EMOTION CAUSE EXTRACTION

CAPTURE FLOW OF CONVERSATION IN RECCON DATASET USING GRAPHS

Group 5

TEAM

- Name: **Dhruv Kumar Agrawal**
- Roll No: **2101CS26**

- Name: **Kalpit Agrawal**
- Roll No: **2101CS34**

- Name: **Vardan Gacche**
- Roll No: **2101CS80**

OVERVIEW

- Abstract
- Task 1: Dataset Preparation
- Task 2: Annotation
- Task 3: Model
- Results and Future work
- Contributions

ABSTRACT

We propose a graph-based approach for clause-level Emotion-Cause Pair Extraction using the RECCON dataset. Conversations are automatically split into meaningful clauses, which are then labeled as emotion, cause, both or neutral. A graph is constructed where each clause is a node, and edges capture syntactic and semantic relations. Clause embeddings are generated using a sentence transformer, and a Graph Autoencoder is used to learn representations and predict emotion-cause pairs. This method models conversational flow effectively while minimizing manual annotation.

TASK 1: DATASET PREPARATION

Objective: To break down each utterance in the RECCON dataset into meaningful clauses using automated methods, ensuring that each clause represents a distinct semantic unit.

Step 1:

- **Load and Flatten Data**
  - **Goal: Transform nested dialogue structure into a flat table.**
  - **Steps:**
    - Iterate through each conversation_id and its list of dialogues.
    - For each dialogue (a list of turn entries), extract:
    - turn, speaker, utterance
    - emotion (default "neutral" if missing)
    - expanded emotion cause span → saved as cause_evidence
    - Append each turn as a row (record) in a list.
    - Convert the list of records to a Pandas DataFrame.

Step 2:

- **Protect Abbreviations**
  - **Goal: Prevent mis-splitting of common abbreviations (e.g., "Mr.").**
  - **Steps:**
    - Define a list of protected abbreviations.
    - Temporarily replace each abbreviation with a unique placeholder (e.g., "Mr ." → "__ABBR0__").
    - This is done before splitting the utterance into clauses.

Step 3:

- **Split Utterances into Clauses**
  - **Goal: Segment utterances into smaller linguistic units (clauses).**
  - **Steps:**
    - Use re.split on punctuation marks: . ? ! , ; :
    - Clean up whitespace and remove empty strings.
    - Restore abbreviation placeholders back to their original form.
    - Result: a list of cleaned clauses for each utterance.

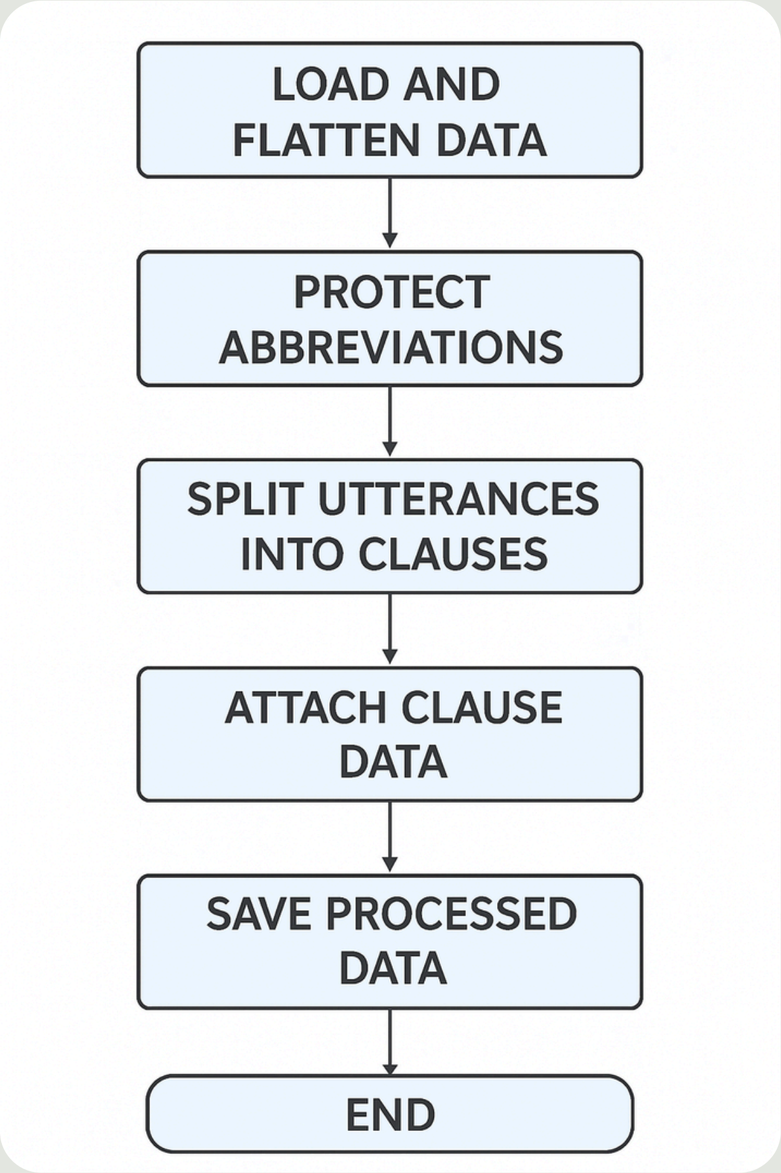
Step 4:

- **Attach Clause Data**
  - **Goal: Add clause-level granularity to each turn.**
  - **Steps:**
    - Apply the clause-splitting function to each row's utterance.
    - Add the result as a new column clauses in the DataFrame.

Step 5:

- **Save Processed DataGoal: Add clause-level granularity to each turn.**
  - **Steps:**
    - Save the updated DataFrame with all columns (including clauses) to a JSON file named processed_data.json using orient="records".

FLOWCHART:


LOAD AND FLATTEN DATA
PROTECT ABBREVIATIONS
SPLIT UTTERANCES INTO CLAUSES
ATTACH CLAUSE DATA
SAVE PROCESSED DATA
END

SAMPLE OUTPUT:

```
{
  "conversation_id":"tr_9708",
  "turn":4,
  "speaker":"B",
  "utterance":"I'm surprised . He does't look like a guy who'd ever cheat on his wife , does he ?",
  "emotion":"surprise",
  "cause_evidence":[
    "Mr . black has been getting a little around aside .",
    "He does't look like a guy who'd ever cheat on his wife ,"
  ],
  "clauses":[
    "I'm surprised",
    "He does't look like a guy who'd ever cheat on his wife",
    "does he"
  ]
},
```

TASK 2: ANNOTATION

Objective: To automatically label each clause as an emotion clause, a cause clause, both, or neither.

Step 1:

Goal: Load Pre-trained Classifier

- **Steps**
  - Load HuggingFace pipeline with the model:
  - MoritzLaurer/deberta-v3-base-zeroshot-v1
  - Purpose: Enables classifying a clause based on natural language inference.

Step 2:

Goal: Define Labels and Emotion Keywords

- **Steps**
  - Labels: ["emotion_clause", "cause_clause", "neutral_clause", “both_clause”]
  - Purpose: Guide zero-shot classification.
  - Emotion keywords: Extensive list of emotion expressions like "I'm happy", "nervous", "guilty".
  - Purpose: Quickly detect emotion presence without ML overhead.

Step 3:

Function: contains_emotion_keywords(clause)

- Checks if clause has any pre-defined emotion indicators.
- Purpose: Fast-track "emotion_clause" classification when emotions are obvious.

