Forum Discussions Categorization

[Deep Learning & Neural Network]

Team ID: SC_2

2021170463	محمد خالد توفيق محمد
2021170392	فرح صفوت عز الرجال علي
2021170454	محمد أسامة علي عبد الحليم
2021170125	بولا عماد إدوارد شحاته
2021170245	سهير خالد محمد السيد

Data Preparation

1. Data Loading and Cleaning

- **Remove Nulls**
- **Handling contractions and abbreviations `expand_abbreviations`, `expand_contractions` functions.**
- **Lower Case & Removing punctuations**

- **Drop Small Length: drop the text that smaller the 20**

- **Handling Mismatches: Replace mismatch rows with "Media" to handle the unbalance in the data using replace_with_media**

- **Replacing all categories of numeric rows to mode**

- **Handling duplicates by drop 306 row of the duplicates rows**

- **Encoding Category: Map each category using category_mapping dictionary**

- **Lemmatization & Removing Stop: Using preprocess_text**

    - **This function splits text into tokens,**
    - **removes stop words and applies lemmatization to normalize words.**
    - **It then joins the processed words back into a single string.**

- **Final data shape after data cleaning = 22707 row**

Models:

1- Word2Vec + Dense

Description

- This model utilizes a custom-trained Word2Vec embedding to transform text into numerical vectors. The embeddings are processed through a series of dense layers to classify text into one of five categories.
- By leveraging Word2Vec, the model captures semantic relationships between words. The dense layers allow the model to learn complex features from the text representations.
- Class balancing is handled via computed class weights, helping to deal with any skew in the dataset.

Architecture

- **Input Layer:** Input sequences are padded/truncated to a fixed length of 128 tokens.
- **Embedding Layer:** Uses the **embedding_matrix** built from the Word2Vec model (vector size = 100), **trainable=False** ensures pre-learned embeddings remain fixed.
- **Global Average Pooling Layer:** Reduces the sequence dimension by computing the average of embeddings for the entire sequence, producing a fixed-size vector.
- **Dense Layer 1:** 128 units with ReLU activation, transforming the pooled embeddings.
- **Dropout Layer 1:** Dropout rate of 0.5 to reduce overfitting.
- **Dense Layer 2:** 128 units with ReLU activation for additional feature extraction.
- **Dropout Layer 2**: Dropout rate of 0.5 to further regularize the model.
- **Dense Layer 3**: 64 units with ReLU activation for deeper feature learning.
- **Dropout Layer 3**: Dropout rate of 0.5 to prevent overfitting.
- **Output Layer**: 5 units with softmax activation, producing a probability distribution across five classes.

Results

- **Validation Loss:** 0.8640
- **Validation Accuracy:** 67.26%
- **Train Loss:** 0.8153
- **Train Accuracy:** 69.12%

2- Glove Embedding + GRU & LSTM

Description

- This model uses pretrained GloVe embeddings (300-dimensional) to convert words into numeric vectors that capture semantic meaning.
- A bidirectional LSTM layer (256 units) followed by a bidirectional GRU layer (128 units) processes the text sequences.
- Global max pooling on each recurrent layer's output is concatenated and passed to a dense layer for final classification.
- The model is trained to predict one of multiple categories using softmax activation.
- Class balancing is handled via computed class weights, helping to deal with any skew in the dataset.

Architecture

- **Input Layer (max_sequence_length=128):** Receives padded token sequences.
- **Embedding Layer (300D GloVe, non-trainable):** Maps tokens to **300-dimensional** GloVe vectors.
- **Bidirectional LSTM (256 units):** Learns long-range dependencies in both forward and backward directions.
- **Bidirectional GRU (128 units):** Further refines sequence representations in both directions.
- **GlobalMaxPooling1D:** Extracts the most prominent features from both LSTM and GRU outputs.
- **Concatenate Layer:** Merges the pooled outputs.
- **Dense Output Layer (Softmax activation):** Produces probability distributions across the target classes.
- **Optimizer and Loss:** Uses Adam optimizer with **sparse_categorical_crossentropy** loss.

