CSE 556: Natural Language Processing Assignment 4

Sanskar Ranjan 2021096 | Saumil Lakra 2021097 | Jeremiah Malsawmkima Rokhum 2021533 | Vishal Singh 2021575

1. Evaluation for Task 1

1.1. MODEL 1: BERT MODEL

- Accuracy: 0.9583
- F1-Score: 0.9584

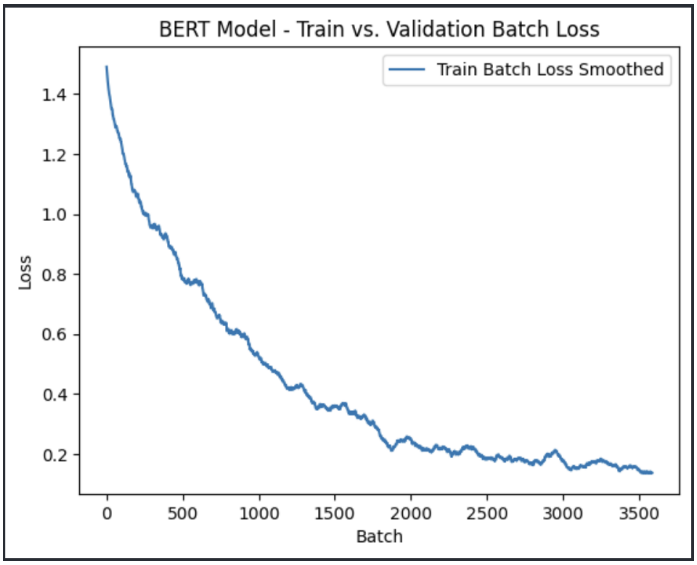

Figure 1. Training Loss of Task 1 - Model 1

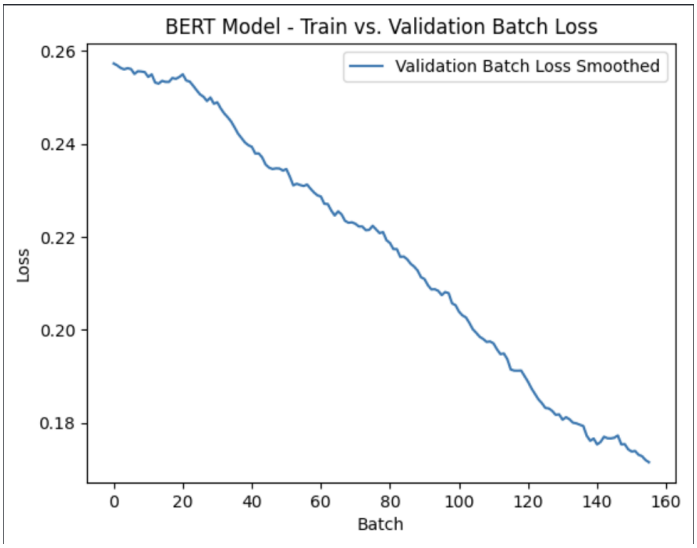

Figure 2. Validation Loss of Task 1 - Model 1

The plot shows a single line labeled "Train Batch Loss Smoothed," which means that the line represents the moving average of the training loss to smooth out the fluctuations for easier visualization of the trend. The line starts with a high loss value at around 1.4 and decreases sharply as the number of batches increases. This decline is quite steep initially, indicating rapid learning and model improvement. As the line moves towards the right, the decrease in loss slows down, showing a plateau around the value of 0.2 loss, which suggests that the model is converging to a minimum loss value and learning is stabilizing.

