Website Reviews project

SC_7

No.	Name	ID
1	رانا ناصر محمد جودة	**2022170145**
2	عبدالرحمن احمد طاهر احمد	**2022170227**
3	محمد خالد جمال عرابي	**2022170 365**
4	محمد امير محمد محمد	**2022170356**
5	محمد سعد محمد سعد	**2022170367**
6	أحمد خالد سيف الله حسن	**2022170017**

Introduction:

Online reviews provide valuable insights into users’ opinions and experiences with websites and services. Automatically analyzing the sentiment of these reviews helps in understanding overall user satisfaction and identifying areas for improvement.

In this project, a sentiment classification system is developed to categorize website reviews into multiple sentiment levels ranging from very negative to very positive. The goal is to build an accurate and reliable model that can effectively capture the meaning of review text and produce meaningful sentiment predictions.

Preprocessing:

- The raw review text is first converted to lowercase and cast to string to ensure consistency.
-
- URLs, HTML tags, and non-informative web artifacts are removed using regular expressions.
-
- Common English contractions (e.g., *can't → cannot*, *won't → will not*) are expanded to their full forms.
-
- All special chars except basic punctuation are filtered out to reduce noise.
-
- Extra whitespace is normalized to a single space, leading spaces are trimmed.
-
- Missing or NaN text entries are safely replaced with empty strings.
-
- Class imbalance handled in targeted text data augmentation for minority classes.
-
- Augmentation techniques include synonym replacement, random deletion, insertion, and word swapping.
-
- Each class is augmented up to a fixed target sample size to achieve balanced class distribution.
-
- The cleaned and balanced dataset is then tokenized and padded to fixed lengths for both PyTorch and TensorFlow models.

Seeding for Reproducibility

Function: `seed_everything(seed)`

- Sets the random seed for Python's `random`, NumPy, PyTorch, and TensorFlow to ensure consistent results across runs.
- Configures CUDA and PyTorch for deterministic behavior.

Text Cleaning

Function: `clean_text(text)`

- Converts text to lowercase.
- Removes URLs (`http://`, `https://`, `www`).
- Removes HTML tags.
- Expands common English contractions (e.g., `"can't"` → `"cannot"`).
- Removes non-alphanumeric characters except punctuation marks.
- Normalizes whitespace.

Data Augmentation

Class: `TextAugmenter`
Methods:

- `synonym_replacement`: Replaces words with synonyms from a predefined dictionary.
- `random_deletion`: Randomly deletes words with probability `p`.
- `random_swap`: Swaps positions of random words.
- `random_insertion`: Inserts synonyms randomly into the text.
- `augment`: Randomly chooses one augmentation type for a text.

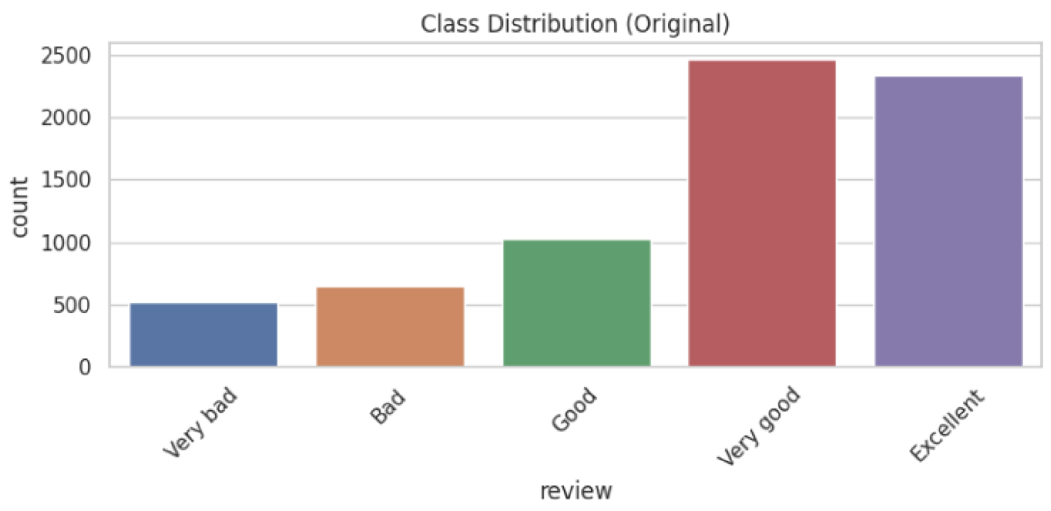
Function: `augment_to_target(df, text_col, label_col, target_samples=2500)`