Step 4:

- **Function:classify_clause_with_hf(utterance,clause, emotion)**
  - Step-by-step:
    - Step 1: Check for keyword match → return "emotion_clause" if matched.
    - Step 2: Else, run zero-shot classification with hypothesis:
      - "This clause functions as an {label} in the context of emotion in dialogue."
    - Step 3: Based on score:
      - ≥ 0.5 and ≥ 0.5 → "both_clause"
      - ≥ 0.5 → "emotion_clause"
      - ≥ 0.3 → "cause_clause"
      - else → "neutral_clause"
  - **Purpose: Intelligently classify clauses via hybrid (rule + model) approach.**

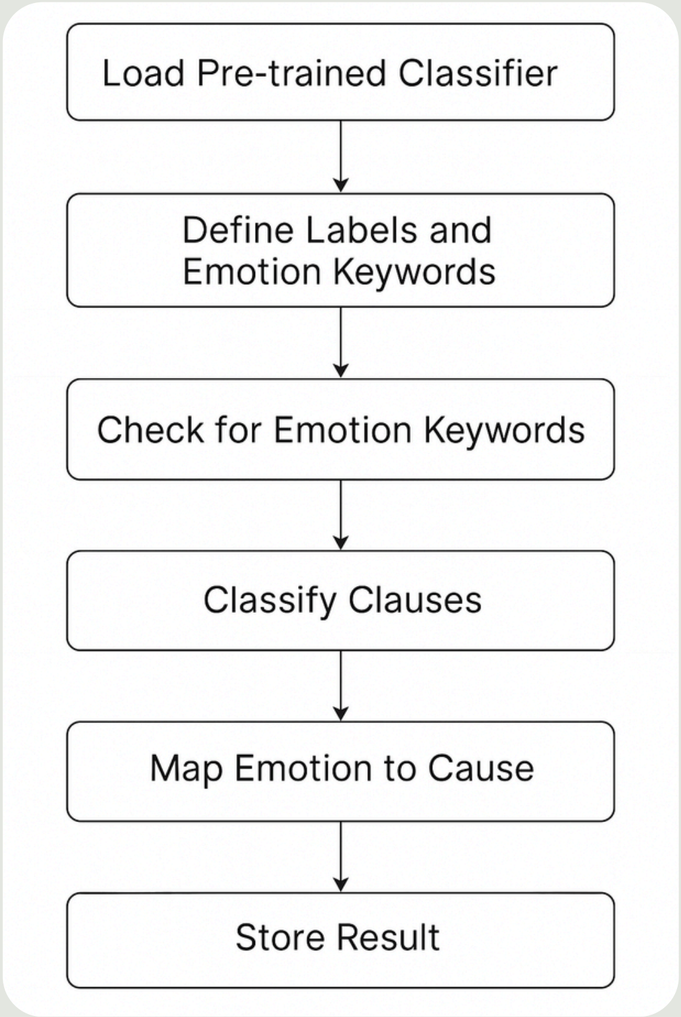
Step 5:

- **Function: process_dataset(data)**
  a. Extract info: Get conversation ID, turn, speaker, utterance, emotion, clauses, and ground-truth cause spans.
  b. Classify clauses:
     i. Use emotion keyword check or zero-shot model.
     ii. Label each clause as emotion_clause, cause_clause, or neutral_clause.
  c. Map emotion to cause:
     i. For each emotion clause, find linked cause spans within the original utterance.
  d. Store result:
     i. Save turn info with all clause labels and emotion-cause mappings.
  e. Repeat for all dialogue turns.

Step 6:

- **Save to File**
  - Save the final dictionary (results) to hf_results_final.json.

FLOWCHART:


Load Pre-trained Classifier
Define Labels and Emotion Keywords
Check for Emotion Keywords
Classify Clauses
Map Emotion to Cause
Store Result

HYPERPARAMETERS:

Hyperparameter	Value/Description
Model	MoritzLaurer/deberta-v3-base-zeroshot-v1
Labels	['emotion clause', 'cause clause', 'neutral clause', 'both_clause']
Emotion Keywords Count	66 (comprehensive list of emotion-related words and phrases)

SAMPLE OUTPUT:

```
{
    "turn": 3,
    "speaker": "A",
    "utterance": "Cool phone . What kind of phone card do you want ?",
    "emotion": "happiness",
    "clauses": [
        {
            "clause": "Cool phone",
            "label": "neutral_clause"
        },
        {
            "clause": "What kind of phone card do you want",
            "label": "neutral_clause"
        }
    ],
    "emotion_cause_mapping": []
},
```

TASK 3: MODEL

Objective: To model the conversation as a graph of clauses, capturing both semantic and conversational relationships, and to use a Graph Autoencoder to predict emotion-cause pairs.