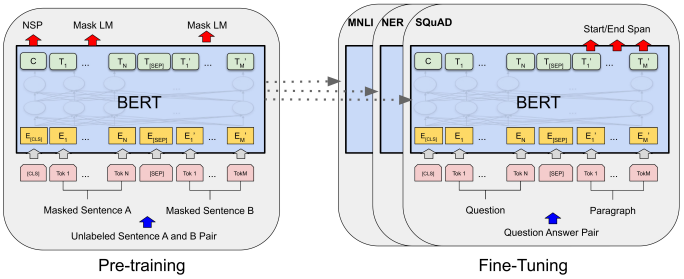

Figure 3. BERT architecture

The line represents the "Validation Batch Loss Smoothed," indicating that the plotted values are a smoothed version of the raw loss data from the validation batches. Smoothing is commonly done to make it easier to see trends by reducing noise. The plot shows a clear downward trend, indicating that as the BERT model is trained over more batches, its loss on the validation set is decreasing. This suggests that the model is learning and improving its performance in predicting or classifying the validation data over time. The smooth decline without any upward spikes suggests that the model is not overfitting during the batches shown.

1.1.1 Explanation of the Model, intuition behind the models, splits and everything relevant

- We took every conversation sentence as an input variable and its emotion as an output point.
- Input sentences were preprocessed(lowered text, removed stop words) and output were one hot encoded manually.
- We used Bert-tokenizer (bert-base-uncased) to tokenize the sentences.
- Classification model used was BertForSequenceClassification for the classification task and Cross Entropy was used as loss function
- Split the data (X_train, X_val, y_train, y_val) that allowed us for training on labeled examples (X_train, y_train) and evaluating on unseen data (X_val, y_val).

1.2. MODEL 2: Graph Convolutional Networks MODEL with GLOVE encoding

- Accuracy: 0.7933258753598925
- F1-Score: 0.7288189988599781

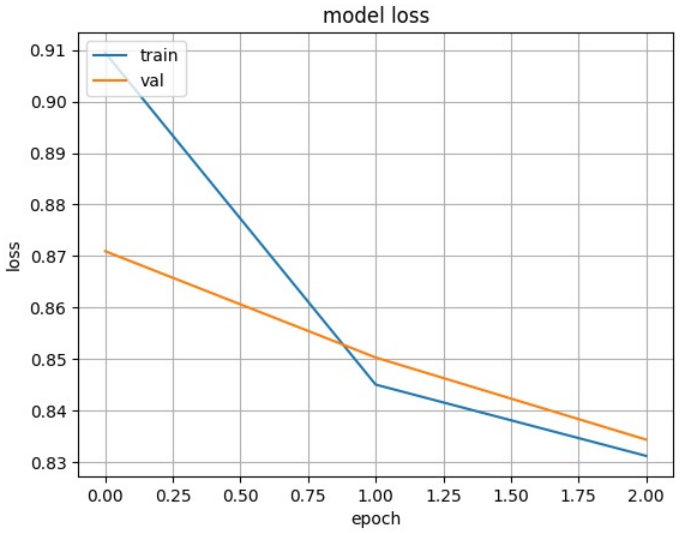

Figure 4. Training and Validation Loss against Epochs plots

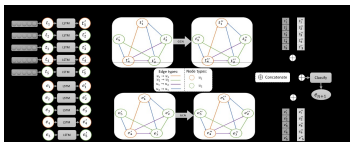

Figure 5. GCN architecture

Overall, the graph suggests that over the course of two epochs, both the training loss and validation loss are decreasing, which is a positive indication that the model is learning and generalizing well up to this point.

1.2.1 Explanation of the Model, intuition behind the models, splits and everything relevant

Base Concept: A GCN leverages the structure of a graph, which consists of nodes (vertices) and edges (connections), to learn a function that maps node features to an output. The network aims to learn a representation (embedding) of each node that captures not only its own features but also the features of its neighbors.

Input Representation:

- Each Utterance in a conversation is represented by a of text vectors and emotion vectors
- Text Vectors: Typically encoded using pre-trained word embeddings like GloVe.
- Emotion Vectors: Typically one-hot encoded vectors representing the emotion of the current utterance

Representation of this network as a graph:

- Nodes (Vertices): Each node represents an utterance equipped with initial information about what is said and the expressed emotion.
- Edges (Connections): These define which utterances are related or influence each other, such as consecutive utterances or multiple utterances by the same speaker.
- Graph Convolution: This operation updates each utterance's information by combining its own data with data from connected utterances, effectively capturing the conversational context and how emotions evolve throughout the dialogue.