Results

- **Validation Loss: 0.**8476
- **Validation Accuracy:** 70.67%
- **Train Loss:** 0.7371
- **Train Accuracy:** 74.59%

3- BERT Feature Extraction + Dense

Description

- This model leverages a pretrained BERT **(bert-base-uncased)** model to generate contextual embeddings for each text.
- Specifically, it extracts features from the **[CLS]** token (the first token in each sequence) in the last hidden state of BERT. These features are then fed into a simple feed-forward network to perform classification into five categories.
- Class balancing is handled via computed class weights, helping to deal with any skew in the dataset.

Architecture

- **BERT Tokenizer:** Converts raw text into token IDs and attention masks, ensuring all sequences are at most 128 tokens.
- **BERT Model:**
    - Processes the token IDs and attention masks on GPU (if available).
    - We extract the [CLS] token representation **(shape: batch_size × hidden_dim)** from outputs.last_hidden_state.
- **Feed-Forward Neural Network:**

  • **Dense (128 units, ReLU)**: Learns high-level features from the BERT embeddings.
  • **Dropout (0.5)**: Mitigates overfitting by randomly dropping connections.
  • **Dense (64 units, ReLU)** + **Dropout (0.5)**: Adds another layer of feature transformation and regularization.
  • **Dense (5 units, Softmax)**: Outputs probabilities across the five target categories.

Results

- **Validation Loss:** 0.8011
- **Validation Accuracy:** 69.96%
- **Train Loss:** 0.7037
- **Train Accuracy:** 73.85%

4- Embedding all-mpnet-base-v2 + Dense

Description

- This model uses the **all-mpnet-base-v2** SentenceTransformer to convert each discussion into a numerical embedding.
- A feed-forward neural network with L2 regularization then classifies these embeddings into the target categories.
- Class balancing is handled via computed class weights, helping to deal with any skew in the dataset.

Architecture

- **SentenceTransformer Embeddings (all-mpnet-base-v2):** Generates a fixed-size embedding for each text discussion.
- **Feed-Forward Network:**
  • **Dense (256, ReLU, L2 regularization) + Dropout (0.3)**: Learns higher-level representations from embeddings.
  • **Dense (128, ReLU, L2 regularization) + Dropout (0.3)**: Further refines extracted features.
  • **Dense (num_classes, Softmax)**: Outputs probabilities for each category.

- **Optimizer and Loss:**
  • Adam optimizer with a learning rate of 1e-4.
  • Sparse categorical **crossentropy** loss for multi-class classification.

Results

- **Validation Accuracy:** 76.13%
- **Validation Loss:** 0.7421
- **Train Accuracy:** 77.59%
- **Train Loss:** 0.6948

5- Transformer Model + SentenceTransformer

Description

- This model first uses a **SentenceTransformer (all-roberta-large-v1)** to convert each discussion into a dense embedding that captures semantic meaning.
- In addition, **four extra features** are extracted from each text:
    1. Text length,
    2. Word count,
    3. Sentence count,
    4. Average word length.

    These are concatenated with the SentenceTransformer embeddings, creating a richer feature representation.
- The **EnhancedTransformer** then applies a single self-attention block (multi-head attention) and a feed-forward layer to learn contextual relationships.
- A final classification head outputs probabilities for each category.

Architecture

- **Input Features:** Concatenation of SentenceTransformer embeddings + 4 handcrafted text features.
- **Input Layer (Linear + LayerNorm):** Projects the combined features into a *d_model* dimension (default 256).
- **Multi-Head Attention:** 8 attention heads to capture multiple relationships between features, followed by a residual connection and LayerNorm.
- **Feed-Forward Network:** Expands and contracts the dimensionality (4×d_model → d_model) with **GELU** activations and dropout, followed by another residual connection and LayerNorm.
- **Classification Head (Linear + GELU + Linear):** Reduces feature dimension from *d_model* to *d_model/2*, then to *num_classes* with a softmax output.

