

# Group 24 Final Report: Reddit Advice Intent Classification

Ahmad Allabadi, Rayan Nasrallah, Karamvir Hira, Aizaz Lodhi  
{allabada,nasrallr,hirak,lodhia8}@mcmaster.ca

## 1 Introduction

Advice-seeking is one of the most common uses of online discussion platforms, yet most NLP research on user-generated content focuses on sentiment analysis or toxicity detection rather than the functional role a comment plays in a conversation. In this project, we classify Reddit comments from advice-oriented subreddits into one of five categories: advice, warning, emotional support, anecdote, and appraisal. Each category corresponds to a distinct communicative function, and our label taxonomy is grounded in speech act theory (Searle, 1969), which describes how utterances perform functions beyond simply conveying information.

The most closely related area of NLP research is dialogue act classification, which labels utterances by their communicative function in conversation (Lee and Deroncourt, 2016). Our setting differs in that we classify isolated Reddit comments rather than sequential dialogue turns, with a label set tailored to advice-oriented communities. Recent work has explored related tasks on Reddit, such as extracting alternative suggestions from r/Advice replies (Kostiuk et al., 2025), but to our knowledge no prior work has focused on classifying the type of advice given across multiple advice subreddits.

This task is challenging because advice types are often subtle and overlapping. A single comment may contain both a personal story and a practical suggestion, requiring models to identify the dominant communicative intent rather than relying on surface-level cues. Our dataset consists of approximately 2,000 Reddit comments collected from r/Advice, r/legaladvice, r/personalfinance, and r/relationship\_advice, manually annotated by our team following a structured annotation protocol with a defined label priority system.

## 2 Related Work

Our task is most closely related to dialogue act classification, which assigns communicative function labels to utterances in conversation. Stolcke et al. (2000) introduced a foundational approach to this problem on the Switchboard corpus, classifying conversational speech into acts such as statements, questions, and agreements. Lee and Deroncourt (2016) later applied recurrent and convolutional neural networks to short-text dialogue act classification, showing that deep learning methods could effectively capture patterns in communicative function labeling.

Our work differs from traditional dialogue act classification in that we classify isolated Reddit comments rather than sequential conversation turns, and our label set targets the communicative functions specific to advice-oriented communities. The theoretical basis for our labels draws on speech act theory (Searle, 1969), which frames language as action, where utterances serve functions such as directing, warning, or expressing emotion. Our five categories (advice, warning, emotional support, anecdote, and appraisal) can each be understood as distinct speech act types.

The most closely related study is Kostiuk et al. (2025), who used Reddit r/Advice posts to extract alternative decision-making suggestions from user replies. While their focus was on generating alternatives rather than classifying communicative intent, their work confirms the value of advice-oriented Reddit data for NLP research. For our modeling approach, we leverage pre-trained transformers such as BERT (Devlin et al., 2019), which have become standard for text classification in low-resource settings where fine-tuning on a small labeled dataset can substantially outperform models trained from scratch.

### 3 Dataset

#### 3.1 Source and Collection

Our dataset consists of 2,000 Reddit comments collected from four advice-oriented subreddits: r/Advice, r/legaladvice, r/personalfinance, and r/relationship\_advice. These communities were selected because their primary purpose is to provide help or guidance, making them well-suited for studying different types of advice and support behavior. Comments were obtained using the Arctic Shift public download tool, which provides access to historical Reddit data archives. Only publicly available comments were collected, and no private or deleted content was accessed.

Data was filtered in multiple stages. Comments marked as [deleted] or [removed], moderator messages, bot responses, and off-topic replies were removed. The remaining comments were randomly shuffled using a fixed seed for reproducibility, and a final subset of 2,000 instances was selected. To avoid subreddit imbalance, the dataset was stratified so that each subreddit contributed exactly 500 comments.

The label distribution across the train, validation, and test splits is shown in Table 1. The dataset is notably imbalanced, with advice being the most frequent class at roughly 42% and emotional support being the least common at under 5%. The distributions are consistent across all three splits, indicating that stratified splitting was applied correctly.

| Label             | Train | Val | Test |
|-------------------|-------|-----|------|
| Advice            | 585   | 125 | 126  |
| Warning           | 88    | 19  | 19   |
| Emotional Support | 65    | 14  | 14   |
| Anecdote          | 316   | 68  | 67   |
| Appraisal         | 346   | 74  | 74   |
| <b>Total</b>      | 1,400 | 300 | 300  |

Table 1: Label distribution across train, validation, and test splits.

#### 3.2 Annotation

Each comment was labeled by a member of our team into one of five mutually exclusive categories: advice, warning, emotional support, anecdote, and appraisal. Annotators were instructed to assign a single label based on the dominant communicative function of the comment, following a structured priority order: advice, then warning, then emotional

support, then anecdote, then appraisal. This priority system was used to resolve cases where a comment contained elements of multiple categories.

A subset of 300 instances was double-annotated to measure inter-annotator agreement. We obtained a Cohen’s Kappa score of 0.534 with 65% percent agreement, indicating moderate agreement. For instances where the two annotators disagreed, a third annotator reviewed the comment and served as a tie-breaker to determine the final label. This reflects the inherent ambiguity in classifying communicative intent, particularly for comments that blend multiple functions.

#### 3.3 Annotation Highlights

Several patterns emerged during annotation that are worth noting. Extremely short comments such as "Divorce him" could still carry clear intent, showing that classification is not length-dependent. Conversely, long multi-paragraph comments often blended explanation, perspective, and guidance, making the dominant function harder to isolate. Some comments produced disagreement between annotators, for example "Get out before it’s too late man" was split between advice and warning, since it contains both an instruction and an emphasis on risk. These observations informed our expectation that models would struggle most with comments where multiple communicative functions overlap.