This model excels in environments where the interplay between different speakers and the sequence of spoken emotions significantly impact the emotional trajectory of conversations. Such capabilities make it suitable for applications in empathetic response generation systems, emotional analysis of dialogues, and more dynamic human-machine interaction platforms.

1.3. Reason why Model 1 performed better

- BERT (Bidirectional Encoder Representations from Transformers) is designed to derive deep contextual relationships between words in a text. Unlike GLOVE embeddings that provide a fixed representation for each word regardless of its contextual usage, BERT dynamically generates word embeddings based on the words around them. This allows BERT to handle nuances in language better, like homonyms or context-specific meanings.
- BERT is pre-trained on a massive corpus of text data using tasks like masked language modeling and next sentence prediction. This extensive pre-training allows BERT to develop a nuanced understanding of language structure and context, which can be beneficial for a wide range of downstream tasks like sentiment analysis, question answering, and language inference.
- With BERT, you can fine-tune the entire model on a specific task with relatively small datasets and still achieve high performance. This is because the pre-trained model has already learned a significant amount of language understanding. In contrast, using GCN with GLOVE embeddings might require more task-specific tuning and additional layers specifically tailored to integrate the graph structure and the embeddings effectively.
- BERT's architecture allows it to adapt to a wide variety of tasks without major changes to its structure. It can handle tasks ranging from classification to generation simply by adjusting the output layers and fine-tuning. GCN, being inherently designed for graph-structured data, is specialized for tasks where data relationships are explicitly structured as graphs (e.g., social networks, molecular structures).
- BERT benefits greatly from transfer learning, where knowledge from one task can assist in another. This is less straightforward with GLOVE embeddings used in a GCN architecture, where the embeddings are static and the model's adaptability is more constrained to the specifics of graph-based learning.