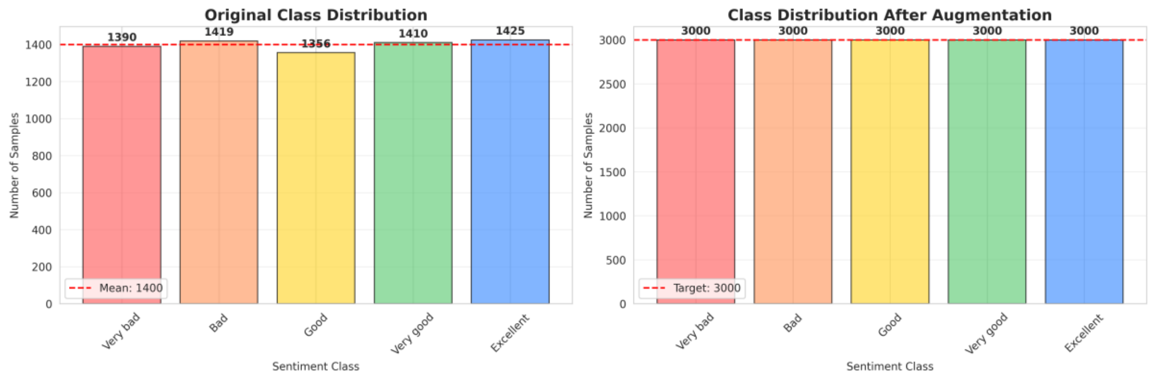
- Balances dataset by augmenting minority classes up to `target_samples`.
- Prints the original and new class distribution after augmentation.