#### 3.4 Preprocessing

The original data was collected in JSON format containing extensive metadata including usernames, timestamps, and vote counts. For annotation and modeling, all metadata was removed and each instance was reduced to a unique identifier and the raw comment text. No additional text cleaning such as lowercasing or stopword removal was applied, as our approach relies on pre-trained transformer tokenizers that handle raw text directly.

#### 3.5 Dataset Changes Since Progress Report

No changes were made to the dataset since the progress report. The same 2,000 annotated instances, label definitions, and annotation protocol were used for the final experiments.

### 4 Features

Our experiments compare several levels of text representation, ranging from sparse lexical features to static word-embedding representations to dense contextual embeddings and hybrid combinations.

This progression lets us isolate how much of the classification difficulty stems from representation quality versus inherent label ambiguity. Below, we describe the features used by each model; Sections 5 and 6 cover implementation details and comparative results, respectively.

#### 4.1 Baselines

The majority vote and random baselines use no text features. The majority baseline predicts the most frequent training label (advice) for every input, while the random baseline samples from the training label distribution. The TF-IDF + Logistic Regression baseline converts each comment into a sparse bag-of-words vector using unigram and bigram TF-IDF weighting, with a minimum document frequency of 2 and a maximum vocabulary size of 50,000. This representation captures word frequency patterns but does not encode word order or contextual meaning, making it a useful point of comparison against more expressive models.

#### 4.2 RoBERTa (Ahmad)

The RoBERTa model (Liu et al., 2019) takes raw comment text as input with no manual feature engineering. Each comment is tokenized using RoBERTa’s Byte-Pair Encoding (BPE) tokenizer, which splits text into subword tokens and converts them to integer IDs from a 50,265-token vocabulary. Inputs are truncated or padded to a maximum sequence length of 512 tokens. The model produces 768-dimensional contextual embeddings per token, meaning the same word receives different representations depending on its surrounding context. Classification is performed using the special start token representation from the final hidden layer, passed through a dropout layer and a linear classification head. The entire pipeline operates end-to-end as a single neural network, with no features extracted separately.

Contextual embeddings are well-suited for this task because the same words can signal different communicative intents depending on context. For example, "you should leave" could function as advice or a warning depending on the surrounding language. Static representations like TF-IDF cannot capture this distinction, which is why we expect the transformer-based approach to outperform the bag-of-words baseline. Comparing these two input representations, sparse TF-IDF versus dense contextual embeddings, also serves as a way of varying the features used across our experiments.

For two of the four RoBERTa runs, we applied targeted Easy Data Augmentation (Wei and Zou, 2019) to minority classes only, upsampling emotional support, warning, anecdote, and appraisal to match the majority class count of 585. Augmentation techniques included synonym replacement, random insertion, random swap, and random deletion, applied to raw text before tokenization. The majority class (advice) was left untouched. This allowed us to compare model performance with and without augmentation as part of our ablation study.

#### 4.3 Stacked Ensemble Model (Karamvir)

Where RoBERTa relies on a single end-to-end representation, the stacked ensemble takes the opposite approach: it extracts multiple complementary feature views from each comment and trains a separate classifier on each before combining their predictions.

The input text is first preprocessed by removing URLs, Reddit-specific patterns, and extra whitespace, and converting to lowercase. Three feature representations are then constructed. The first is a dense sentence embedding generated using the pretrained `intfloat/e5-large-v2` model (Wang et al., 2022), building on modern sentence embedding methods (Cao, 2024). These vectors capture the overall semantic meaning of the text without requiring task-specific fine-tuning. The second representation is word-level TF-IDF with  $n$ -grams, capturing important words and short phrases. The third is character-level TF-IDF, which captures subword patterns such as prefixes, suffixes, and writing-style markers which are useful for handling the informal and variable spelling found in Reddit comments.

These three representations capture fundamentally different signals: embeddings encode meaning and context, word-level TF-IDF captures exact lexical cues, and character-level TF-IDF captures morphological and stylistic patterns. By combining them, the ensemble can leverage both semantic and surface-level information.

To address class imbalance in the embedding space, mixup augmentation generates synthetic minority-class samples for minority classes by interpolating between existing embeddings (Duffy, 2025). This differs from the text-level EDA used in the RoBERTa experiments, and the comparison between these two augmentation strategies forms part of our analysis in Section 6.

#### 4.4 TextCNN (Aizaz)

The input to the TextCNN model is the raw Reddit comment text. Comments are lowercased, tokenized at the word level, and converted into sequences of token IDs using a vocabulary built from the training split. Rare or unseen tokens are mapped to an unknown token, and each sequence is padded or truncated to a fixed maximum length.

Unlike RoBERTa, this model does not use contextualized transformer embeddings. Instead, it uses a 200-dimensional embedding layer initialized with Word2Vec vectors trained on the training comments (Mikolov et al., 2013). These embeddings are then fine-tuned during supervised training. Unlike the stacked ensemble, the TextCNN uses a single end-to-end neural representation rather than combining multiple feature views and a meta-learner. This makes it more expressive than TF-IDF + Logistic Regression while still being simpler than the transformer and ensemble approaches.

The goal of this representation is to let the model capture local phrase patterns such as directives, warnings, or supportive expressions. To study the effect of the input representation, we compared random embedding initialization against Word2Vec initialization. We also compared weighted and unweighted loss to see how class imbalance handling affected the results.

#### 4.5 BiLSTM with Flair Embeddings (Rayan)

The BiLSTM model takes raw Reddit comment text as input, without any manual feature engineering. Each comment is first converted into pretrained Flair contextual embeddings (Akbik et al., 2018) using a forward and backward character-level language model (news-forward-fast and news-backward-fast), each producing 1024-dimensional token representations. For each token, these two representations are concatenated to form a 2048-dimensional embedding. Because Flair builds these embeddings from character-level context, they capture not only word meaning but also subword and contextual information, which makes them especially useful for Reddit text where spelling, wording, and style are often informal or inconsistent.