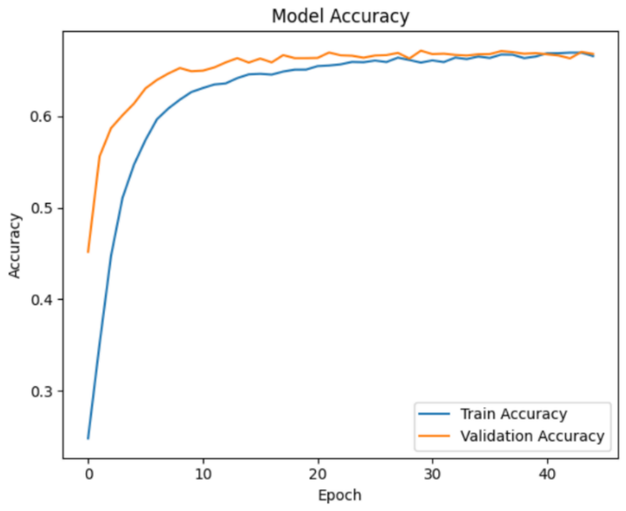
Results

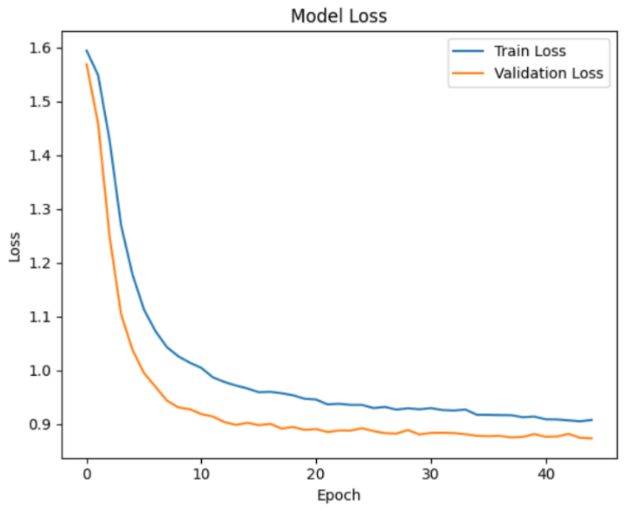
Classification Report:

```
            precision  recall  f1-score   support
        0       0.75      0.79      0.77       782
        1       0.88      0.83      0.85       996
        2       0.67      0.67      0.67       723
        3       0.69      0.66      0.68      1014
        4       0.81      0.85      0.83      1027
   accuracy                         0.77      4542
  macro avg       0.76      0.76      0.76      4542
weighted avg      0.77      0.77      0.77      4542
```