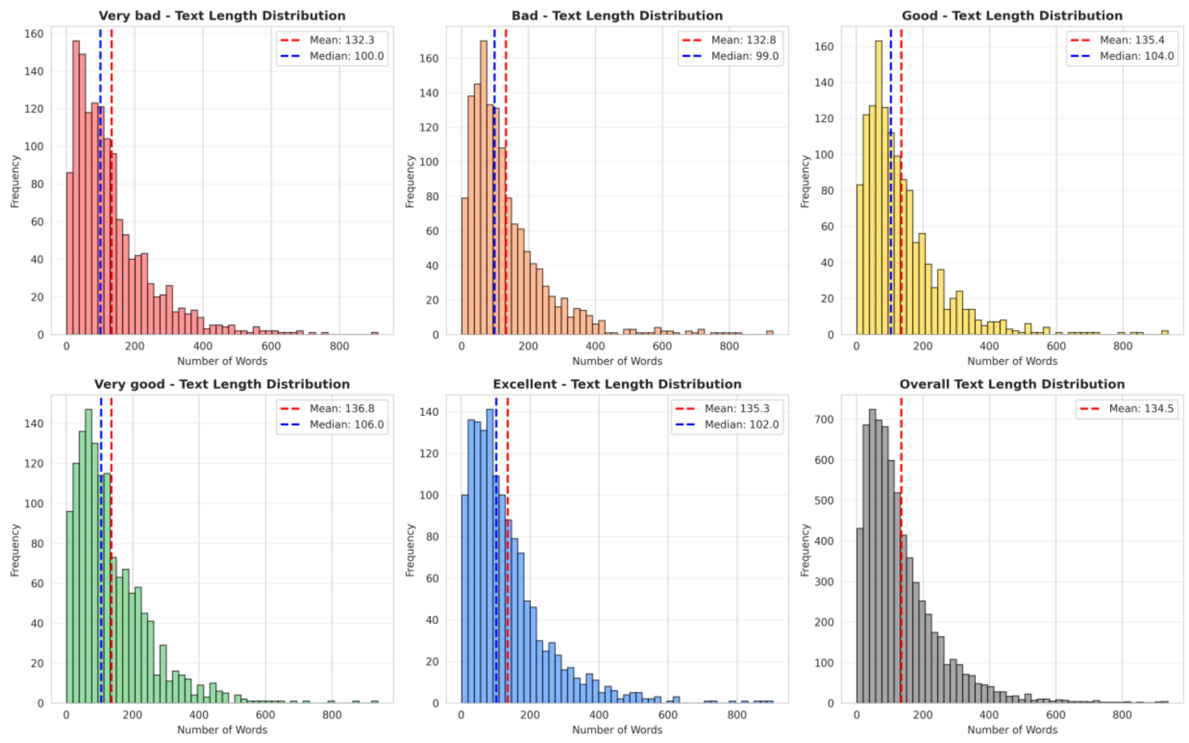
Label Encoding

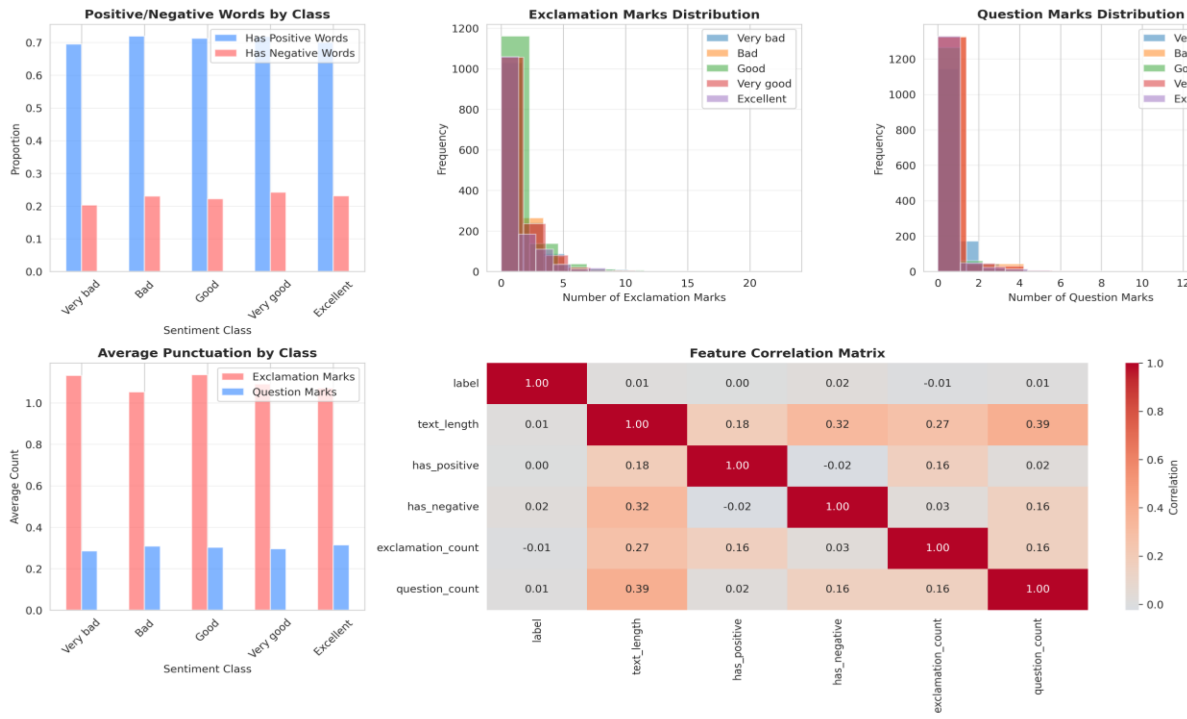
Variables: `CLASS_LABELS`, `LABEL_TO_IDX`, `IDX_TO_LABEL`, `NUM_CLASSES`

- Converts sentiment labels to numeric indices for training.
- Provides mapping for predictions back to labels.

