Phase 2 Report: Emotion Drift in Customer Support AI

1. Project Identification

Project Title: Emotion Drift in Customer Support AI

Student: Noah Meduvsky

2. Groups

- **Noah Meduvsky** - noahmeduvsky@oakland.edu

3. Data Availability

Access to Data

The project utilizes publicly available conversational datasets that contain multi-turn dialogues with emotion labels for each utterance. These datasets are not specifically labeled for emotion drift, but rather provide emotion annotations that allow the project to detect and analyze emotion drift patterns across dialogue sequences. The primary datasets include:

DailyDialog: Available through Hugging Face datasets library

- Access method: `load_dataset("daily_dialog")`
- Contains daily conversations with emotion labels for each utterance
- Emotions mapped to: no_emotion, anger, disgust, fear, happiness, sadness, surprise

EmotionLines: Available through Hugging Face or GitHub repositories

- Access method: `load_dataset("emotionlines")`

- Multi-party conversations with emotion annotations
- Provides rich emotional context across dialogue turns

MELD (Multimodal EmotionLines Dataset): Available through research repositories

- Extends EmotionLines with multimodal features
- Includes textual emotion labels suitable for this project

Data Access Implementation

The data loading functionality is implemented in `src/data_loader.py`, which provides: - Automated dataset downloading via Hugging Face datasets library - Local caching to avoid repeated downloads - Functions to combine multiple datasets for increased sample diversity - Export to CSV/JSON formats for offline processing

Data Challenges and Solutions

Challenge 1: Label Imbalance - Some emotions (e.g., anger, sadness) may be underrepresented compared to neutral emotions - **Solution**: Implement class weighting and oversampling techniques in preprocessing pipeline

Challenge 2: Long-term Emotional Dependencies - Emotional context spans across multiple dialogue turns - **Solution**: Use sequence models (LSTM/Transformer) that capture temporal dependencies

Challenge 3: Dataset Format Variations - Different datasets use different emotion label schemes - **Solution**: Implement emotion label normalization to map all datasets to consistent emotion taxonomy

4. Data Preparation

Data Structures

The project works with **sequential text data** in the form of multi-turn dialogues. Each dataset is structured as:

```
dialogue_id | turn_id | speaker | text | emotion
```

- **dialogue_id**: Unique identifier for each conversation
- **turn_id**: Sequential turn number within the dialogue
- **speaker**: Speaker identifier (user/agent or user/AI)
- **text**: Utterance text content
- **emotion**: Emotion label for the utterance

Data Cleaning

The preprocessing pipeline (`src/preprocessing.py`) implements the following cleaning steps:

Text Normalization:

- Removal of URLs and email addresses
- Whitespace normalization
- Optional lowercasing (configurable)

Missing Value Handling:

- Empty or NaN text fields are replaced with empty strings
- Missing emotion labels default to 'neutral'

Special Character Handling:

- Standard text cleaning while preserving punctuation that may carry emotional meaning

Data Transformation

Tokenization:

- Uses BERT or RoBERTa tokenizers from Hugging Face Transformers
- Converts text to token IDs with attention masks
- Handles variable-length sequences with padding and truncation
- Maximum sequence length: 128 tokens (configurable)

Emotion Label Encoding:

- Maps emotion strings to numerical indices using LabelEncoder
- Maintains mapping dictionary for inverse transformation
- Normalizes emotion labels across datasets to consistent taxonomy:
  - joy, anger, sadness, fear, disgust, surprise, neutral

Sequence Preparation:

- Groups utterances by dialogue_id to maintain conversation context
- Sorts by turn_id to preserve temporal order
- Creates sequences suitable for sequence model input

Data Normalization

Emotion Label Normalization:

- Standardizes emotion labels across different datasets
- Maps variations (e.g., "happy", "happiness" $\rightarrow$ "joy")
- Ensures consistent emotion taxonomy

Class Balancing (Optional):

- Implements oversampling for minority emotion classes
- Computes class weights for imbalanced datasets
- Can apply undersampling as alternative approach

Data Augmentation

For future phases, potential augmentation strategies include: - Synonym replacement to increase text diversity - Paraphrasing while preserving emotional content - Synthetic emotion shifts for training drift detection