Flair embeddings are contextualized, so the same word can receive different representations depending on the sentence it appears in. This matters for our task because many Reddit comments contain words like “leave,” “fine,” or “serious” whose

meaning shifts depending on whether the comment is giving advice, expressing concern, or offering support. Unlike transformer-based models, Flair captures this context through recurrent character-level language models rather than self-attention. In our experiments, the Flair encoder was kept fully frozen and only the downstream classifier was trained. We made this choice because our dataset is relatively small, with only 1,400 training examples, and fully fine-tuning the pretrained encoder would likely overfit and weaken the general language knowledge it had already learned. Freezing the encoder let us use those pretrained contextual representations as stable features while training a lighter classifier to adapt them to our five intent labels.

To address the strong class imbalance in the dataset, we also experimented with Easy Data Augmentation (EDA) (Wei and Zou, 2019) on the minority classes. Following the same setup used in the RoBERTa experiments, we applied four simple text-level augmentation operations—synonym replacement, random insertion, random swap, and random deletion—before generating the Flair embeddings. The majority class, advice, was not augmented, since it was already much more common than the others. The goal was to give the model more varied examples of warning, emotional support, anecdote, and appraisal, and to test whether increasing minority-class diversity would improve performance.

### 5 Implementation

This section describes the training procedure and hyperparameters for each model. All models were trained on the same 1,400-instance training set and evaluated on the same held-out test set of 300 instances; details of the data split are given in Section 3.

#### 5.1 Baselines

We implemented three baselines: a majority vote baseline that always predicts the most frequent training label (advice), a random baseline that samples from the training label distribution, and a TF-IDF + Logistic Regression model using unigram and bigram features. On the test set, the majority baseline achieved 42.0% accuracy and 0.118 macro F1, the random baseline achieved 24.7% accuracy and 0.170 macro F1, and TF-IDF + Logistic Regression achieved 58.3% accuracy and 0.342 macro

F1.

## 5.2 RoBERTa (Ahmad)

We fine-tuned the roberta-base checkpoint (Liu et al., 2019), a 125M parameter pre-trained transformer with 12 layers and a hidden size of 768. A classification head consisting of a dense layer (768 to 768) with Tanh activation, a dropout layer (rate 0.3), and a final linear layer (768 to 5) was added on top of the encoder. The model was trained using the HuggingFace Trainer API with AdamW optimization at a learning rate of  $2e-5$  and weight decay of 0.01. A linear learning rate scheduler with warmup over the first 10% of training steps was used. Training ran for up to 4 epochs with a batch size of 8, and early stopping with a patience of 2 was applied based on validation macro F1, loading the best model at the end. Half-precision (fp16) training was used on an NVIDIA Tesla T4 GPU.

To address class imbalance, we computed inverse frequency class weights, giving higher weight to underrepresented classes such as emotional support (weight 4.31) and warning (weight 3.18), while downweighting advice (weight 0.48). We ran four configurations to compare the effects of loss function and data augmentation:

|                    | No Augmentation | With EDA |
|--------------------|-----------------|----------|
| <b>Weighted CE</b> | Run 1           | Run 4    |
| <b>Focal Loss</b>  | Run 2           | Run 3    |

Table 2: RoBERTa run configurations.

Weighted cross-entropy applies the class weights directly to the standard cross-entropy loss. Focal loss adds a modulating factor of  $(1 - p_t)^\gamma$  with  $\gamma = 2.0$ , which downweights easy examples and focuses training on harder cases. For the augmented runs, we applied EDA (Wei and Zou, 2019) to minority classes only, upsampling emotional support, warning, anecdote, and appraisal to match the majority class count of 585, increasing the training set from 1,400 to approximately 2,925 instances.

The best performing configuration was Weighted CE without augmentation, achieving 64.7% accuracy and a macro F1 of 0.503 on the test set. This outperforms all baselines, improving over TF-IDF + Logistic Regression by 6.4 percentage points in accuracy and 47% in relative macro F1. EDA improved accuracy (66.3% for Weighted CE + EDA) but reduced macro F1 to 0.478, suggesting that augmentation benefited majority-class predictions

at the expense of minority-class performance. Focal loss did not outperform weighted cross-entropy in any configuration, indicating that the simpler loss function was sufficient for this dataset. A full comparison across all configurations is given in Section 6.

## 5.3 Stacked Ensemble Model (Karamvir)

The stacked ensemble (Wolpert, 1992) trains a separate base classifier on each of the three feature representations described in Section 4.3, then combines their predictions through a meta-learner.

The embedding features are passed to a logistic regression classifier, which learns linear decision boundaries over the dense vectors. The word-level and character-level TF-IDF features are each passed to linear support vector classifiers, which are effective for high-dimensional sparse data. Since SVMs do not produce probabilistic outputs, Platt scaling is applied to convert their decision scores into calibrated probabilities.

Each base model produces a five-dimensional probability vector (one value per class), and these 15 values are concatenated to form the input to the meta-learner, also a logistic regression model. This second-level classifier learns how to weight and reconcile the predictions from each base model, allowing the system to exploit whichever representation is most informative for a given input.

Class weighting is applied in all base classifiers and the meta-learner to counteract the label imbalance. In addition, mixup augmentation in the embedding space generates synthetic minority-class samples during training. After training, per-class decision thresholds are tuned on the validation set rather than using the default 0.5 cutoff, providing an additional mechanism to boost minority-class recall.

To evaluate the contribution of each component, we also report results for each base model individually (embedding-only, word TF-IDF-only, and character TF-IDF-only), which serves as an ablation showing how each feature type contributes to the final ensemble.

