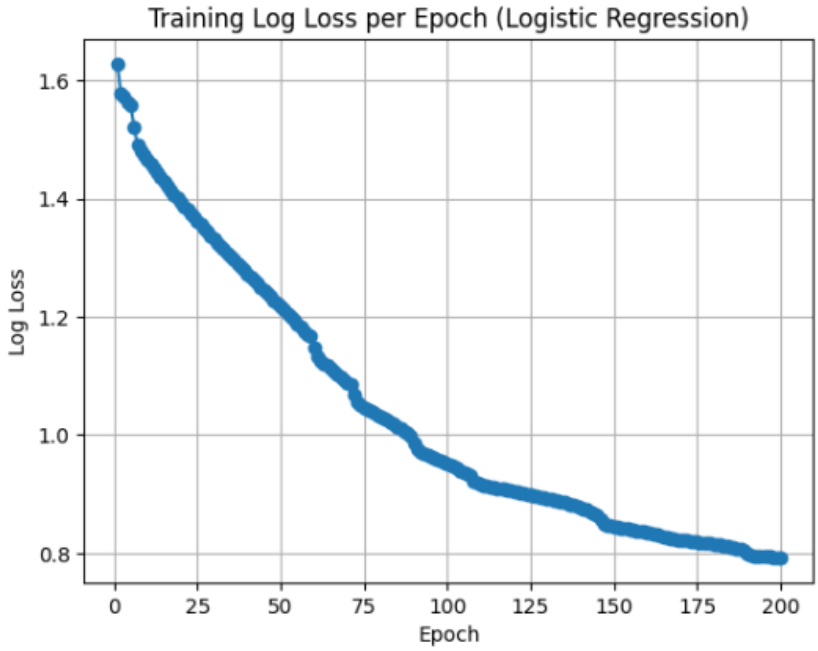

Figure 2: Loss for Logistic Regression Model

After training, the classification report was printed.

Classification Report:

	precision	recall	f1-score	support
0	0.64	0.75	0.69	1714
1	0.82	0.75	0.78	905
2	0.85	0.86	0.85	1085
3	0.72	0.35	0.47	376
4	0.87	0.45	0.59	181
5	0.62	0.64	0.63	1386
accuracy			0.71	5647
macro avg	0.75	0.63	0.67	5647
weighted avg	0.72	0.71	0.70	5647

Figure 3: Loss for Logistic Regression Model

4.2 Support Vector Machine

Support Vector Machine (SVM) is a machine learning algorithm used for classification and regression.
It works by finding the best hyperplane (or surface) that separates different classes of data with the widest possible margin. The hyperparameters for the model were set as:

- The kernel function used was the Radial Basis Function (RBF), also known as the Gaussian kernel.
- RBF allows the SVM to model non-linear decision boundaries, making it more flexible than a linear kernel.
- The regularization parameter was set to 1.0
- It controls the trade-off between having a wide margin (simpler model) and classifying all training points correctly
- The kernel coefficient for the RBF kernel was set as scale, which computes gamma according to scale and makes it adaptable to the data automatically.
- The probability estimates via cross-validation during training is enabled
- The class weights are set to balanced to take care of the class imbalance
- The model was run till convergence.
- The final Hinge Loss recorded was 0.188.
- After training, the metrics were noted and classification report printed.

	precision	recall	f1-score	support
0	0.87	0.75	0.81	1714
1	0.82	0.83	0.82	905
2	0.92	0.88	0.90	1085
3	0.86	0.98	0.92	376
4	0.90	1.00	0.95	181
5	0.81	0.91	0.86	1386
accuracy			0.85	5647
macro avg	0.86	0.89	0.88	5647
weighted avg	0.86	0.85	0.85	5647

Figure 4: Classification report for Support Vector Machine Model

4.3 Random Forest

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their outputs to improve accuracy and reduce overfitting.
Thhe hyperparameters of the model was set as:

- The number of trees in the forest was set to 1 and later the model was incrementally trained, once for each tree addition.
- The tracking for oob score was enabled.