5. Design Specifications

ML Pipeline Architecture

The complete machine learning pipeline consists of the following stages:

Stage 1: Data Collection - Download datasets (DailyDialog, EmotionLines, MELD) from Hugging Face datasets library - Load datasets using automated download and caching mechanisms - Store processed data locally in CSV/JSON format for offline processing

Stage 2: Preprocessing - Text Cleaning: Remove URLs, normalize whitespace, handle special characters - Tokenization: Use BERT/RoBERTa tokenizers to convert text to token IDs - Emotion Label Normalization: Map emotion labels across datasets to consistent taxonomy - Sequence Grouping: Group utterances by dialogue_id to maintain conversation context - Class Balancing: Apply oversampling or class weighting if needed for imbalanced emotions

Stage 3: Feature Extraction - Generate BERT/RoBERTa embeddings to create contextual word representations - Encode sequences while maintaining temporal order of dialogue turns - Format input as tensors: [batch*size, seq*len, embedding_dim]

Stage 4: Model Training - Base Models: Train BiLSTM (Bidirectional LSTM) and Transformer (BERT/RoBERTa-based) architectures - Emotion Classification Head: Add classification layers to predict emotion labels - Sequence-to-Sequence Prediction: Predict emotions for each turn in

the dialogue - Drift Detection Wrapper: Analyze emotion transitions to detect significant shifts

Stage 5: Evaluation - Emotion Classification Metrics: Calculate F1-score (per emotion class and macro-averaged), accuracy, and confusion matrix - Drift Detection Metrics: Measure average sentiment shift, emotion transition accuracy, and drift detection precision/recall - Visualization: Create emotion trajectory graphs and transition heatmaps

Stage 6: Visualization & Analysis - Plot emotion trajectories across conversations to visualize emotional progression - Highlight drift points where emotions shift significantly (e.g., positive to negative) - Analyze correlation between AI responses and subsequent emotion changes - Generate insights for improving AI response strategies based on detected patterns

Pipeline Stages

Data Collection:

- Input: Dataset identifiers (DailyDialog, EmotionLines, MELD)
- Output: Raw DataFrame with dialogue*id, turn*id, speaker, text, emotion
- Tools: Hugging Face datasets library, pandas

Preprocessing:

- Input: Raw dialogue DataFrames
- Output: Cleaned, tokenized sequences with encoded emotions
- Tools: Transformers library (BERT/RoBERTa tokenizers), scikit-learn LabelEncoder

Feature Extraction:

- Input: Tokenized text sequences
- Output: Contextual embeddings (768-dim for BERT-base)
- Tools: Pre-trained BERT/RoBERTa models from Hugging Face

Model Training:

- Input: Embedded sequences and emotion labels
- Output: Trained model capable of emotion classification and drift detection
- Tools: PyTorch, Transformers library

Evaluation:

- Input: Model predictions and ground truth labels
- Output: Performance metrics and visualizations
- Tools: scikit-learn metrics, matplotlib, seaborn

Input/Output Structures

Input Structure: - Text sequences: `[batch_size, max_seq_len]` (token IDs) - Attention masks: `[batch_size, max_seq_len]` (1 for real tokens, 0 for padding) - Dialogue metadata: List of dialogue_ids for sequence grouping

Output Structure: - Emotion logits: `[batch_size, seq_len, num_emotions]` - Predicted emotions: `[batch_size, seq_len]` (class indices) - Drift scores: `[batch_size, seq_len-1]` (magnitude of emotion changes)

6. Approach

ML Approach: Supervised Learning

This project employs **supervised learning** to solve the emotion drift detection problem. The task is framed as:

- **Input**: Sequences of dialogue utterances with speaker information
- **Output**: Emotion labels for each utterance, enabling drift detection through sequence analysis

ML Algorithms/Models

The project implements and compares two primary model architectures:

Bidirectional LSTM (BiLSTM):

- Architecture: Multi-layer bidirectional LSTM with fully connected classification head
- Input: Pre-computed BERT embeddings (768 dimensions)
- Hidden dimensions: 256 (512 total with bidirectional)
- Layers: 2 stacked LSTM layers
- Rationale: Captures temporal dependencies in dialogue sequences while being computationally efficient