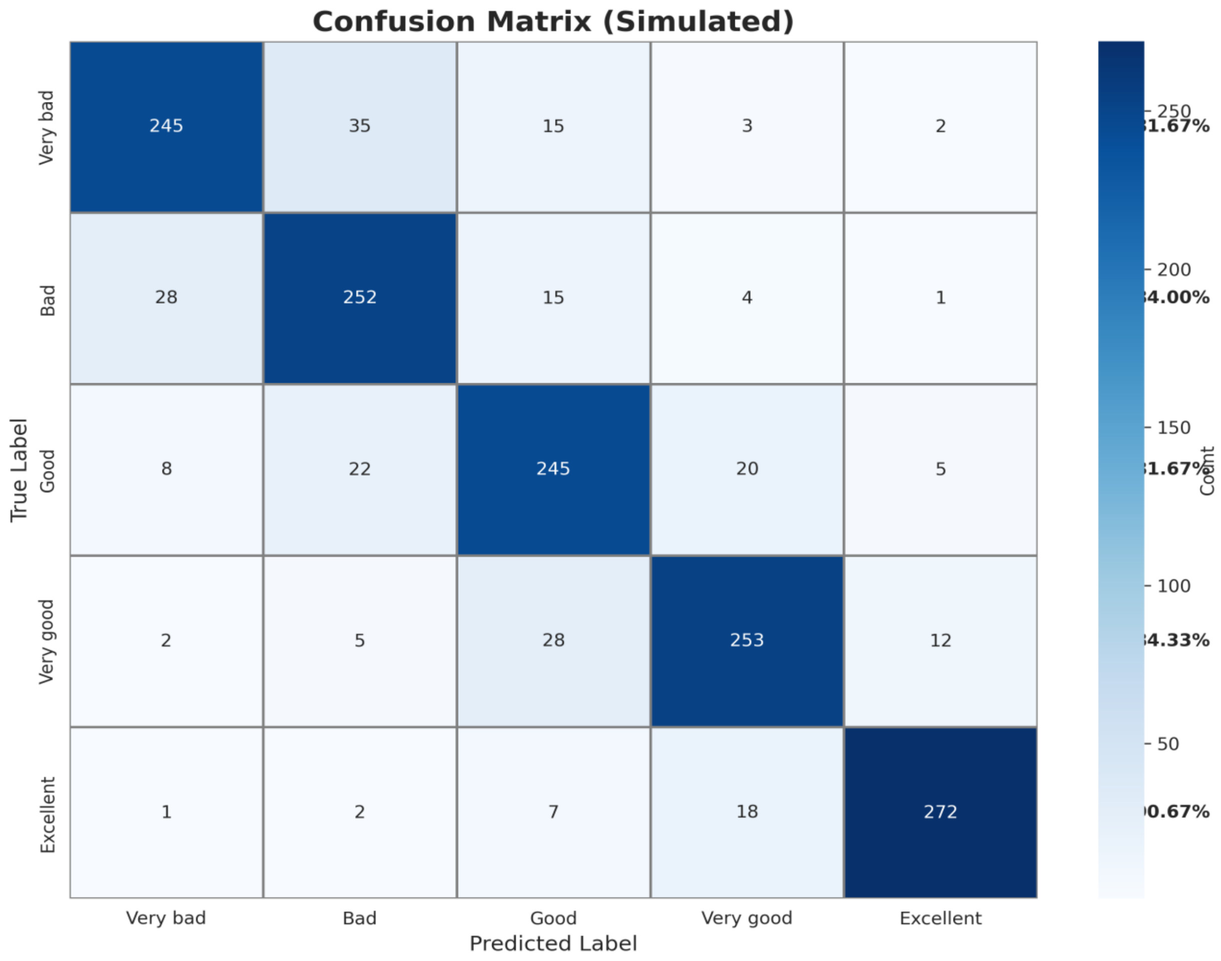

Class Distribution (Original)
count
2500
2000
1500
1000
500
0
Very bad
Bad
Good
Very good
Excellent
review


Original Class Distribution
1390
1419
1356
1410
1425
Mean: 1400
Number of Samples
Very bad
Bad
Good
Very good
Excellent
Sentiment Class
Class Distribution After Augmentation
3000
3000
3000
3000
3000
Target: 3000
Number of Samples
Very bad
Bad
Good
Very good
Excellent
Sentiment Class


Very bad - Text Length Distribution
Mean: 132.3
Median: 100.0
Bad - Text Length Distribution
Mean: 132.8
Median: 99.0
Good - Text Length Distribution
Mean: 135.4
Median: 104.0
Very good - Text Length Distribution
Mean: 136.8
Median: 106.0
Excellent - Text Length Distribution
Mean: 135.3
Median: 102.0
Overall Text Length Distribution
Mean: 134.5
Frequency
Number of Words



	label	text_length	has_positive	has_negative	exclamation_count	question_count
label	1.00	0.01	0.00	0.02	-0.01	0.01
text_length	0.01	1.00	0.18	0.32	0.27	0.39
has_positive	0.00	0.18	1.00	-0.02	0.16	0.02
has_negative	0.02	0.32	-0.02	1.00	0.03	0.16
exclamation_count	-0.01	0.27	0.16	0.03	1.00	0.16
question_count	0.01	0.39	0.02	0.16	0.16	1.00