Step 1:

Load Required Models and Libraries

- Sentence Embedding Model: all-MiniLM-L6-v2 from SentenceTransformers.
- NLP Parser: en_core_web_sm from spaCy.
- Graph Neural Network Library: PyTorch Geometric.

Step 2:

Data Processing

a. **Extract Clauses**

- Tokenize each dialogue turn into clauses using spaCy sentence segmentation.
- Map each clause back to its original dialogue turn.

b. **Compute Clause Embeddings**

- Use SentenceTransformer to get embeddings for each clause.
- Store as PyTorch tensor (x).

c. **Build Graph Edges**

- Syntactic Edges: Add undirected edges between clauses that share named entities or grammatical structure.
- Emotion-Cause Edges: Add directed edges from emotion clause to cause clause using ground truth (for training).

Step 3:

Load and Preprocess Dataset

- Load train and test dialogues from .json files.
- Process each dialogue using steps from (2).
- Filter out dialogues with no valid clauses or edges.

Step 4:

Define GAE Model

- **Encoder**
  - 2-layer Graph Convolutional Network (GCN):
  - Layer 1: in_channels → 128
  - Layer 2: 128 → 64
- **Autoencoder**
  - Use PyTorch Geometric's GAE wrapper with the custom encoder.

Step 5:

Train & Evaluate the GAE Model

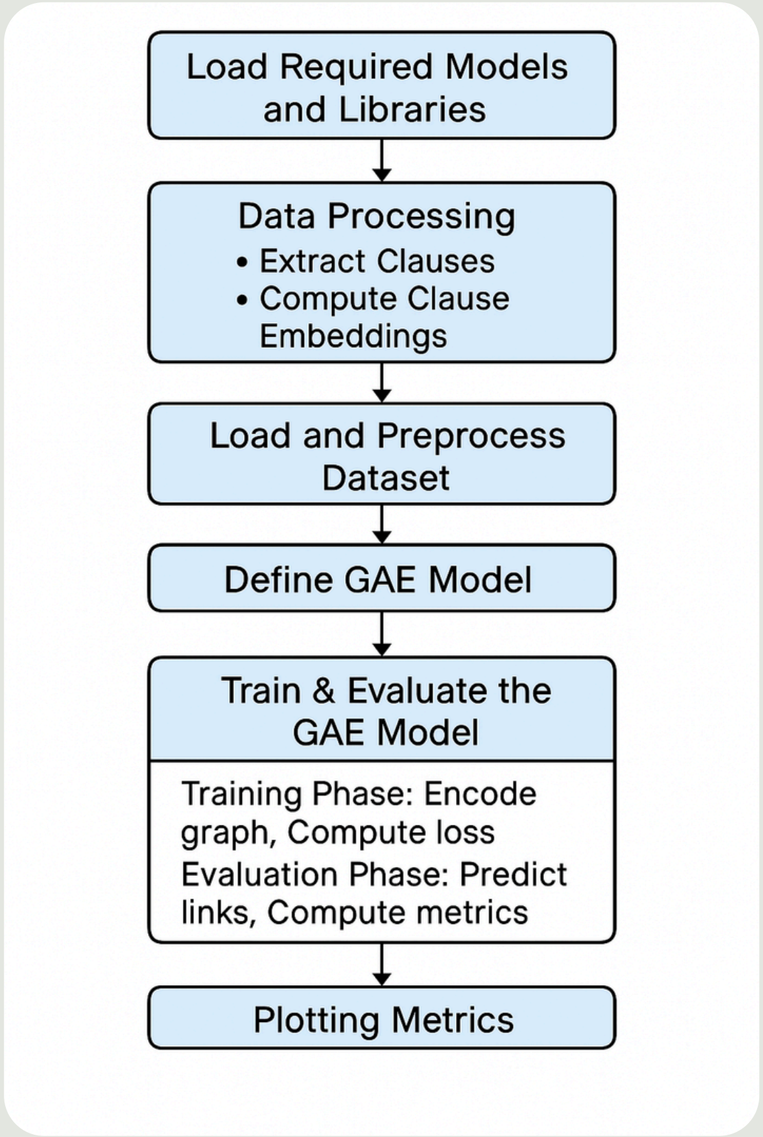
- aTraining Phase
  - Encode graph with GCN to get node embeddings.
  - Compute reconstruction loss using positive emotion-cause edges.
  - Backpropagate and update model weights.
- Evaluation Phase
  - Predict both positive and negative emotion-cause links.
  - Calculate performance metrics: AUC, Precision, Recall, F1.