The full ensemble achieves 67.0% accuracy and a macro F1 of 0.506 on the test set, outperforming all baselines and performing comparably to the best RoBERTa configuration. The ensemble’s strength lies in combining semantic and lexical signals, though as discussed in Section 7, this comes with trade-offs in per-class performance relative to RoBERTa.

## 5.4 TextCNN (Aizaz)

The TextCNN model was implemented as a sentence classifier following [Kim \(2014\)](#). Each comment is represented as a sequence of 200-dimensional word embeddings, then passed through three parallel one-dimensional convolution layers with kernel sizes 3, 4, and 5. Each convolution layer uses 100 filters. After ReLU activation, global max pooling is applied to each filter bank, and the pooled outputs are concatenated, passed through dropout with rate 0.5, and fed into a final linear layer that predicts one of the five intent classes.

The vocabulary was built from the training split only, using a minimum token frequency of 2. Sequence length was set to the 95th percentile of training comment length, capped at 256, which gave a maximum length of 179 tokens. The final vocabulary size was approximately 3,800 tokens.

Training used Adam with a learning rate of  $1e-3$  and weight decay of  $1e-4$ . The batch size was 32 for training and 64 for evaluation. We trained for up to 12 epochs with early stopping patience of 3 based on validation macro F1, and the best checkpoint came from epoch 10.

We ran three versions of the model: random embeddings with weighted loss, Word2Vec initialization with weighted loss, and Word2Vec initialization with unweighted loss. The best validation macro F1 came from the Word2Vec + weighted-loss setting, so this was selected as the final configuration. Although the unweighted run had slightly higher accuracy, its macro F1 was lower, so weighted loss was preferred because the dataset is imbalanced.

## 5.5 BiLSTM with Flair Embeddings (Rayan)

To make training practical, Flair embeddings were precomputed once for each data split and cached in memory, so the character-level language model only had to run once per experiment rather than at every epoch. As explained in section 4.5, each token was represented by a 2048-dimensional embedding formed by concatenating the 1024-dimensional forward and backward Flair language model outputs. Within each batch, sequences were padded using PyTorch’s `pad_sequence`, and packed sequences were used during the BiLSTM forward pass so computation was not wasted on padding tokens.

On top of these frozen Flair embeddings, we

trained a bidirectional LSTM classifier to model the sequence of words in each comment. The idea is that Flair already provides strong contextual word representations, while the BiLSTM learns how those word-level signals combine across the full comment to indicate a single communicative intent. To turn the BiLSTM outputs into one fixed representation for classification, we used concat-pooling: we combined the final hidden state from the last LSTM layer with both a max-pooled and mean-pooled summary over all hidden states. This gave the model access to three complementary views of the comment: what the LSTM retained at the end, the strongest signals that appeared anywhere in the sequence, and the overall average pattern across the comment. The pooled BiLSTM representation was then passed to a small feedforward classifier (MLP) to produce the final label prediction,  $\text{Dropout}(0.4) \rightarrow \text{Linear}(3072, 256) \rightarrow \text{ReLU} \rightarrow \text{Dropout}(0.4) \rightarrow \text{Linear}(256, 5)$ , to produce the final prediction. We also used inverse-frequency class weights in the cross-entropy loss so that errors on minority classes had a larger effect during training.

Training used the Adam optimizer ([Kingma and Ba, 2015](#)) with a learning rate of  $5e-4$ . We also applied gradient clipping at 1.0, `ReduceLROnPlateau` based on validation macro F1, and early stopping with patience 4. These choices were meant to keep training stable and reduce overfitting, which was a major concern given the small size of the dataset. After training, we restored the checkpoint with the best validation macro F1 before running the final evaluation.

We tested three versions of the model. Run 1 used a 2-layer BiLSTM with hidden size 512 and no augmentation, giving a baseline for how well the frozen Flair embeddings worked on their own. Run 2 kept the same 2-layer BiLSTM but added EDA to the minority classes, so we could measure the effect of augmentation without changing the model architecture. Run 3 again used EDA but increased the classifier to 3 BiLSTM layers, allowing us to test whether a deeper recurrent model could make better use of the augmented training data.

# 6 Results and Evaluation

## 6.1 Evaluation Strategy

The dataset was split into train (1,400), validation (300), and test (300) sets using a two-stage stratified split with a random seed of 42, preserving the class distribution across all three sets. A 70/15/15

ratio was chosen to maximize training data for fine-tuning while retaining enough validation instances for early stopping and hyperparameter selection, and enough test instances for reliable evaluation across all five classes. We report accuracy, macro F1, and weighted F1 on the held-out test set. Macro F1 is our primary metric as it weights all classes equally regardless of their frequency, which is important given the significant class imbalance in our dataset. Weighted F1 accounts for class support and provides a complementary view of performance.

## 6.2 Baseline and Model Results

Table 3 summarizes the test set performance of all baselines and model configurations.

| Model                   | Acc  | F1-M | F1-W |
|-------------------------|------|------|------|
| Majority                | 42.0 | .118 | .249 |
| Random                  | 24.7 | .170 | .247 |
| TF-IDF + LR             | 58.3 | .342 | .531 |
| RoBERTa WCE             | 64.7 | .503 | .640 |
| RoBERTa FL              | 62.0 | .483 | .623 |
| RoBERTa FL+EDA          | 63.3 | .493 | .619 |
| RoBERTa WCE+EDA         | 66.3 | .478 | .642 |
| Stacked Ensemble        | 67.0 | .506 | .650 |
| TextCNN (W2V + weights) | 57.3 | .426 | .555 |
| BiLSTM Flair Base       | 60.0 | .395 | .574 |
| BiLSTM Flair EDA        | 42.0 | .339 | .415 |
| BiLSTM Flair Deep+EDA   | 54.7 | .427 | .549 |

Table 3: Test set results. Acc = Accuracy (%), F1-M = Macro F1, F1-W = Weighted F1.