Transformer-based Model (BERT/RoBERTa):

- Architecture: Fine-tuned BERT or RoBERTa with classification head
- Base model: `bert-base-uncased` or `roberta-base`
- Classification: Token-level emotion prediction using transformer's sequence output
- Rationale: Leverages pre-trained contextual embeddings that capture nuanced emotional cues

Emotion Drift Detector:

- Wrapper model that compares consecutive emotion predictions
- Calculates drift scores based on emotion transition magnitude
- Identifies significant emotion shifts (configurable threshold)

Justification

Why Supervised Learning? - Emotion labels are available in the datasets, enabling supervised training - Direct mapping from text to emotions allows for interpretable results - Ground truth labels enable quantitative evaluation of drift detection accuracy

Why Sequence Models? - Emotions in dialogue are context-dependent; previous turns influence current emotion - Sequential modeling captures how emotions evolve across conversation timeline - Enables detection of emotional progression and regression patterns

Why Transformer Embeddings? - Pre-trained BERT/RoBERTa models capture rich semantic and emotional information - Contextual embeddings understand emotional nuances better than static word embeddings - Transfer learning reduces data requirements and improves generalization

Why Compare LSTM vs Transformer? - LSTM provides baseline for sequential modeling - Transformer leverages pre-training but requires more computational resources - Comparison identifies best trade-off between accuracy and efficiency

7. Technical Design

ML Libraries and Frameworks

PyTorch (>=2.0.0):

- Primary deep learning framework
- Used for model definition, training loops, and tensor operations
- Provides flexibility for custom architectures

Transformers (Hugging Face, >=4.30.0):

- Pre-trained BERT and RoBERTa models
- Tokenizers for text preprocessing
- Model loading and fine-tuning utilities

Datasets (Hugging Face, >=2.14.0):

- Dataset loading and caching
- Efficient data streaming for large datasets

scikit-learn (>=1.3.0):

- Label encoding for emotion classes
- Class weight computation for imbalanced data
- Evaluation metrics (F1-score, accuracy, etc.)

Pandas (>=2.0.0) & **NumPy** (>=1.24.0):

- Data manipulation and preprocessing
- Numerical computations

Matplotlib (>=3.7.0) & **Seaborn** (>=0.12.0):

- Visualization of emotion trajectories
- Plotting confusion matrices and transition heatmaps

Development Tools

Jupyter Notebooks:

- Interactive data exploration (`notebooks/data_exploration.ipynb`)
- Prototyping and visualization
- Reproducible analysis workflows

Version Control (Git):

- Track code changes and experiment iterations
- Collaborate and maintain project history
- `.gitignore` configured for data and model files

Experiment Tracking (Optional - Future):

- MLflow for experiment logging
- Track hyperparameters, metrics, and model versions
- Compare different model configurations

Project Structure

Project Structure: - `src/data/` - Raw and processed datasets - `src/models/` - Trained model checkpoints - `src/notebooks/` - Jupyter notebooks - `data_exploration.ipynb` - Data exploration notebook - `src/src/` - Source code modules - `__init__.py` - Package initialization - `data_loader.py` - Dataset loading utilities - `preprocessing.py` - Preprocessing pipeline - `models.py` - Model architectures - `requirements.txt` - Python dependencies - `README.md` - Project documentation - `test_setup.py` - Setup verification script

8. Experimental Setup

Research Question

Primary Research Question: "Can an emotion drift detection model identify when user emotion shifts during a customer support AI conversation, and does the AI's response correlate with that shift?"

Secondary Questions: - Which model architecture (LSTM vs Transformer) better captures emotion drift patterns? - What are the most common emotion transition patterns in customer support dialogues? - Can we predict emotion drift before it occurs to enable proactive response adjustment?