- Incremental training of the model was enabled to track the errors after each addition.
- The maximum depth of tree was set to 20 and the maximum number of features to square-root.
- The class weights were set to balanced subsample to handle class imbalance

The model was run as many times as the number of trees(which was set to 100) by incrementally adding one tree by tree and the out of bag error was noted on each tree addition.

The error was plotted and is shown below:

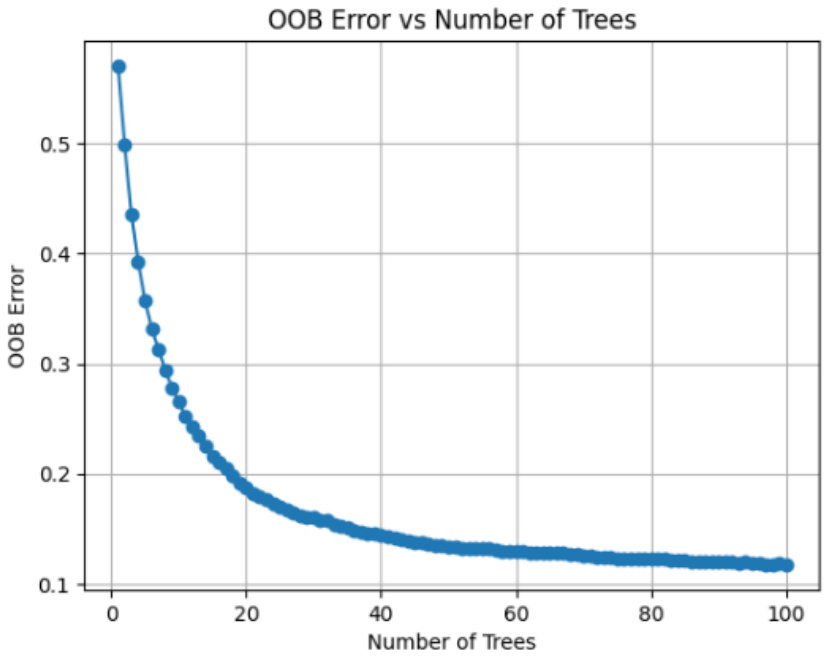

Figure 5: Loss for Logistic Regression Model

The classfication report for the model is:

	precision	recall	f1-score	support
0	0.84	0.91	0.87	1714
1	0.85	0.74	0.79	905
2	0.93	0.83	0.88	1085
3	0.96	0.94	0.95	376
4	0.97	0.99	0.98	181
5	0.93	0.99	0.96	1386
accuracy			0.89	5647
macro avg	0.91	0.90	0.91	5647
weighted avg	0.89	0.89	0.89	5647

Figure 6: Classification report for Random Forest Model

4.4 XGBoost

XGBoost is a powerful and efficient gradient boosting algorithm that builds an ensemble of decision trees to optimize predictive accuracy through sequential learning.
The hyperparameters for the model were set as:

- The number of boosting rounds were set to 300.
- The learning rate set to 0.1
- The maximum depth of tree set to 7.
- Both subsample and colsample_bytree were set to 0.9
- multi:softprob was mentioned for multiclass classification and the evaluation metric was set to multiclass log loss.

The following is the plot of the loss curve:

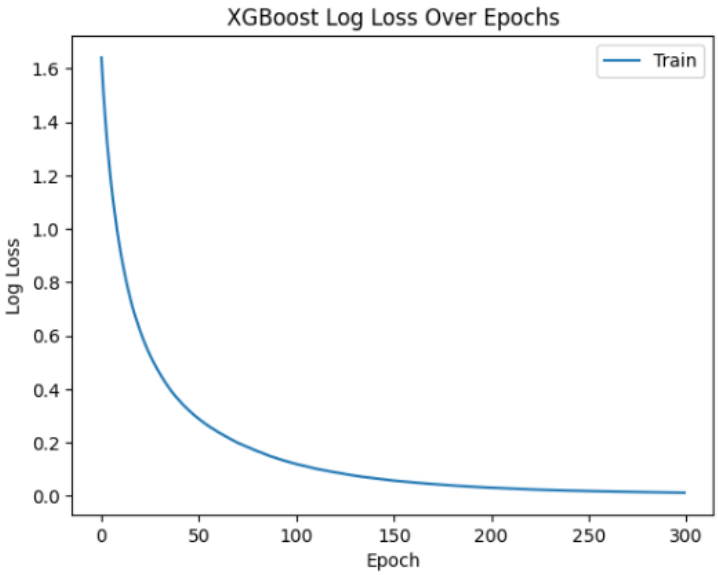

Figure 7: Loss for Logistic Regression Model

The following is the classification report after training:

	precision	recall	f1-score	support
0	0.89	0.86	0.88	1714
1	0.85	0.81	0.83	905
2	0.92	0.88	0.90	1085
3	0.94	0.97	0.95	376
4	0.97	1.00	0.99	181
5	0.90	1.00	0.95	1386
accuracy			0.90	5647
macro avg	0.91	0.92	0.92	5647
weighted avg	0.90	0.90	0.90	5647

Figure 8: Classification report for XGBoost Model

4.5 Comparison of Machine Learning models

- Among the machine learning models, XGBoost achieved the best performance, with both accuracy and F1 scores approaching 90%, indicating strong predictive capability across all classes.
- The Random Forest model also performed competitively, ranking a close second in terms of both accuracy and F1 metrics.
- This can be because both of these models automatically capture non-linear patterns and feature interactions without requiring explicit feature engineering.
- Trees naturally split data hierarchically, which works well when data has complex structure and thus gives an advantage over the other models.
- Logistic Regression assumes more of a linear relationship between the input features and the output.
- Out of all the models, logistic regression performed relatively poor.

5 Deep Learning Models

After machine learning models, deep learning models were tried on the dataset to capture the complex patterns. Cross Entropy Loss was used while training.

5.1 MLP

- A Multi-Layer Perceptron (MLP) is a type of neural network that consists of multiple layers of interconnected neurons and can learn complex non-linear relationships between inputs and outputs.
- The model was made with a single hidden layer with a dimension of 16 here.
- The activation function was chosen to be ReLU
- The model was trained for 300 epochs and the classification report was printed.

Classification Report:

	precision	recall	f1-score	support
0	0.74	0.74	0.74	1714
1	0.83	0.78	0.80	905
2	0.88	0.88	0.88	1085
3	0.79	0.78	0.78	376
4	0.88	0.82	0.85	181
5	0.73	0.78	0.75	1386
accuracy			0.79	5647
macro avg	0.81	0.80	0.80	5647
weighted avg	0.79	0.79	0.79	5647

Figure 9: Classification report for MLP Model

5.2 RNN

- A Recurrent Neural Network (RNN) is a type of neural network designed for sequential data, where connections between neurons form cycles to allow information to persist across time steps.
- The hidden dimension was set to 32 and the number of layers to 1
- The model was trained for 300 epochs.
- After training, the classification report was printed:

Classification Report:

	precision	recall	f1-score	support
0	0.80	0.78	0.79	1714
1	0.83	0.78	0.81	905
2	0.89	0.88	0.88	1085
3	0.79	0.78	0.79	376
4	0.92	0.91	0.92	181
5	0.79	0.86	0.83	1386
accuracy			0.82	5647
macro avg	0.84	0.83	0.84	5647
weighted avg	0.83	0.82	0.82	5647

Figure 10: Classification report for RNN Model

5.3 Bidirectional RNN

- Along with RNN, a bidirectional RNN was also tried out.
- A Bidirectional RNN differs from a standard RNN by having two separate hidden layers that process the sequence forward and backward, enabling it to use both past and future context for better predictions.
- The hidden dimension was set to 64 and the number of layers to 2.
- The model was run for 300 epochs.
- After training, classification report was noted:

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.80	0.84	1714
1	0.82	0.76	0.79	905
2	0.90	0.87	0.88	1085
3	0.89	0.97	0.93	376
4	0.92	0.98	0.95	181
5	0.85	0.99	0.91	1386
accuracy			0.87	5647
macro avg	0.88	0.90	0.88	5647
weighted avg	0.87	0.87	0.87	5647

Figure 11: Classification report for Bidirectional RNN Model

5.4 LSTM

- Long Short-Term Memory (LSTM) is a special type of recurrent neural network designed to remember information for long periods by using gated cells that control the flow of information, helping to overcome the problem of forgetting in standard RNNs.
- The hidden dimension set to 64 and number of layers to 2.
- The model was trained for 300 epochs and the classification report was printed

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.80	0.84	1714
1	0.82	0.77	0.79	905
2	0.90	0.88	0.89	1085
3	0.89	0.97	0.93	376
4	0.95	1.00	0.97	181
5	0.86	0.99	0.92	1386
accuracy			0.88	5647
macro avg	0.89	0.90	0.89	5647
weighted avg	0.88	0.88	0.87	5647

Figure 12: Classification report for LSTM Model

5.5 Bidirectional LSTM

- A Bidirectional LSTM differs from a standard LSTM by having two separate LSTM layers processing the input sequence in both forward (past to future) and backward (future to past) directions, allowing it to capture context from both before and after each time step for better sequence understanding.
- The hidden dimension set to 64 and number of layers to 2.
- The model was trained for 300 epochs and the classification report was printed

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.82	0.85	1714
1	0.83	0.78	0.80	905
2	0.91	0.88	0.90	1085
3	0.89	0.98	0.93	376
4	0.95	1.00	0.97	181
5	0.88	0.99	0.93	1386
accuracy			0.88	5647
macro avg	0.89	0.91	0.90	5647
weighted avg	0.88	0.88	0.88	5647

Figure 13: Classification report for Bidirectional LSTM Model

5.6 GRU

- A Gated Recurrent Unit (GRU) is a simplified type of recurrent neural network that uses gating mechanisms to control the flow of information, enabling it to capture dependencies in sequential data with fewer parameters than LSTM.
- A GRU differs from an LSTM by combining the forget and input gates into a single update gate and using fewer parameters, making it simpler and often faster to train while still effectively capturing long-term dependencies.
- The hidden dimension set to 64 and number of layers to 2.
- The model was trained for 300 epochs and the classification report was printed.

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.80	0.84	1714
1	0.83	0.77	0.80	905
2	0.91	0.88	0.89	1085
3	0.89	0.98	0.93	376
4	0.96	1.00	0.98	181
5	0.86	1.00	0.92	1386
accuracy			0.88	5647
macro avg	0.89	0.90	0.89	5647
weighted avg	0.88	0.88	0.88	5647

Figure 14: Classification report for GRU Model

5.7 Bidirectional GRU

- A Bidirectional GRU differs from a regular GRU by having two separate GRU layers that process the input sequence forward and backward.
- The hidden dimension set to 64 and number of layers to 2.
- The model was trained for 300 epochs and the classification report was printed.

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.81	0.85	1714
1	0.83	0.78	0.81	905
2	0.91	0.88	0.90	1085
3	0.88	0.98	0.93	376
4	0.94	0.99	0.97	181
5	0.86	0.99	0.92	1386
accuracy			0.88	5647
macro avg	0.89	0.91	0.90	5647
weighted avg	0.88	0.88	0.88	5647

Figure 15: Classification report for Bidirectional GRU Model

5.8 CNN for Text

- A Convolutional Neural Network (CNN) for text applies convolutional filters over word embeddings or character embeddings to automatically extract important local patterns (like phrases or n-grams) from the text, which helps capture features useful for tasks like classification or sentiment analysis.
- The model uses several 1D convolutional layers with kernel sizes [3, 4, 5] to capture different n-gram features (e.g: trigrams, 4-grams, 5-grams) from the input text embeddings simultaneously.
- After each convolution, max pooling is applied over the entire feature map to extract the most important feature for each filter, and the pooled outputs from all filters are concatenated to form a fixed-length feature vector representing the input.
- The concatenated features pass through a dropout layer (to reduce overfitting) before being fed into a fully connected linear layer that outputs the classification scores.
- The model was trained for 500 epochs and the classification report was printed.