All trained models improve substantially over the trivial baselines. The TF-IDF baseline achieves competitive accuracy because of the dominant advice class, but it completely fails to predict emotional support and warning (0.00 F1 for both), confirming that richer representations are necessary for recognizing minority communicative intents.

**RoBERTa ablation:** Among the four RoBERTa configurations, weighted cross-entropy without augmentation achieves the best macro F1 (0.503), improving over TF-IDF + Logistic Regression by 47% in relative terms. EDA augmentation improved accuracy (66.3% for Weighted CE + EDA) but reduced macro F1 to 0.478, indicating that the synthetic training examples benefited majority-class predictions at the expense of minority-class precision. Focal loss did not outperform weighted cross-entropy in any configuration, suggesting the simpler loss function was sufficient for this dataset.

**Stacked ensemble:** The stacked ensemble achieves the highest accuracy (67.0%) and a macro F1 of 0.506. However, the difference between the ensemble and the best RoBERTa configuration is

small sitting at 0.3 percentage points in macro F1 and 2.3 points in accuracy and is unlikely to be statistically significant given a test set of only 300 instances. The more informative comparison is at the per-class level: as shown in the error analysis (Section 7), RoBERTa performs notably better on warning ( $F1 = 0.40$  vs.  $\approx 0.17$ ), while the ensemble has an edge on emotional support ( $F1 \approx 0.33$  vs. 0.10). The two models make complementary errors rather than one strictly dominating the other.

**TextCNN.** The selected configuration, a TextCNN with Word2Vec initialization and class-weighted cross-entropy, achieved 57.3% accuracy, 0.426 macro F1, and 0.555 weighted F1 on the test set. This model improved over TF-IDF + Logistic Regression in macro F1 (0.426 vs. 0.342), although its accuracy was slightly lower (57.3% vs. 58.3%). This suggests that the CNN was better at handling class imbalance than the linear baseline, even though it did not improve overall accuracy. However, the TextCNN still performed below RoBERTa and the stacked ensemble, which indicates that the simpler CNN architecture was less effective on this task than the stronger contextual and hybrid models.

**BiLSTM Flair.** Using macro F1 as our primary metric, the best BiLSTM configuration was Run 3, which used a 3-layer BiLSTM together with EDA and achieved a macro F1 of 0.427. This outperformed the TF-IDF + Logistic Regression baseline on macro F1 (0.427 vs. 0.342). Run 1 achieved the highest accuracy (60.0%) and weighted F1 (0.574), but its macro F1 was lower at 0.395. Since macro F1 is the main metric in this report because of the strong class imbalance, we treat Run 3 as the strongest overall BiLSTM configuration.

Run 2 produced the weakest result, with a macro F1 of 0.339, slightly below the TF-IDF baseline. This suggests that adding EDA alone did not help when the classifier remained relatively shallow. When the model depth was increased in Run 3, performance improved beyond both Run 1 and Run 2, indicating that the augmented data was only useful once the recurrent classifier had enough capacity to make use of it.

**Cross-model observations.** Across all approaches, the primary bottleneck is minority-class performance. No model achieves strong F1 on both emotional support and warning at the same time. RoBERTa is the strongest model on warning, while the stacked ensemble, TextCNN, and the best BiLSTM Flair configuration show relatively better

performance on emotional support. Augmentation and class-imbalance strategies produce mixed results across models: in some cases they improve recall on minority classes, but often at the cost of precision or overall macro F1. This suggests that additional labeled data for emotional support and warning would likely be the most impactful improvement.

### 6.3 Comparison to Others

| Participant    | Approach             | Macro F1 |
|----------------|----------------------|----------|
| hirak19        | Stacked Ensemble     | 0.506    |
| allabada       | RoBERTa              | 0.500    |
| nasrallr       | BiLSTM Flair         | 0.427    |
| lodhia8        | TextCNN              | 0.426    |
| gus46          | Classmate submission | 0.484    |
| kansas         | Classmate submission | 0.403    |
| sharmg36       | BERT-base-uncased    | 0.403    |
| nickp McMaster | Classmate submission | 0.400    |

Table 4: CodaBench development leaderboard scores (Macro F1) for selected submissions.

Table 4 shows selected Macro F1 scores from the CodaBench development leaderboard. The first four entries in the table are submissions from our own group: the stacked ensemble, RoBERTa, BiLSTM Flair, and TextCNN. For most external submissions, we only had access to the leaderboard usernames and scores, since CodaBench does not display the model approach used in each submission. As a result, those entries are labeled generically as “Classmate submission.” The only exception is sharmg36, whose approach was identified as BERT-base-uncased based on information shared with us by email.

Among our group’s submissions, the stacked ensemble achieved the strongest result at 0.506, closely followed by RoBERTa at 0.500. Our BiLSTM with Flair embeddings achieved 0.427, placing it below the strongest contextual and ensemble-based models but still above several other submissions on the leaderboard.

This ranking is broadly consistent with the results in Section 6.2. The stacked ensemble and RoBERTa both combine stronger representations with more flexible decision boundaries, which likely explains why they outperform the BiLSTM Flair model. In contrast, the BiLSTM Flair approach still improves over simpler baselines, but its performance is limited by weaker minority-class generalization, especially on warning and emotional support. Overall, these comparisons suggest

that pretrained contextual recurrent features are useful, but they remain less effective on this task than transformer-based or hybrid ensemble approaches.

## 7 Error Analysis

All models perform well on the three majority classes, which are advice, anecdote, and appraisal, but struggle with the two minority classes, emotional support and warning. Below, we analyze each model’s error patterns individually, then identify shared failure modes that cut across all approaches.