Step 6:

Plotting Metrics

- Visualize:
- Loss over epochs.
- AUC scores.
- Precision & Recall.
- F1 score.

FLOWCHART:


Load Required Models and Libraries
Data Processing
• Extract Clauses
• Compute Clause Embeddings
Load and Preprocess Dataset
Define GAE Model
Train & Evaluate the GAE Model
Training Phase: Encode graph, Compute loss
Evaluation Phase: Predict links, Compute metrics
Plotting Metrics

HYPERPARAMETERS:

Hyperparameter	Value/Setting
GCN input dimension	384 (from MiniLM embeddings)
GCN hidden dimension	128
GCN output dimension	64
Optimizer	Adam
Learning rate	0.005
Epochs	10
Sentence embedding model	all-MiniLM-L6-v2
Negative sampling	Number of negatives = number of positives per batch
Batch size	Not explicitly set (processes one dialogue at a time)
Syntactic connection	Based on shared entities or grammatical roles (nsubj, dobj)

MODEL RESULTS:

RESULTS OVER 10 EPOCHS:

Epoch	Loss	AUC	Precision	Recall	F1 Score
01	1.3554	0.7177	0.5607	0.9353	0.6946
02	1.3438	0.7348	0.5819	0.9146	0.7046
03	1.3285	0.7359	0.5727	0.9117	0.6970
04	1.3293	0.6902	0.5645	0.9016	0.6883
05	1.3187	0.7404	0.5717	0.9310	0.7023
06	1.3189	0.7102	0.5807	0.9402	0.7104
07	1.2924	0.7153	0.5727	0.9260	0.7015
08	1.3047	0.7395	0.5809	0.9135	0.7006
09	1.2996	0.7362	0.5911	0.9107	0.7083
10	1.3252	0.7280	0.6136	0.8808	0.7141