2. Evaluation for Task 2

2.1. MODEL 1: BERT MODEL

- Accuracy: 0.8482
- F1-Score: 0.7785

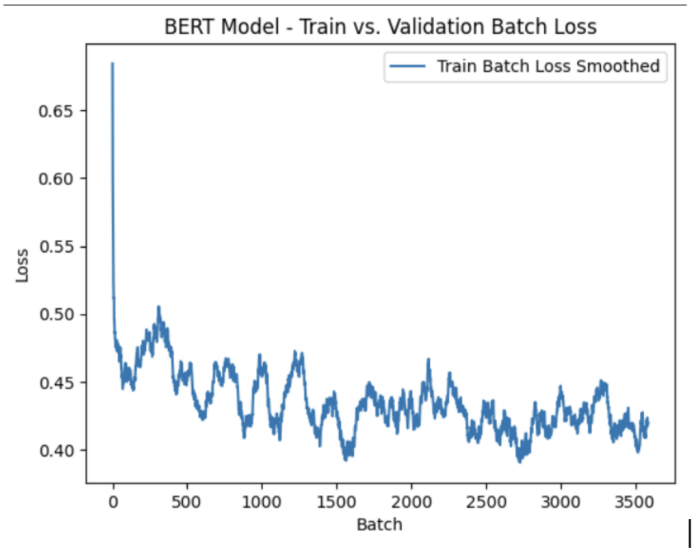

Figure 6. Training Loss of Task 2 - Model 1

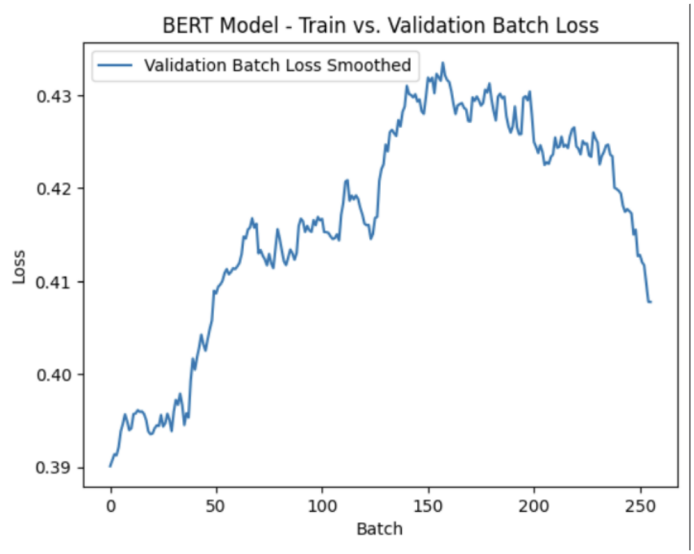

Figure 7. Validation Loss of Task 2 - Model 2

The line in the graph, labeled "Train Batch Loss Smoothed," suggests that it is a smoothed version of the raw loss data, which typically involves averaging the loss over several batches to reduce the noise and make trends clearer. The line starts off quite high at the very beginning, which is normal as the model begins learning from an untrained state. It quickly drops and then continues to oscillate as the model iteratively updates its parameters. The general trend is a decrease in loss, indicating learning and improvement over time.

The line on the graph is labeled "Validation Batch Loss Smoothed", which suggests it shows a smoothed loss curve over the validation dataset. The line shows a general increasing trend in loss, starting from below 0.40 and reaching peaks of just above 0.42. Towards the end of the graph, there is a sharp decrease in loss, indicating an improvement in model performance on the validation set at those batches.

2.1.1 Explaination of the Model, intuition behind the models, splits and everything relevant

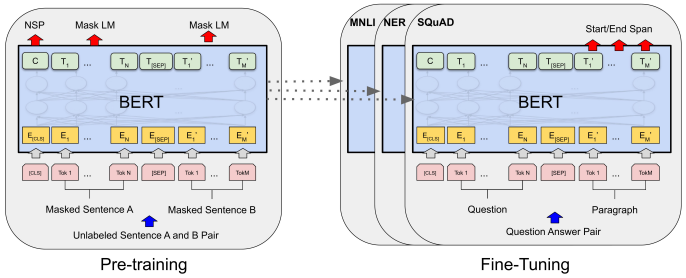

Figure 8. BERT architecture

- **BERT Model**: BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based machine learning model for natural language processing pre-trained on a large corpus of text. BERT models both the left and right context in all layers, making it deeply bidirectional. The model effectively captures the context and relationships of words in sentences through its attention mechanisms, where different words are weighted differently based on the task.

- **BERT for Sequence Classification**:BertForSequenceClassification is a BERT model with a sequence classification/regression head on top (a linear layer on top of the pooled output). In your case, it's used for classifying whether an utterance will cause an emotional flip, i.e., a significant change in the emotional tone of a conversation.

- **Tokenization(Input Processing) :**BERT requires text to be tokenized in a specific way. In this model, BertTokenizer is used to tokenize the sentences. This tokenizer converts each token into an ID that BERT can process.

- **Padding and Truncation(Input Processing)** : The sentences are padded to a maximum length to ensure consistent input size. Truncation ensures that any sentence longer than the maximum length is appropriately shortened.

- **Tensors and DataLoaders(Input Processing)** : After tokenization, sentences are converted into tensors that the model can process. The attention mask tensor is also created here, which helps the model differentiate between the meaningful data and the padding. The DataLoader handles batches of these tensors, making it efficient to process larger amounts of data.

- **Training and Validation**: The data is split into training and validation sets. The training set is used to train the model, and the validation set is used to evaluate the model's performance on unseen data. During training, the model uses the

CrossEntropyLoss function, which is suitable for classification tasks with multiple classes. The loss function measures the model's predictions against the true labels and updates the model's weights through backpropagation. The AdamW optimizer is used, which is an extension of the Adam optimizer with improved handling of weight decay.

- **Epochs and Iterations**: The model trains across multiple epochs. In each epoch, it iterates over all batches in the training set, updating weights with each batch. It then evaluates its performance on the validation set. Performance metrics (loss) are collected and printed for each epoch, showing how the model improves over time.
- **Evaluation**: After training, you might typically want to plot the training and validation loss to visually assess convergence and overfitting.
- **Architectural Highlights:** Bidirectional Nature: BERT's bidirectional nature allows it to understand the context of a word based on all other words in a sentence, rather than just the words before it as in traditional recurrent neural networks. Self-Attention Mechanisms: These allow the model to weigh words differently based on their relevance to the task, providing a nuanced understanding of sentence structure and meaning. Fine-Tuning for Specific Tasks: While BERT is pre-trained on general language understanding, fine-tuning it on specific tasks like flip detection allows it to apply its broad language understanding capabilities to specialized problems.