### 7.1 RoBERTa Error Patterns (Ahmad)

Figure 1 shows the confusion matrix for RoBERTa + Weighted CE. The model correctly identifies 89 of 126 advice comments (F1 = 0.742), 48 of 67 anecdotes (F1 = 0.658), and 49 of 74 appraisals (F1 = 0.616). On minority classes, it correctly classifies 7 of 19 warnings (F1 = 0.400) but only 1 of 14 emotional support comments (F1 = 0.100).

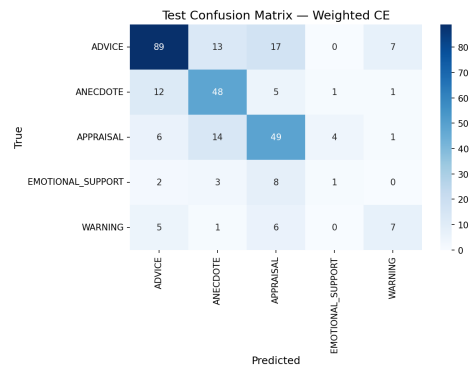


Figure 1: Confusion matrix for RoBERTa + Weighted CE on the test set.

The most common confusion pairs are informative. The model most frequently misclassifies advice as appraisal (17 cases) and appraisal as anecdote (14 cases), suggesting difficulty distinguishing comments that blend opinion, storytelling, and guidance. Advice–anecdote confusion is bidirectional (13 and 12 cases), consistent with our annotation experience that many Reddit comments wrap actionable guidance inside personal stories. Emotional support is most often misclassified as appraisal (8 of 14 cases), likely because both categories involve evaluative or reflective language but differ in their intent to comfort versus judge.

Figure 2 shows the training and validation loss curves. Training loss decreases steadily across all 4 epochs, while validation loss plateaus after epoch

2 and begins to slightly increase, indicating the model is approaching the limits of what it can learn from 1,400 training examples without overfitting.

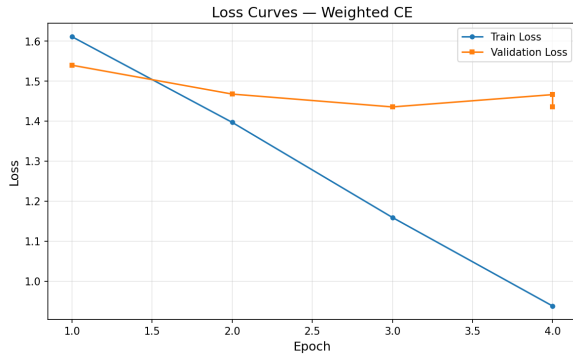


Figure 2: Training and validation loss curves for RoBERTa + Weighted CE.

## 7.2 Stacked Ensemble Error Patterns (Karamvir)

The ensemble shows a different error profile from RoBERTa. On majority classes it is competitive or slightly better: advice achieves an F1 of approximately 0.80 and anecdote approximately 0.66. However, its warning performance ( $F1 \approx 0.17$ ) is substantially worse than RoBERTa’s ( $F1 = 0.40$ ), while its emotional support performance ( $F1 \approx 0.33$ ) is higher. This trade-off likely reflects the different augmentation and loss strategies: RoBERTa’s class-weighted loss directly upweights warning examples during training, whereas the ensemble’s mixup augmentation in the embedding space may smooth over the sharp directive cues that distinguish warnings from other categories.

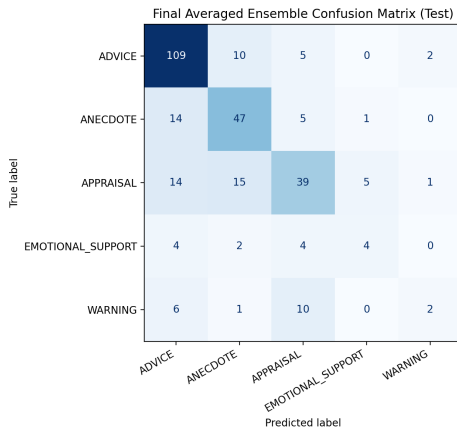


Figure 3: Normalized confusion matrix for the stacked ensemble on the test set.

The ensemble’s confusion matrix reveals that warning is most frequently misclassified as ap-

praisal (53%) and advice (32%), indicating that cautionary language is interpreted as general evaluation or suggestion. Emotional support predictions are distributed roughly evenly across advice, appraisal, and emotional support, suggesting the model cannot reliably isolate comforting intent from related categories.

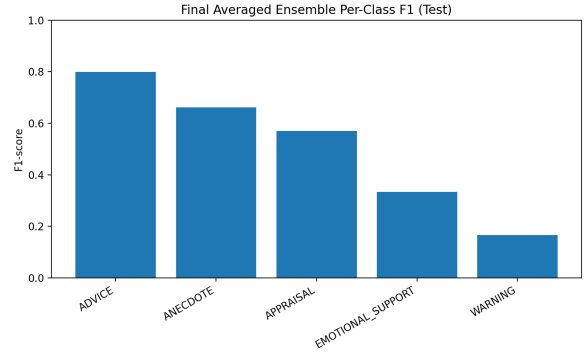


Figure 4: Per-class F1 scores for the stacked ensemble on the test set.

## 7.3 TextCNN Error Patterns (Aizaz)

Figure 5 shows the confusion matrix for the selected TextCNN configuration. The model correctly classified 91 of 126 advice comments ( $F1 = 0.703$ ) and 45 of 67 anecdote comments ( $F1 = 0.584$ ). Appraisal performance was weaker at 32 of 74 correct ( $F1 = 0.444$ ). On the minority classes, it correctly classified only 3 of 14 emotional support comments ( $F1 = 0.316$ ) and 1 of 19 warning comments ( $F1 = 0.083$ ).

