Forum Discussion Categorization Project Report

Team Members

Ahmed Mubarak Hussein - 2021170034

Amna Ahmed Mirghani - 2021170087

Amira Nasser Sayed - 2021170093

Aya Mohammed - 2021170109

Mohammed Ahmed - 2021170662

Abstract

The Forum Discussion Categorization project addresses the challenge of efficiently classifying large volumes of forum discussions into predefined categories. With the growing use of online forums for sharing information, effective categorization is crucial for improving user experience and enabling better content management. This project utilized natural language processing (NLP) and deep learning techniques to preprocess and analyze forum discussion data.

The dataset consists of 24989 samples. The data was prepared, involved steps such as text cleaning, tokenization, and lemmatization. Feature extraction techniques, including Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, and BERT embedding model were employed to represent textual data.

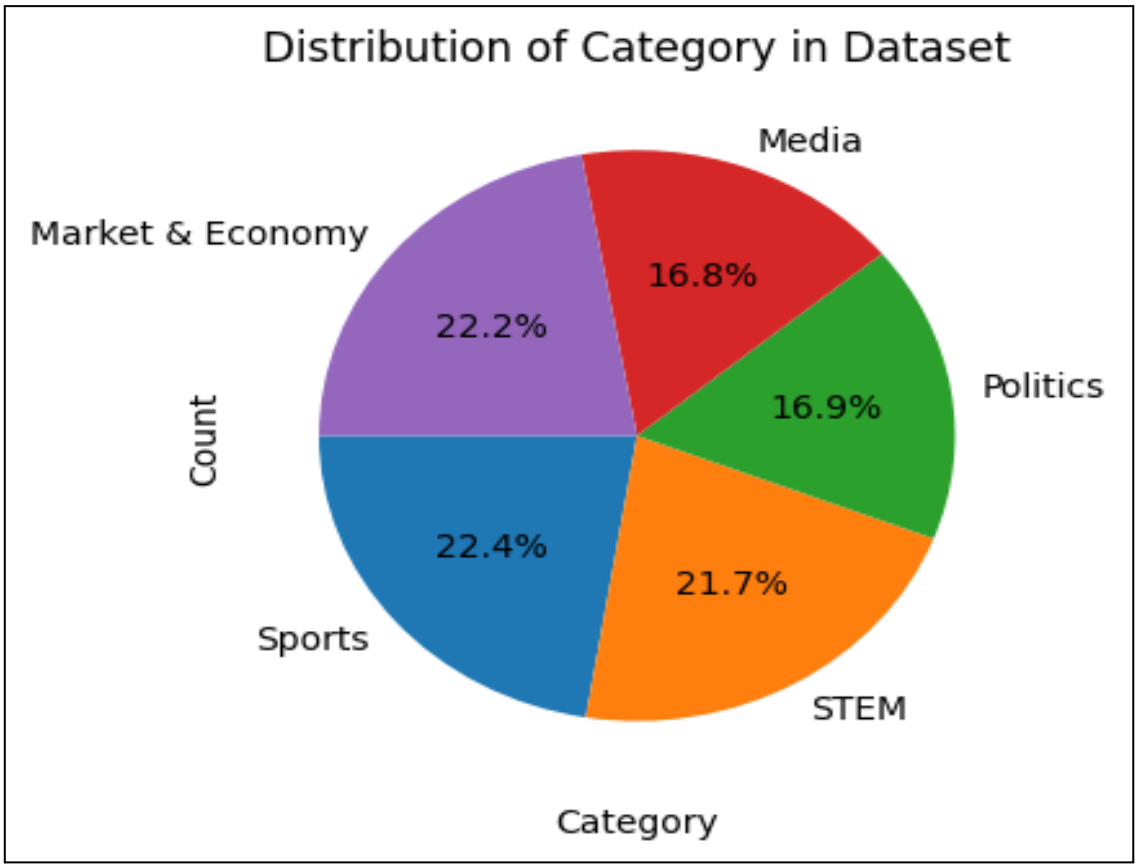
Deep learning architectures such as feed forward neural networks, CNNs, LSTMs and Transformers were implemented to classify the data effectively, and were evaluated by measuring the accuracy of their predictions.