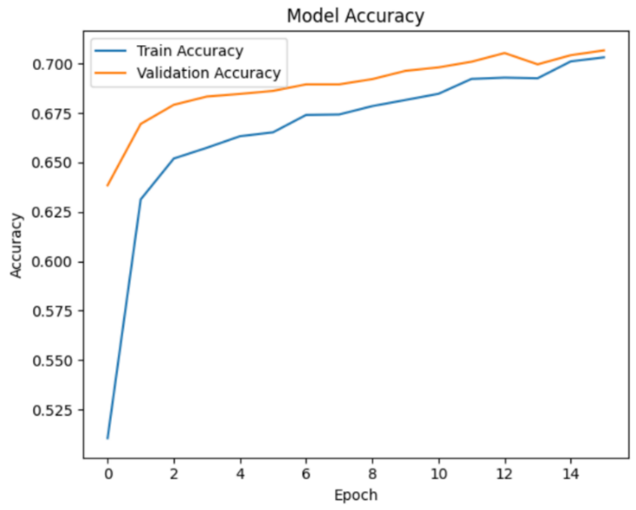
Visualization

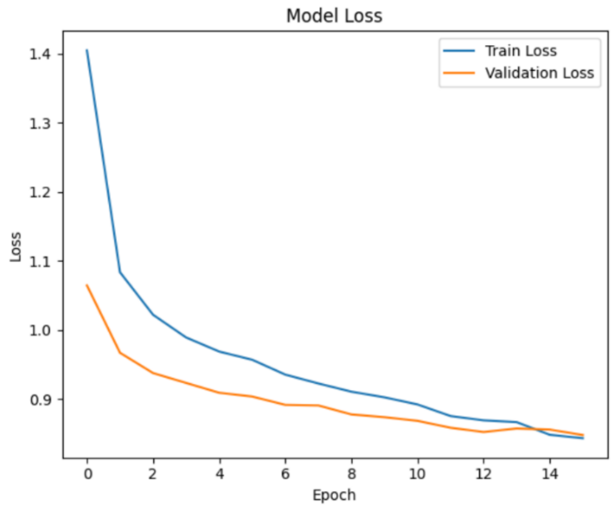
Word2Vec + Dense



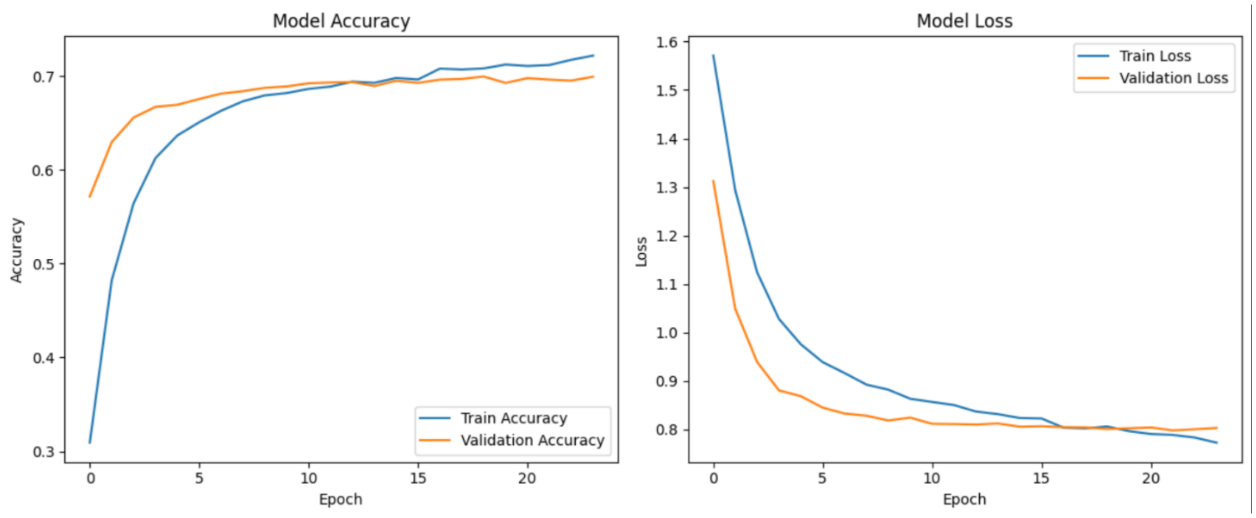


Glove Embedding + GRU & LSTM

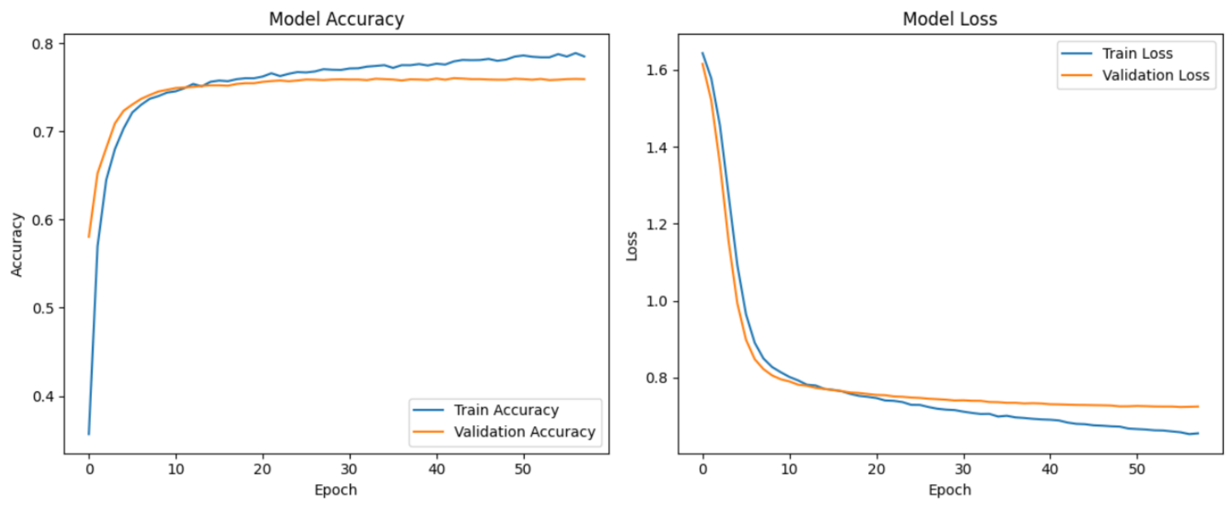




BERT Feature Extraction + Dense



Embedding all-mpnet-base-v2 + Dense



Conclusion

The Word2Vec + Dense model, the training accuracy is 68.19%, with a validation accuracy of 67.12%. The relatively small difference between these accuracies indicates that the model is not overfitting as severely as the LSTM model, but there is still room for improvement in its ability to generalize to new data. Adjustments to the network's architecture, such as increasing the model's complexity or exploring other pre-trained embeddings, could help enhance performance.

Glove Embedding + GRU & LSTM builds on pretrained embeddings and a dual recurrent architecture, improving validation accuracy to 70.65%, though it still shows a gap of about 6% from its training accuracy of 76.69%.

BERT Feature Extraction + Dense utilizes the deep contextual power of BERT, achieving 77.36% training accuracy and 70.32% validation accuracy. However, its 7% accuracy gap indicates that fine-tuning or additional regularization may improve generalization.

The Embedding all-mpnet-base-v2 + Dense model emerges as the best-performing approach, with a validation accuracy of 76.05% and a training accuracy of 78.59%. Its small accuracy gap of about 2.5% highlights excellent generalization and the strength of MPNet embeddings.