2.2. Model 2: Custom LSTM Architecture

- Accuracy: 0.8438
- Macro F1-Score: 0.63
- Weighted F1-Score: 0.83

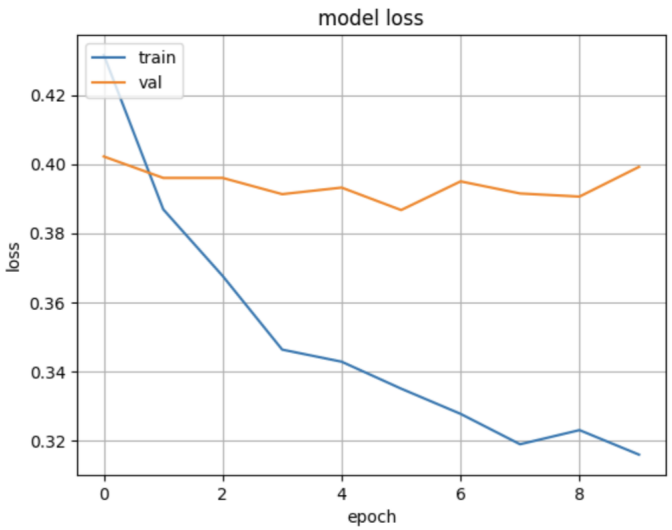

Figure 9. Training and Vaildation Loss of Task 2 - Model 2

The blue line shows a significant decrease from the first epoch and continues to decline, albeit at a slower rate, as the number of epochs increases. This trend suggests the model is learning from the training data and is able to reduce its prediction error over time. The orange line for validation loss, however, starts higher than the training loss, decreases slightly, but then largely stabilizes around a loss value of 0.38. It shows some variability, but no clear downward trend as with the training loss. Since the validation loss does not increase and the gap is not widening significantly, the model could still be performing acceptably.

2.2.1 Explanation of the Model, Intuition behind the models, splits and everything relevant

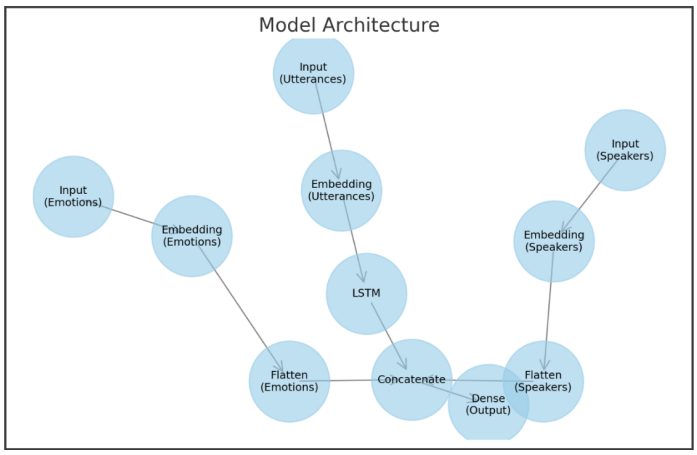

Figure 10. Custom Model Architecture

The model we've constructed is designed to predict whether a given utterance (and its associated context) will trigger a specific emotional response in the future. This is typically used in applications such as dialogue systems, emotional analysis in conversational AI, or any scenario where understanding emotional triggers is crucial.

- **Input Data**(Utterances): Text data that has been tokenized and padded to create sequences of uniform length.
- **Input Data**(Speakers and Emotions): Categorical data that are encoded to numerical values to represent different speakers and emotional states in the dialogue.
- **Embedding Layers**(Utterance Embedding): Transform tokenized text into dense vectors which capture semantic meanings more effectively than raw tokens.
- **Embedding Layers**(Speaker and Emotion Embeddings): Convert categorical data into a format that can be processed alongside textual data, capturing nuances related to who is speaking and the emotional context.
- **LSTM Layer**: Processes the sequence of embedded utterances, capturing temporal dependencies and context within the dialogue. LSTM is particularly useful for sequences where past information (like the flow of a conversation) is crucial for understanding the current state or predicting the next state.
- **Concatenation**: The output of the LSTM layer (which contains learned features from the utterance sequence) is concatenated with flattened speaker and emotion embeddings.