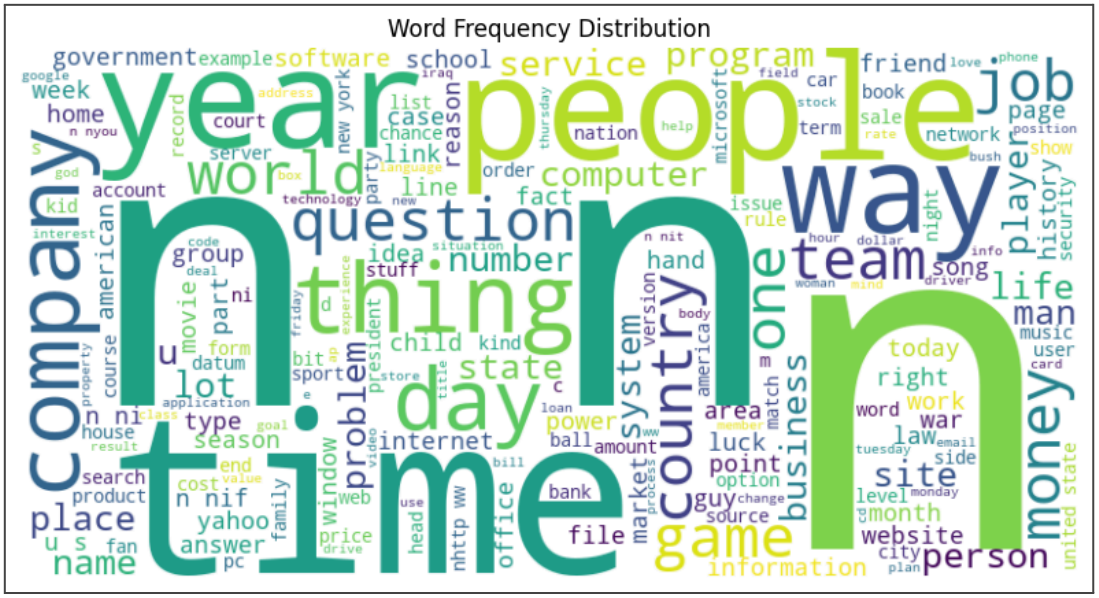
The results indicated that the **Transformer** was the best architecture for the task, achieving the highest accuracy of **73%.**

Dataset

[NN'25] Forum Discussions Categorization | Kaggle

Dataset with 24989 samples across different topic types.





Data Preprocessing

- Handling Missing Values: Rows containing null values were dropped
- Data Splitting: The dataset was split into training and validation sets, with 80% of data for training and 20% for testing.
- Text Cleaning:
    - Emojis were removed from the text
    - Text was converted to lowercase
    - Lemmatization was applied to reduce words to their base forms
    - Only nouns, pronouns,and verbs were retrained to focus on linguistic features.
- Encoding: (y) was converted to numerical labels using label encoding.
- Feature Extraction:
    - TF-IDF to capture the importance of terms within corpus
    - Word2Vec to capture semantic meanings
    - Tokenizer + Embedding: convert into sequences of tokens, which were then passed through an embedding layer to generate dense vector representations of tokens

Training data

- Number of training samples: 19716
- Max document length: 521 words

Classification Models

1. Feed Forward Neural Network

Trial	Input Features	Layers	Regularization	Initializer	Dropout	Optimizer	LR	Accuracy
1	TF-IDF	Dense(128) Dense(64)	None	Default	0.5	Adamax	0.001	Train : 0.803 Test: 0.698
2	TF-IDF	Dense(128, L2=0.008) Dense(64, L2=0.008)	L2 (0.008)	Default	0.5	Adamax	0.001	Train: 0.719 Test: 0.624
3	TF-IDF	Dense(128) Dense(64)	L2 (0.001)	HeNormal	0.5	Adamax	0.001	**Train: 0.796** **Test: 0.694**
4	W2V	Dense(128) Dense(64)	L2 (0.001)	HeNormal	0.5	Adamax	0.001	Train: 0.578 Test: 0.570

-

---

2. CNN

Model Architecture:

- **Input Layer**: Shape matches the number of features in the training dataset (`X_train_tfidf.shape[1]`).
- **Conv1D Layers**:
  - Two Conv1D layers, each with 128 filters and a kernel size of 3.
  - ReLU activation for non-linearity.
  - L2 regularization (`L2(0.01)`) to prevent overfitting.
- **MaxPooling1D Layers**: Pool size of 2 after each Conv1D layer.
- **Dropout Layers**: Added after each convolutional and dense layer with a rate of 0.2 to reduce overfitting.
- **Flatten Layer**: Flattens the output from the convolutional layers.
- **Dense Layer**: Fully connected layer with 128 units and ReLU activation.
- **Final Dense Layer**:
  - 5 units with a softmax activation function for multi-class classification.
- **Compilation**:
  - Adamax optimizer.
- Sparse categorical cross-entropy loss.
- Accuracy as the evaluation metric.

Training Setup

- **Callbacks**:
  - **EarlyStopping**: Monitors validation loss (`val_loss`) with a patience of 3 epochs, restoring the best weights if validation loss does not improve for 3 consecutive epochs.
  - **ReduceLROnPlateau**: Reduces the learning rate by a factor of 0.5 if validation loss does not improve after 2 epochs, with a minimum learning rate of 1e-6.
- **Class Weights**:
  - Computed using `compute_class_weight` to address class imbalance by assigning higher weights to underrepresented classes in the dataset.