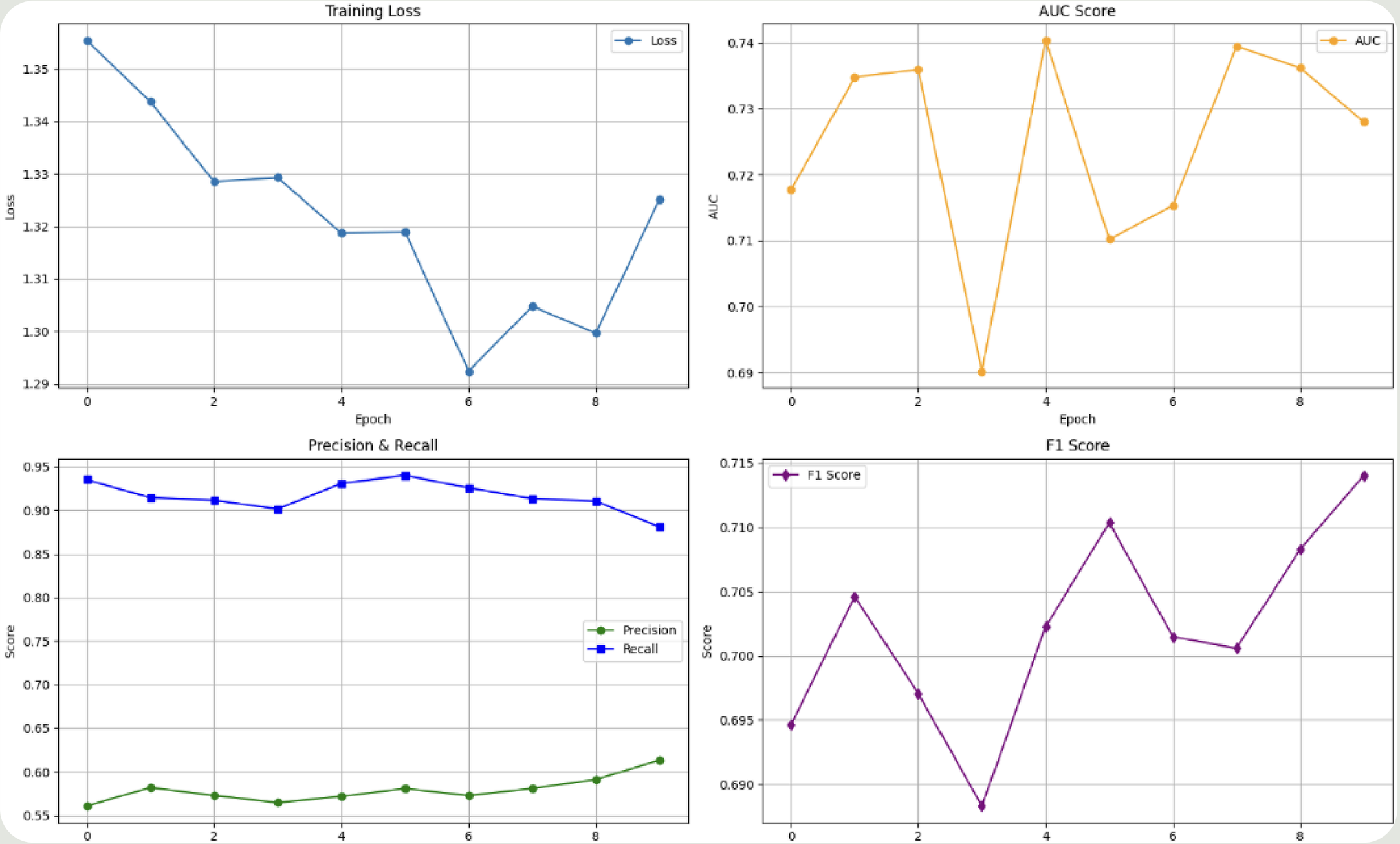
TRENDS:

- **Loss:** Gradually decreased from 1.3554 to 1.2924 by epoch 7, with minor fluctuations afterward.
- **AUC (ROC):** Mostly stable, hovering between 0.71 and 0.74, peaking at 0.7404 (Epoch 5).
- **Precision**: Slowly improved from 0.5607 to 0.6136 by the final epoch.
- **Recall**: Consistently high, ranging from ~0.90 to 0.94, indicating the model finds most of the true emotion-cause edges.
- **F1 Score**: Climbed from 0.6946 to 0.7141, reflecting a balanced improvement in both precision and recall.