The Enhanced Transformer model incorporates attention and additional handcrafted features, achieving a training accuracy of 80.96% and a strong validation accuracy of 76.86%, showcasing the power of combining advanced embeddings with attention mechanisms.

Among the models, the Word2Vec + LSTM has the widest gap (about 6.5%), indicating its simpler embeddings and architecture are less effective for the task compared to more advanced approaches.

Both Glove Embedding + GRU & LSTM and BERT Feature Extraction + Dense provide incremental improvements, with better handling of semantics and sequential patterns, but they still fall short of the best-performing models.

The Embedding all-mpnet-base-v2 + Dense model is highly efficient, offering a balance between simplicity and performance by using pretrained embeddings and a straightforward dense architecture.

The Enhanced Transformer, while slightly more complex, pairs attention with sentence embeddings and handcrafted features, yielding competitive performance and slightly higher validation accuracy than the MPNet-based model.

When considering accuracy, generalization, and implementation complexity, the Embedding all-mpnet-base-v2 + Dense model stands out as the most practical and effective choice.

It achieves the highest validation accuracy among all models while maintaining a small training-validation gap, making it robust for unseen data.

While the Enhanced Transformer delivers similar performance, the additional complexity may not justify the marginal improvement over the Embedding all-mpnet-base-v2 + Dense model for many use cases.