This step is crucial as it combines textual information with contextual cues about the speaker and the emotional tone.

- **Output Layer**: A fully connected layer with a sigmoid activation function makes the final prediction. It outputs a probability indicating whether the current context and utterance will lead to a specific emotional trigger.
- **Prediction Mechanism**: The model predicts whether an utterance will trigger a future emotion by analyzing the text of the utterance in conjunction with who is speaking and the current emotional context.
- **Contextual Understanding**: By including speaker and emotion data, the model doesn't just consider what is said, but also who says it and in what emotional state. This context is often critical in human interactions where the same phrase might be interpreted differently depending on these factors.
- **Temporal Sequence Analysis**: The LSTM layer allows the model to remember aspects of the conversation that precede the current utterance. This memory helps the model understand the progression of the conversation, which is often key to predicting emotional responses.
- **Integration of Multiple Data Types**: By embedding different types of data (text, speaker identity, and emotions) and combining them, the model leverages a more holistic view of the dialogue. This integration helps in making more accurate predictions about emotional triggers.
- **Training and Evaluation**: The model is trained on labeled data (where labels indicate whether an emotional trigger occurs). During training, it learns to correlate the input data with these labels.
- The accuracy of predictions and the model's ability to generalize are typically evaluated using a validation set, which helps in tuning the model to avoid **overfitting** and **underfitting**.
- This architecture is effective for scenarios where the outcome (emotional trigger) depends heavily on both the **content of the dialogue** and the **context** in which it is delivered. It's **well-suited** for advanced conversational AI applications, such as virtual assistants, automated support systems, or any platform where understanding and responding appropriately to human emotions is beneficial.

2.3. Reason why Model 2 performed better in Task 2

Impact of Limited Input Features

- **Reduced Contextual Understanding**: One of the key strengths of BERT lies in its ability to integrate and interpret vast amounts of contextual information. By limiting the input to just the utterances and triggers, the model misses out on potentially critical information that could influence the understanding of the conversation's dynamics. In conversations, who says something (speaker identity) and in what emotional state they say it (emotional state) can drastically change the meaning and reception of the same words.
- **Potential Bias and Generalization Issues**: Without speaker and emotion data, the model might develop biases based on the training data's content alone. This could lead to poorer generalization when exposed to new data where the dynamics of speaker interaction and emotional variance are different from the training set.
- **Sensitivity to Subtext and Nuance**: Conversations often involve subtleties and implicit meanings that can be significantly influenced by the speaker's identity and their emotional tone. Without this data, the model might miss out on these nuances, potentially leading to misclassifications or an inability to detect finer emotional shifts.

Comparing to a Full-Context Model, a model that incorporates speaker and emotion data alongside the utterance might have several advantages:

- **Richer Feature Set**: By including speakers and emotions, the model can capture more complex patterns of interaction and more accurately model the dynamics of conversational exchanges. This can be particularly important in applications like mental health analysis, customer service, and other domains where understanding the interplay of emotions and identities is crucial.
- **Improved Accuracy and Fidelity**: The additional context provided by speaker and emotion data can lead to a more accurate and nuanced understanding of the conversation, potentially improving the performance in tasks like sentiment analysis, emotion detection, or flip detection.
- **Better Handling of Diverse Data**: Including a broader range of contextual features makes the model more robust to variations in new data, enhancing its ability to generalize across different conversational settings or demographic groups.

1. **Saumil Lakra:** Task1 Model 1 & Task2 Model 1
2. **Jeremiah Rokhum:**Task1 Model 2 & Task2 Model
3. **Sanskar Ranjan:** Task1 Model 2 & Task2 Model 2
4. **Vishal Singh:** Task1 Model 1 & Task2 Model 1