Experimental Comparisons

The project will compare:

Model Architectures:

- **BiLSTM**: Baseline sequential model with pre-computed embeddings
- **Transformer (BERT/RoBERTa)**: Pre-trained transformer with fine-tuning

- Comparison metrics: Emotion classification accuracy, drift detection precision/recall, computational efficiency

Embedding Strategies (Future):

- Static word embeddings (Word2Vec, GloVe)
- Contextual embeddings (BERT, RoBERTa)
- Compare impact on emotion classification performance

Sequence Modeling Approaches:

- Unidirectional vs Bidirectional LSTM
- Different sequence lengths and context windows
- Impact on long-term emotion dependency capture

Evaluation Metrics

1. Emotion Classification Metrics:

F1-Score (Primary):

- Macro-averaged F1 across all emotion classes
- Per-class F1 to identify class-specific performance
- Handles imbalanced emotion distribution

Accuracy:

- Overall classification accuracy
- Percentage of correctly predicted emotions

Confusion Matrix:

- Visual representation of classification errors
- Identifies common misclassification patterns (e.g., anger vs sadness)

2. Emotion Drift Detection Metrics:

Average Sentiment Shift:

- Mean magnitude of sentiment change between consecutive turns
- Formula: `mean(|sentiment[t+1] - sentiment[t]|)`
- Higher values indicate more volatile emotional trajectories

Drift Detection Accuracy:

- Precision: Proportion of detected drifts that are actual emotion changes
- Recall: Proportion of actual drifts that are successfully detected
- F1-score for drift detection binary classification

Emotion Transition Matrix:

- Frequency matrix of emotion-to-emotion transitions
- Identifies common drift patterns (e.g., neutral → anger, joy → neutral)

3. Trajectory Analysis Metrics:

Emotion Stability Score:

- Variance of emotions within a dialogue
- Lower variance indicates more stable emotional states

Drift Correlation:

- Correlation between AI response features and subsequent emotion changes
- Measures how well AI responses predict emotion shifts

Quantitative Criteria

- **Classification Performance**: Target F1-score > 0.70 for emotion classification
- **Drift Detection**: Precision > 0.75 for significant emotion shifts (>1 emotion class change)
- **Computational Efficiency**: Training time per epoch < 30 minutes on standard GPU
- **Generalization**: Validation accuracy within 5% of training accuracy (no overfitting)

Experimental Procedure

Data Split:

- Training: 70% of dialogues
- Validation: 15% of dialogues
- Test: 15% of dialogues
- Stratified split to maintain emotion distribution

Training Configuration:

- Batch size: 16-32 (depending on model and GPU memory)
- Learning rate: 2e-5 for transformers, 1e-3 for LSTM
- Optimizer: AdamW with weight decay
- Early stopping: Stop if validation F1 doesn't improve for 5 epochs

Baseline Comparisons:

- Majority class baseline (always predict most common emotion)

- Random baseline (random emotion prediction)
- Simple logistic regression on bag-of-words features

9. References

Zhang, J., et al. (2024). The Impact of Emotional Expression by Artificial Intelligence. *Decision Support Systems*, 181, 114075.

Pezenka, I., et al. (2024). Emotionality in Task-Oriented Chatbots – The Effect of Emotion Expression on Customer Satisfaction. *Journal of Applied Communication Research*.

Han, E., et al. (2022). Should AI Agents Express Positive Emotion in Customer Service? *SSRN*.

Yun, J., et al. (2022). The Effects of Chatbot Service Recovery With Emotion on Customer Satisfaction and Trust. *Frontiers in Psychology*.

Gupta, R., Ranjan, S., & Singh, A. (2024). Comprehensive Study on Sentiment Analysis: From Rule-Based to Modern LLM-Based Systems. *arXiv preprint arXiv:2409.09989*.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*, 4171-4186.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Hsu, C. C., Chen, S. Y., Kuo, C. C., Huang, T. H., & Ku, L. W. (2018). EmotionLines: An Emotion Corpus of Multi-Party Conversations. *Proceedings of LREC*, 1597-1601.

Li, Y., Su, H., Shen, X., Li, W., Cao, Z., & Niu, S. (2017). DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. *Proceedings of IJCNLP*, 986-995.

Poria, S., Hazarika, D., Majumder, N., Naik, G., Cambria, E., & Mihalcea, R. (2019). MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations. *Proceedings of ACL*, 527-536.