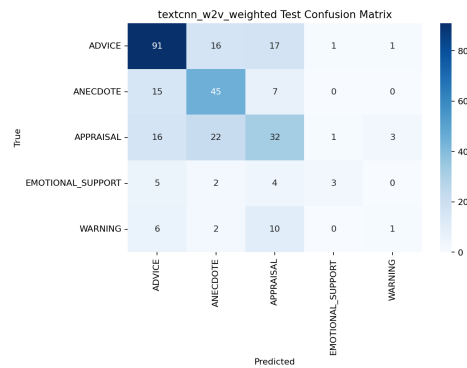


Figure 5: Confusion matrix for the selected TextCNN configuration on the test set.

The most common confusion pairs were appraisal → anecdote (22 cases), advice → appraisal (17 cases), appraisal → advice (16 cases), and advice → anecdote (16 cases). This shows that the model often struggled to separate communicative function when comments had similar wording or

narrative structure. Warning was the weakest class, with most warning comments predicted as appraisal or advice.

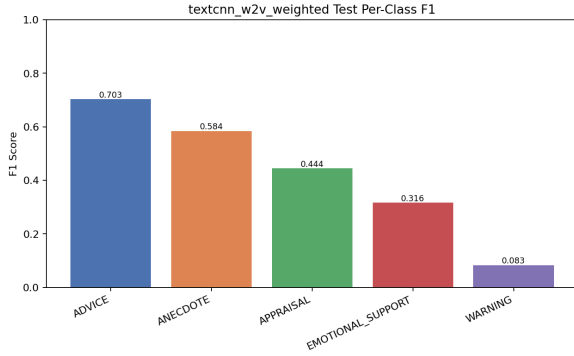


Figure 6: Per-class F1 scores for the selected TextCNN configuration on the test set.

The qualitative errors show the same pattern. A warning comment about a manufacturer warranty was predicted as advice, which suggests the model focused on the explanatory or directive wording instead of the cautionary intent. Several appraisal comments were predicted as anecdote because they included conversational or personal framing. Emotional support comments such as “Stay strong” were also often predicted as advice when they contained follow-up suggestions. Overall, the model learned useful local phrase patterns, but it had difficulty when the label depended more on intent than on obvious wording.

#### 7.4 BiLSTM Flair Error Patterns (Rayan)

Figure 7 shows the confusion matrix for Run 3, the strongest BiLSTM Flair configuration by macro F1. The model performs reasonably well on the three majority classes, correctly identifying 83 of 126 advice comments, 45 of 67 anecdotes, and 30 of 74 appraisals. However, like the other models in this project, it struggles much more on the minority classes. Only 3 of 14 emotional support comments and 3 of 19 warning comments are classified correctly.

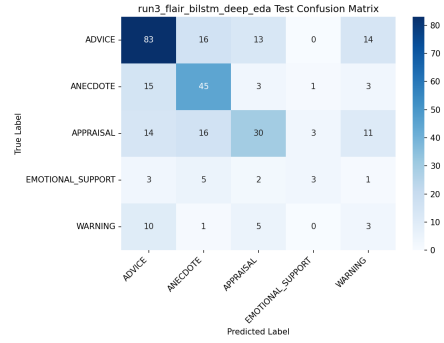


Figure 7: Confusion matrix for BiLSTM Flair Run 3 (Deep+EDA) on the test set.

One clear error pattern is that the model predicts warning too aggressively. Although there are only 19 true warning instances in the test set, a noticeable number of advice and appraisal comments are assigned to that class. This suggests that the model is reacting to cautionary or directive wording without always distinguishing whether the overall intent is actually warning, advice, or evaluation. Since Run 3 uses EDA on the minority classes, this may reflect a side effect of upsampling warning examples: the model becomes more willing to predict the class, which can improve recall slightly but hurts precision.

Emotional support is the other class the model handles poorly. These comments are often confused with anecdote, advice, or appraisal, which is consistent with the ambiguity we observed during annotation. Supportive comments often include reflective language, personal perspective, or indirect suggestions, so their wording overlaps heavily with neighboring categories. Even with pretrained contextual embeddings, the model does not seem able to separate comforting intent reliably when the number of training examples is this limited.

Figure 8 shows the training and validation loss curves for Run 3. Training loss decreases steadily across epochs, while validation loss begins to rise after the early stages of training. This is a clear sign of overfitting. In other words, the model continues to fit the training set more closely, but those gains do not transfer to unseen data.

This trend is especially important in our setting because the dataset is relatively small. Although the frozen Flair encoder provides strong pretrained token representations, the BiLSTM classifier on top is still fully trainable and can adapt too closely to the training examples. The loss curves therefore suggest that the pretrained embeddings are

helpful, but they are not enough on their own to prevent overfitting once the classifier becomes too specialized to the training set.

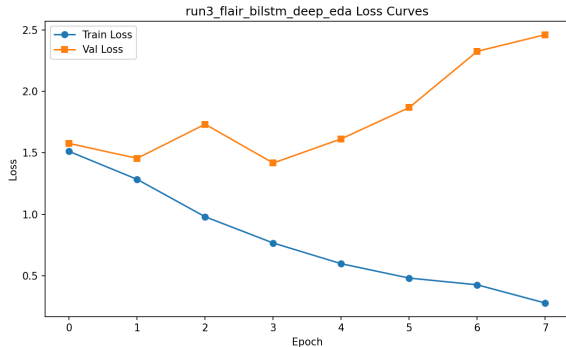


Figure 8: Training and validation loss curves for BiLSTM Flair Run 3 (Deep+EDA).

