Assignment 2
Text Classification Report

Umanshiva Ladva AI22BTECH11016
Siddhesh Gholap AI22BTECH11007
Rajiv Chaudhary AI22BTECH11021

1. Introduction

In this report, we work with two datasets to perform text classification using four different models. The datasets contain research article titles, abstracts, and topic labels, and our objective is to predict the topic based on the provided text.

To achieve this, we apply various preprocessing techniques, including lowercasing, tokenization, stopword removal, and lemmatization. After preprocessing, we train and evaluate four different models:

- FastText
- Logistic Regression
- Long Short-Term Memory (LSTM)
- CNN+LSTM

We analyze and compare the performance of these models using different evaluation metrics to determine their effectiveness in text classification.

2. Dataset

The datasets used are:

- **Dataset 1**
  - Contains 20,972 entries.
  - Includes research article titles, abstracts, and topic labels.
  - Topics include Computer Science, Physics, Mathematics, Statistics, Quantitative Biology, and Quantitative Finance.
  - The task is to predict the topic of the research article based on its title and abstract.
- **Dataset 2**
  - Contains 3,927 entries.
  - Includes text titles, content, and domain labels.
  - Domains include categories as: Entertainment, Healthcare, Sports, Technology and Tourism.
  - The task is to classify text into its appropriate domain.

3. Preprocessing Steps

To prepare the text data for classification, we applied the following preprocessing techniques:

- **Lowercasing:** Converts text to lowercase for consistency.
- **Punctuation and Special Character Removal:** Removes all non-alphanumeric characters except spaces.
- **Number Removal:** Eliminates numerical values from the text.
- **Tokenization:** Splits text into individual words using NLTK.
- **Stopword Removal:** Eliminates common English stopwords.
- **Lemmatization:** Converts words into their base forms using WordNetLemmatizer.

For both the dataset we preprocess the text data by combining the TITLE and ABSTRACT/CONTENT columns into a single Texts column. The text is then tokenized, and Word2Vec embeddings (dim=100 for both dataset)are used to convert the text into numerical features.

4. Models Used

We experimented with the following four classification models:

4.1. Logistic Regression

Sentence vectors were generated by averaging word embeddings for each sentence, ensuring that the text was converted into a numerical format suitable for model training.The data was first preprocessed by normalizing the feature vectors using StandardScaler to ensure consistency in scale.

For Dataset 1, The model selection involved using a OneVsRestClassifier with Logistic Regression, employing the ”lbfgs” solver with a maximum iteration limit of 3000 to handle the multi-class classification problem.For Dataset 2, Multi-class labels were converted to their respective class indices using argmax. A Logistic Regression model with multinomial classification was used, utilizing the ”lbfgs” solver with a maximum of 1000 iterations. The model was trained on the word2vec-transformed training dataset, and after training, it was tested on the standardized test dataset.Where its performance was evaluated based on relevant metrics. The evaluation metrics include accuracy, precision, recall, and F1-score.

4.2. Long Short-Term Memory (LSTM)

The LSTM model is designed to capture sequential dependencies in text data. The text is tokenized using the Tokenizer class from Keras, and sequences are padded to a uniform length of 150 tokens for 1st Dataset and 850. Word2Vec embeddings are used to initialize the embedding layer, which maps words to 100-dimensional vectors. The LSTM layer has 64 units for Dataset-1 and 128 for Dataset-2 with a dropout rate of 0.2 to prevent overfitting. A dense layer with 32 units for Dataset 1 and 64 units for Dataset 2 and ReLU activation follows, with another dropout layer (rate=0.5) for regularization. The output layer uses a sigmoid activation function to handle multi-label classification in case of Dataset 1 and for Dataset 2 a softmax layer is used. The model is compiled with the Adam optimizer. Binary cross-entropy loss is used and it is trained for 10 epochs with a batch size of 32. Evaluation metrics include accuracy, precision, recall, and AUC.

4.3. CNN+LSTM

The CNN+LSTM model combines convolutional neural networks (CNN) for local feature extraction and LSTM for sequential modeling. The input text is tokenized and padded similarly to the LSTM model. The embedding layer uses pre-trained Word2Vec embeddings. The model architecture includes two 1D convolutional layers with 128 and 64 filters, respectively, followed by max-pooling layers. A bidirectional LSTM layer with 10 units is used to capture sequential dependencies. The output layer uses a sigmoid activation function for multi-label classification for Dataset 1 and softmax for Dataset 2. The model is compiled with the Adam optimizer and binary cross-entropy loss, and it is trained for 20 epochs with a batch size of 32. Evaluation metrics include accuracy, precision, recall, and AUC.

4.4. FastText

FastText is an efficient text classification model developed by Facebook AI. It represents words as vectors using a continuous bag-of-words (CBOW) approach and can handle large datasets efficiently. We use FastText for its ability to provide quick predictions and handle out-of-vocabulary words effectively.

5. Results

We evaluated model performance using accuracy, precision, recall, and F1-score. The comparative results are shown in Table 1.

Model	Precision	Recall	F1-Score
Logistic	0.61	0.27	0.38
LSTM	0.82	0.77	0.80
CNN+LSTM	0.80	0.75	0.78
FastText	0.80	0.73	0.76

Table 1. Performance Comparison of Models on Dataset 1

Model	Accuracy	Precision	Recall	F1-Score
Logistic	0.82	0.82	0.82	0.82
LSTM	0.98	0.98	0.98	0.98
CNN+LSTM	1	1	1	1
FastText	0.89	0.89	0.89	0.89

Table 2. Performance Comparison of Models on Dataset 2

Analysis of Model Performance

Dataset 1

For Dataset 1, the LSTM model achieved the highest F1-score of 0.80, with a strong balance between precision 0.82 and recall 0.77. The CNN+LSTM model followed closely, with an F1-score of 0.78, suggesting that adding CNN to LSTM did not significantly improve performance. FastText performed similar to CNN LSTM with an F1-score of 0.78, precision of 0.80, and recall of 0.7, making it competitive with CNN+LSTM. This indicates that FastText is a viable alternative for LSTM-CNN. Logistic Regression had the lowest performance, with an F1-score of 0.38 and a recall of 0.27, making it the least effective model for this dataset.

Dataset 2

For Dataset 2, CNN+LSTM achieved perfect classification, with an accuracy, precision, recall, and F1-score of 1.00, proving its capability to fully capture the dataset's patterns. The LSTM model also performed exceptionally well, with an accuracy of 0.98 and an F1-score of 0.98, maintaining its strong generalization ability. FastText maintained solid performance, achieving 0.89 in all metrics, making it a strong alternative when slightly lower complexity is preferred. Logistic Regression, which performed poorly in Dataset 1, showed significant improvement in Dataset 2, with an F1-score of 0.82.

Conclusion

Deep learning models, especially LSTM and CNN+LSTM, performed best across both datasets, proving to be the most effective for text classification tasks.Logistic Regression, while weak in Dataset 1, showed notable improvements in Dataset 2.Overall, LSTM-based architectures remain the top choice, while FastText emerges as a competitive alternative.