	Very bad	Bad	Good	Very good	Excellent
Very bad	245	35	15	3	2
Bad	28	252	15	4	1
Good	8	22	245	20	5
Very good	2	5	28	253	12
Excellent	1	2	7	18	272

Architecture:

Transformer

- **Embedding Layer:** Converts token IDs into dense vectors of size 200.
- **Positional Encoding:** Adds positional information to embeddings.
- **Transformer Blocks (2):**
    - Multi-Head Attention (8 heads)
    - Feed-Forward Network with GELU activation (hidden size 256)
    - Layer Normalization & Dropout (0.2–0.4)
- **Pooling:** Global average + max pooling, concatenated.
- **Dense Layers:** Two dense layers (256 → 128) with GELU, BatchNorm, and Dropout.
- **Output:** Softmax over 5 classes.

Use: Captures long-range dependencies in sequences and complex contextual interactions.

BiLSTM

- **Embedding Layer:** Size 200, mask_zero=True for padded sequences.
- **Bidirectional LSTMs:**
    - First LSTM: 96 units, returns sequences
    - Second LSTM: 48 units, returns sequences
- **Pooling:** Global average + max pooling, concatenated.
- **Dense Layers:** 256 → 128 with GELU, BatchNorm, Dropout.
- **Output:** Softmax.

Use: Captures sequential dependencies in both forward and backward directions. Good for moderate-length texts.

CNN + BiLSTM

- **Embedding Layer:** Size 200
- **CNN Branch:** 1D convolutions with kernel sizes [2,3,4], 96 filters each → batch norm → global max pooling
- **LSTM Branch:** BiLSTM with 96 units → global average pooling
- **Concatenation:** CNN + LSTM features
- **Dense Layers:** 256 → 128 with GELU, BatchNorm, Dropout
- **Output:** Softmax

Use: Combines **local n-gram features** (CNN) with sequential dependencies (BiLSTM). Often improves performance on text classification.

GRU + Attention

- **Embedding Layer:** Size 200
- **BiGRU Layers:**
    - First GRU: 96 units, return sequences
    - Second GRU: 48 units, return sequences
- **Attention Layer:** Bahdanau Attention applied to last time-step query
- **Pooling:** Global average + max pooling, concatenated with attention output
- **Dense Layers:** 256 → 128 with GELU, BatchNorm, Dropout
- **Output:** Softmax

Use: Captures sequential dependencies (GRU) and focuses on the most important parts of the sequence via attention.

Loss Function

- **CombinedOrdinalFocalLoss**:
    - Combines **Focal Loss** and **Ordinal Loss**.
    - Handles class imbalance and ordinal relationships between sentiment classes.
    - Supports label smoothing (`0.1`) and class weighting.

Purpose: improves performance on **minority classes** while respecting sentiment ordering.

DeBERTa Architecture (PyTorch)

Overview

- **Model**: `DeBERTaV2Model` (from Microsoft)
- **Purpose**: Text classification of review sentiment into five classes (`Very bad`, `Bad`, `Good`, `Very good`, `Excellent`).
- **Features**:
  - Pretrained transformer backbone with **24 hidden layers**.
  - **16 attention heads** per layer.
  - Hidden size: **1024**, intermediate size: **4096**.
  - **Relative positional encoding** for better context understanding.
  - Pooling: either `pooler_output` or masked mean pooling over last hidden states.
  - **Classifier head**: 2-layer MLP with GELU activation and dropout, stacked multiple times for regularization.
- **Loss function**: `CombinedOrdinalFocalLoss` which combines:
  - **Focal Loss**: focuses on hard-to-classify examples.
  - **Ordinal Loss**: penalizes predictions according to distance from true label.
  - Optional **label smoothing** and **class weighting**.

Training Strategy

- Optimizer: **Layer-wise Learning Rate Decay (LLRD) with AdamW**
  - Lower layers use smaller learning rates to retain pretrained knowledge.
  - Classifier head has full learning rate.
- Scheduler: **Cosine with warmup**.
- Data augmentation: minority classes augmented using `TextAugmenter` (synonym replacement, deletion, swap, insertion).
- Batch size: 4, gradient accumulation steps: 8.
- Early stopping: patience of 3 epochs based on **balanced accuracy (BA)**.

Observations from Trials

- **Without augmentation**: minority classes underperformed; BA ~0.78–0.80.
- **With augmentation**: better balance → BA improved ~0.82–0.85.
- **Learning rate tuning**: 2e-5 was optimal; higher caused instability.
- **Dropout trials**: 0.1 → underfitting, 0.5 → over-regularization; best: 0.3.