Classification Report:

	precision	recall	f1-score	support
0	0.49	0.70	0.57	1714
1	0.60	0.52	0.56	905
2	0.69	0.79	0.74	1085
3	0.36	0.03	0.06	376
4	0.69	0.20	0.31	181
5	0.57	0.45	0.50	1386
accuracy			0.56	5647
macro avg	0.57	0.45	0.46	5647
weighted avg	0.56	0.56	0.54	5647

Figure 16: Classification report for CNN Model

5.9 Comparison of different Deep Learning Models

- Out of all the models, the best performing models were LSTM, GRU, Bidirectional LSTM and Bidirectional GRU based on accuracy.
- The accuracy reported was around 88%.
- The highest F1 Score was reported by the Birectional GRU and LSTM models making them the best models out of these.
- Out of the models, CNN performed poorly, this can be made better with hyper parameter tuning.
- The loss curves of all the DL models were plotted and is shown below:

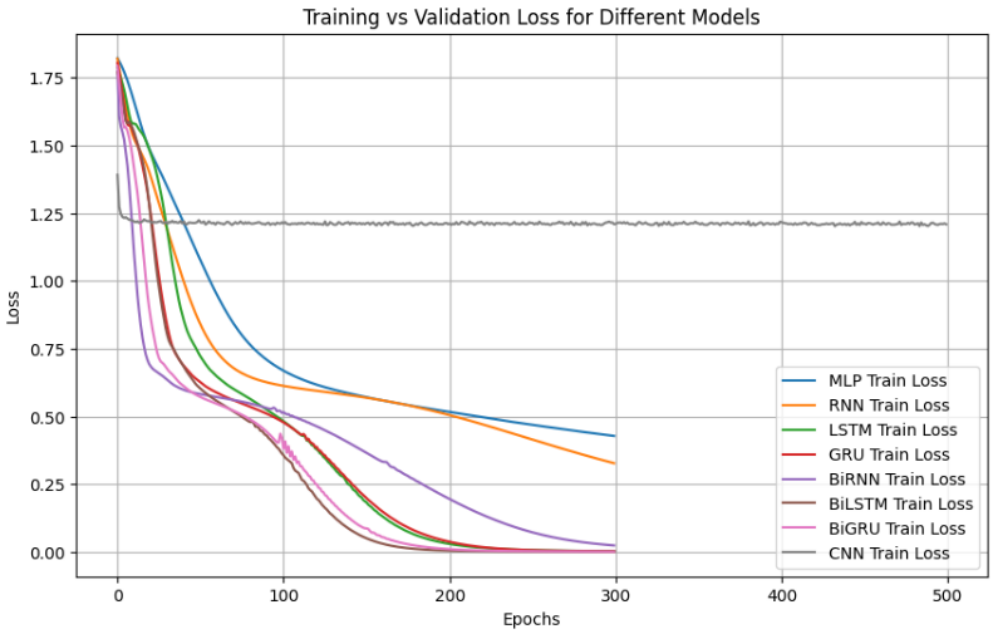

Figure 17: Loss curves of the Deep Learning Models

6 Best Model

- Out of all of the models tried on, the best performing models are XGBoost from the machine learning models and Bidirectional LSTM and Bidirectional GRU among the DL models.
- Comparing the F1 scores and the accuracy score metrics of these models it can be seen that XGBoost produces the best scores and thus is best suited for the dataset.
- The best scores can be seen in the bottom table:

Accuracy	**Macro Precision**	**Macro Recall**	**Macro F1-Score**
0.90	0.91	0.92	0.92

Table 1: Performance metrics of the XGBoost model (best scores)

7 Conclusion

Various different models were tried on a multi class problem of identifying the topic of a paper given 2 features: title and abstract. Based on the evaluation metrics, the best models and processing suited for the dataset was identified.