Figure 9 compares per-class F1 across all three BiLSTM Flair runs. Run 1 performs best on advice and appraisal, but completely fails to predict warning. Run 2 begins to recover warning predictions, but this comes with a large drop in overall performance on the majority classes. Run 3 gives the best overall balance by macro F1: it improves emotional support and remains competitive on advice, anecdote, and appraisal, even though warning is still difficult. Taken together, these runs suggest that augmentation alone was not enough to help the model, but augmentation combined with a deeper recurrent classifier was somewhat more effective.

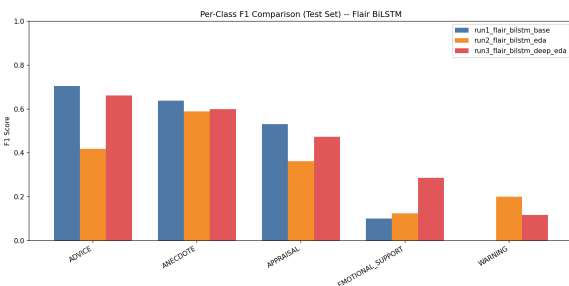


Figure 9: Per-class F1 comparison across all three BiLSTM Flair runs on the test set.

## 7.5 Shared Failure Modes

Despite their different architectures, the models share a core difficulty: distinguishing communicative intent when surface-level language is similar across categories. Table 5 shows representative misclassified examples observed across all models.

| True        | Pred      | Comment (truncated)                                                                                          |
|-------------|-----------|--------------------------------------------------------------------------------------------------------------|
| Advice      | Anecdote  | “Nerdwallet posted that the Discover savings is its #1 pick. I had to call Discover to set up my account...” |
| Anecdote    | Warning   | “If you have HSV your skin is shedding cells. I think come clean to her. Why risk her health?”               |
| Emot. Supp. | Appraisal | “Stay strong, I know breaking up with friends can be tough. Are all your posts about your brother?”          |
| Warning     | Advice    | “Unless she shows willingness to get help, leave. Don’t be a human sacrifice...”                             |
| Appraisal   | Advice    | “My advice: tell them... Your financial situation may not be great but you never know...”                    |

Table 5: Misclassified examples from RoBERTa + Weighted CE, illustrating error patterns shared across models.

In the first example, an advice comment beginning with a personal story was predicted as anecdote because the model latched onto narrative framing rather than the underlying recommendation. In the fourth, a warning containing “leave” was predicted as advice because the directive language resembled actionable guidance. The fifth example begins with “My advice:” but was labeled appraisal by the annotator because the content is evaluative rather than actionable. The model followed the surface cue and predicted advice. These patterns suggest that further gains will require either more training data for minority classes or architectures that can better separate pragmatic intent from lexical form.

## 7.6 Recommendations

Our error analysis points to several specific directions for future work. The most actionable is targeted data collection for emotional support and warning, the two classes where all models fail. With only 14 and 19 test instances respectively, even a single additional correct prediction substantially changes the F1 score, and neither EDA nor mixup augmentation meaningfully addressed this gap. Collecting 200–300 additional examples per minority class from the same subreddits would likely yield larger gains than any architectural change.

Second, the dominant confusion patterns suggest a hierarchical classification approach. The advice–warning and advice–appraisal confusions stem from shared directive or evaluative language, while the models handle the coarser distinction between narrative comments (anecdote) and non-

narrative ones more reliably. A two-stage classifier that first separates narrative from non-narrative comments, then distinguishes within each group (e.g., advice vs. warning vs. appraisal within non-narrative), could reduce errors in exactly the pairs where our models struggle most.

Third, many misclassifications arise because the same comment serves multiple communicative functions, the “My advice: tell them...” example scored as appraisal, and comments wrapping guidance inside personal stories causing bidirectional advice–anecdote confusion. Adopting multi-label annotation rather than forcing a single label with a priority system would better reflect the data and could improve both training signal and evaluation fairness.

Finally, incorporating thread context, the original post and parent comments could resolve ambiguities that are unresolvable from the comment text alone. A warning like “Don’t be a human sacrifice” reads as generic advice in isolation, but given a post describing an abusive relationship, its cautionary intent becomes clear. Prepending the parent post as additional input to the transformer would be a straightforward way to test this hypothesis.

## Team Contributions

Ahmad Allabadi developed and evaluated the RoBERTa models, including the experiments with weighted cross-entropy, focal loss, and EDA. He wrote the RoBERTa feature, implementation, results, and error analysis sections of the report.

Karamvir Hira developed the stacked ensemble model, including the combination of dense embeddings, word-level TF-IDF, and character-level TF-IDF features with a meta-learner. He wrote the corresponding feature, implementation, results, and error analysis sections for the ensemble.

Aizaz Lodhi implemented the TextCNN model and compared different embedding and loss configurations, including Word2Vec initialization and class-weighted training. He wrote the TextCNN feature, implementation, results, and error analysis sections of the report.

Rayan Nasrallah developed the BiLSTM with Flair embeddings model, including the experiments with frozen contextual embeddings, EDA, and deeper recurrent layers. He wrote the BiLSTM feature, implementation, results, and error analysis sections, and helped integrate figures, tables, and model comparisons into the final report.

All team members contributed to dataset collection, annotation, and label verification, and each member reviewed the final report before submission.

## References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Hongliu Cao. 2024. Recent advances in universal text embeddings: A comprehensive review of top-performing methods on the mteb benchmark. *arXiv preprint arXiv:2406.01607*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.
- William Duffy. 2025. Evaluating rule-based and generative data augmentation for text classification. *Knowledge and Information Systems*.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*.
- Yevhen Kostyuk, Clara Seyfried, and Chris Reed. 2025. Automating alternative generation in decision-making. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.
- Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, UK.

- Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–374.
- Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.
- Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388.
- David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.