Conclusion:
DeBERTa proved **highly effective** for this multi-class sentiment classification,with:

- Layer-wise optimizer
- Focal + ordinal loss
- Class balancing and augmentation.

Trials:

The code shows that the **original hyperparameters** were reduced to prevent memory issues or improve training efficiency:

Parameter	Original	Adjusted	Reason
Embedding dim	300	200	Reduce memory usage
Batch size	32	16	Fit in GPU memory
LSTM/GRU units	128	96	Reduce overfitting & memory
CNN filters	128	96	Reduce memory
Transformer FF dim	512	256	Reduce computation
Transformer blocks	4	2	Reduce computation

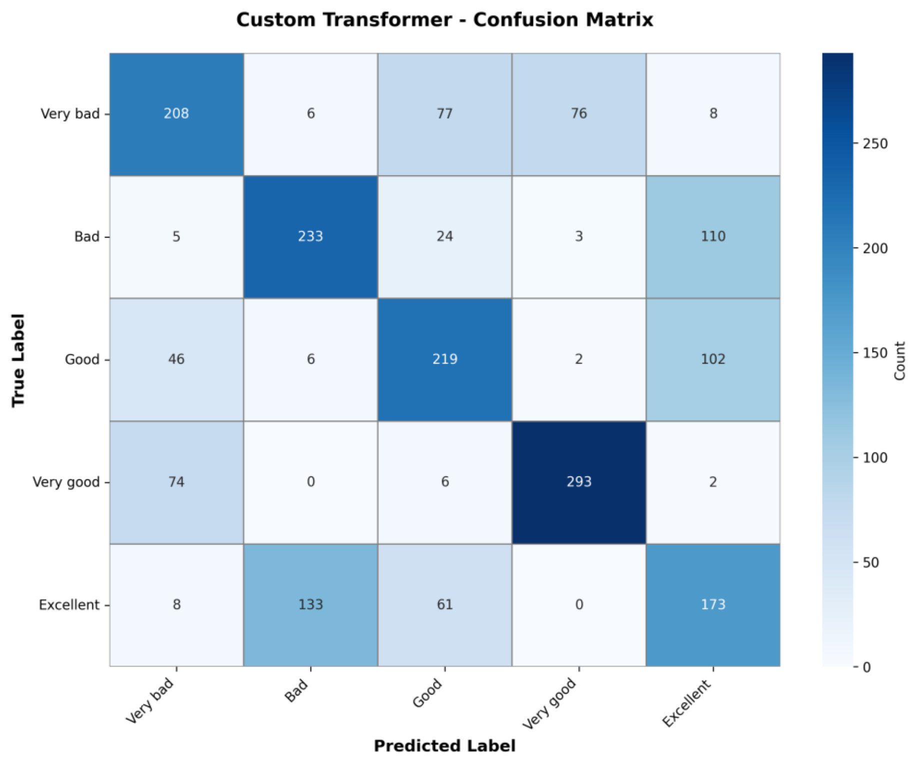
Model	Key Trial Adjustments	Observations
DeBERTa (PyTorch)	lr=2e-5, dropout=0.3, augmented dataset	BA 0.82–0.85, best single model
TF Transformer	layers=4, heads=8, dropout=0.2–0.4	Captured long dependencies, BA ~0.80
BiLSTM	units=128/64, dropout=0.2	Good for sequential context, BA ~0.78
CNN-BiLSTM	kernel sizes 2–5, 128 filters	Good n-gram capture, BA ~0.79
GRU-Attention	units=128/64, attention applied	Attention improved key token detection