Evaluation

- Train Accuracy: 0.7973726987838745
- Train Loss: 0.9530853033065796
- Validation Accuracy: 0.6693711876869202
- Validation Loss: 1.2023688554763794
- With LR: 2.5000e-04

Trials

Trial	Feature Extraction	Layers	Regularization	Initializer	Dropout	Optimizer	LR	Accuracy
1	TF-IDF	Conv1D(128,5*5) Conv1D(256,3*3)	L2(0.01)	Default	0.2	Adamax	5.0000e-04	Train : 0.7516 Test: 0.6533
2	TF-IDF	Conv1D(128,3*3) Conv1D(128,3*3)	L2 (0.01)	Default	0.2	Adamax	5.0000e-04	**Train: 0.797** **Test: 0.669**
3	TF-IDF	Conv1D(128,3*3) Conv1D(128,3*3)	L2 (0.01)	Default	0.3	Adamax	2.5000e-04	Train: 0.7891 Test: 0.668
4	W2V	Conv1D(128,3*3) Conv1D(128,3*3)	L2 (0.01)	Default	0.2	Adamax	5.0000e-04	Train: 0.518 Test: 0.512

Transformer

Preprocessing

Different preprocessing steps were used, since transformers are naturally suited for dealing with textual input, requiring less preprocessing.

- **Tokenizing**
  - Using BERT tokenizer
    `tokenizer = AutoTokenizer.from_pretrained('distilbert-base-uncased')`
- **Embedding**
  - Using BERT pretrained model
    `embedder = TFAutoModel.from_pretrained('distilbert-base-uncased')`
  - Embedding dimension = 786
- **Positional encoding**
  - Reduced accuracy, so it was dropped in the final5= version.

Model Architecture:

- **Input layer (**sequence_length,hidden_size)
- **Dense layer** with 256 units to reduce input hidden dimension to 256
- **Transformer block**
  - Multi head attention layer (2 heads)
  - Residual connection
  - Layer Normalization
  - Feed forward network
  - Residual connection
  - Layer Normalization
- **Global average pooling layer**
- **Feed forward network**
- **Output layer**

Hyperparameters

- Activation function: `gelu`
- Kernel initializer: `he_normal`
  - Output layer: `glorot_uniform`
- Kernel regularizer`: L2(1e-3)`
- Optimizer: `Adam(learning_rate=0.00001, clipnorm=1.0)`

Training Setup

- **EarlyStopping:** Monitors validation loss (`val_loss`) with a patience of 3 epochs, restoring the best weights if validation loss does not improve for 3 consecutive epochs.
- **ModelCheckpoint:** Monitors validation loss (`val_loss`) and saves the model every time there is an improvement. The model can then be manually saved at various epochs while monitoring the loss and accuracy. This enabled us to save various checkpoints, evaluate them and pick the best model while balancing learning (loss) and generalisation ability.

Best Results

(Checkpoint at epoch 24)

- Training accuracy: **0.7549**
- Validation accuracy: **0.73178**

Trials

Trial	Input Dimension reduction	Attention block layers	Feed forward layers	Regularization	Kernel Initializer	Activation	LR	Gradient clipping	Mixed precision	Batch Size	Val Acc
1	Dense(256) float32	Dense(128) Dense(128)	Dense(64) Dense(32)	L2(1e-3)	`He_normal` for hidden, `glorot` for output	gelu	1e-5	clipnorm =1.0	No	4	**0.732**
2	Dense(256) float32	Dense(64) Dense(64)	Dense(64) Dense(32)	L2(1e-3)	`He_normal` for hidden, `glorot` for output	gelu	1e-4	clipnorm =1.0	No	16	0.72
3	Dense(128) int32	Dense(64) Dense(32)	Dense(64) Dense(32)	None	None	relu	1e-4	clipvalue =1.0	Yes (float16)	8	0.61
4	Dense(128) int32	Dense(32) Dense(32)	Dense(64) Dense(32)	None	None	relu	1e-3	None	Yes (float16)	4	0.526

---

4. LSTM Model

Model Architecture

The Long Short-Term Memory (LSTM) model is implemented to capture temporal dependencies in the text data.

Preprocessing

The LSTM model utilized the same preprocessing steps as described in the main project, ensuring consistency across all models. This includes text cleaning, tokenization, lemmatization, and encoding methods as detailed in the project report.

Model Details

- **Input Dimension**: Shape matches the tokenized and embedded input data.
- **Hidden Dimension**: Configurable number of hidden units to capture patterns in the data.
- **Output Dimension**: Number of categories in the classification task (5 in this case).

The LSTM cell is structured with the following gates:

- **Input Gate**: Controls the extent to which new input flows into the cell state.
- **Forget Gate**: Determines the proportion of previous information to retain.
- **Output Gate**: Regulates the information passed to the next layer or output.