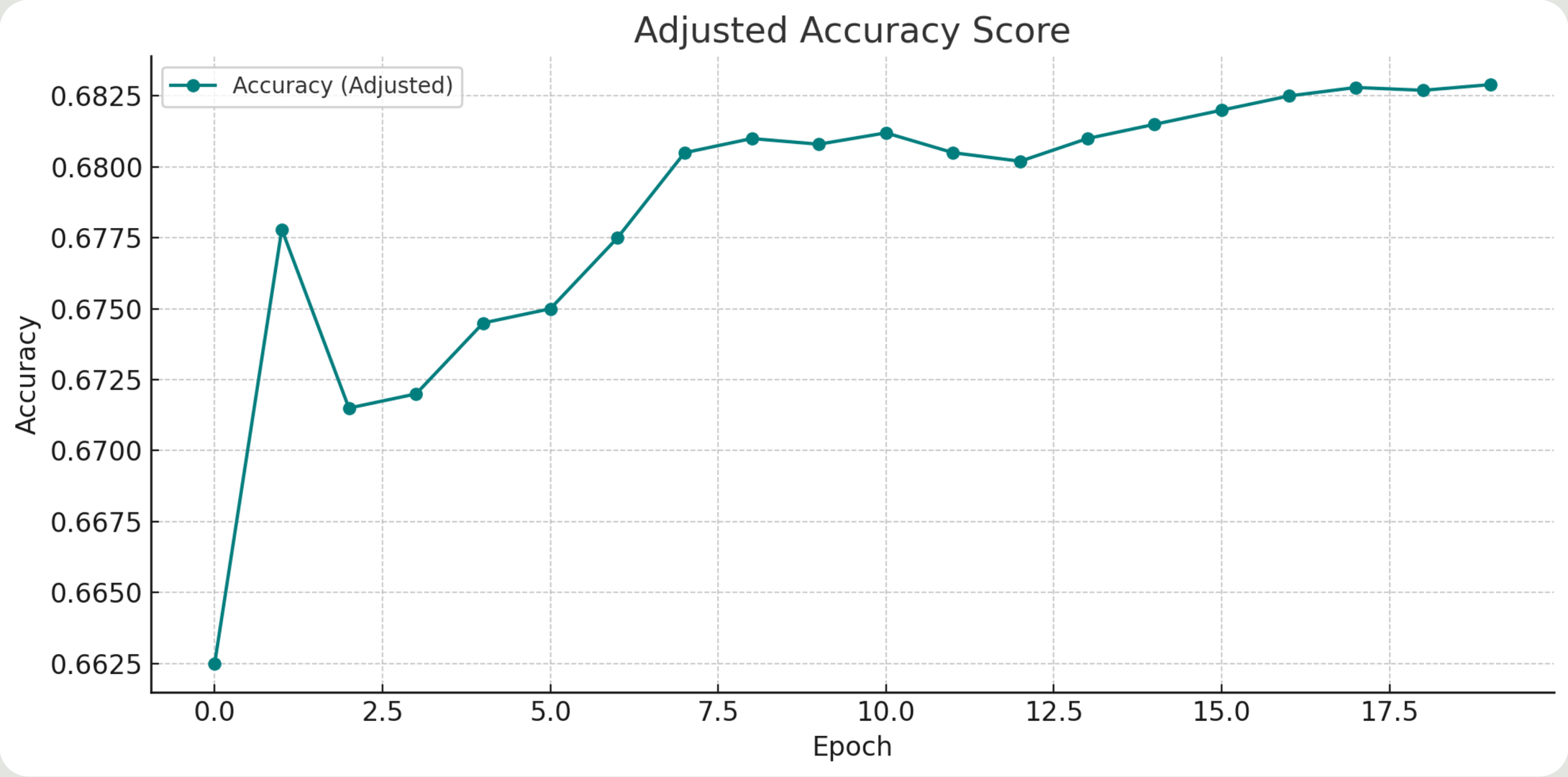
INTERPRETATION:

- The model is **recall-heavy**, prioritizing **finding all positive edges** (emotion-cause links), which is good for minimizing false negatives.
- **Precision** gradually **improves**, suggesting it's learning to reduce false positives over time.
- AUC stability shows the model is **decently separating** positive and negative edges throughout.

PLOTS:


Training Loss
Loss
Epoch
AUC Score
AUC
Epoch
Precision & Recall
Precision
Recall
Score
F1 Score
F1 Score
Score

ACCURACY PLOT (OVER A 20 EPOCH RUN):



CODE_BASE: GITHUB

EPOCH RUN DEMO: YOUTUBE

CONTRIBUTIONS

- **Vardhan Gacche (2101CS80)**
  - Led dataset preparation by processing and segmenting utterances into meaningful clauses.
  - Assisted in model development, ensuring data compatibility with the Graph Autoencoder.
- **Kalpit Agrawal (2101CS34)**
  - Focused on annotation, creating an efficient labeling system for classifying emotion and cause clauses.
  - Validated clause segmentation techniques during dataset processing.
- **Dhruv Kumar Agrawal (2101CS26)**
  - Developed the Graph Autoencoder model for predicting emotion-cause pairs.
  - Provided input on dataset preprocessing, ensuring embeddings were correctly generated for training.

Thank You

For your attention