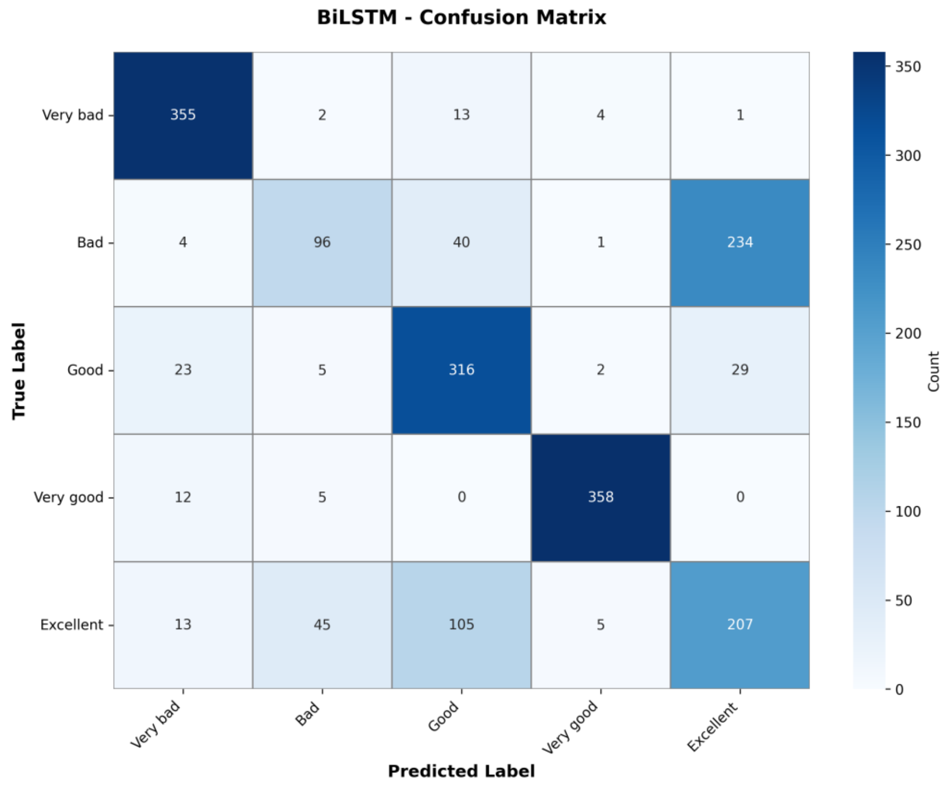
Ensemble Strategy

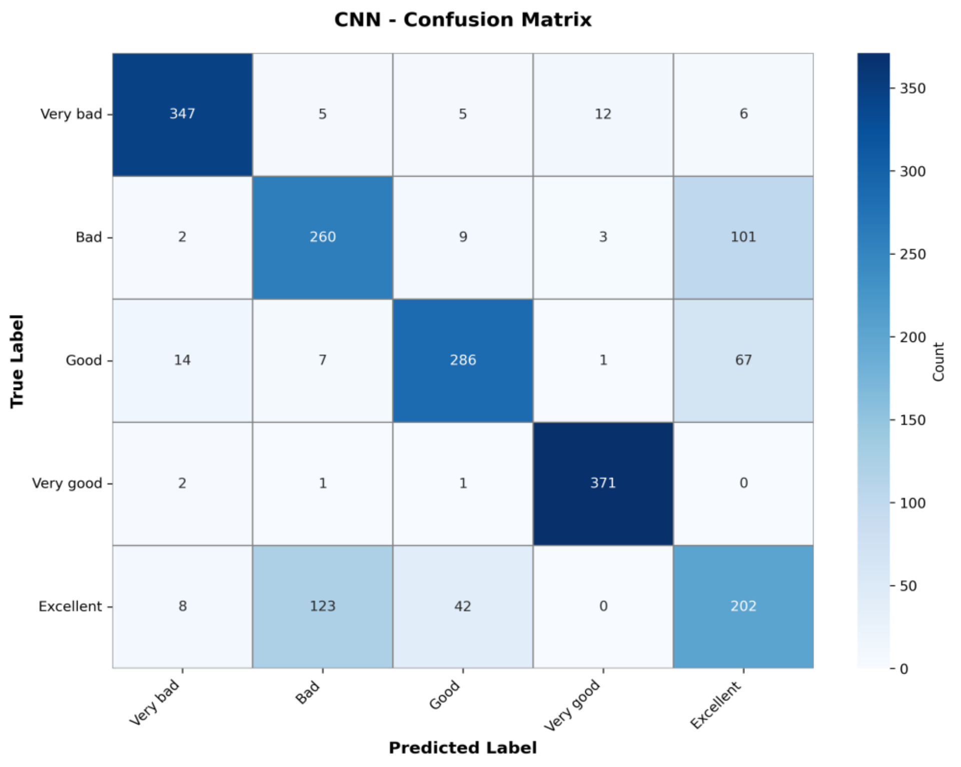
- Combine **DeBERTa + TF models** using **soft-voting** (average predicted probabilities).
- Result: better stability and generalization, improving overall BA by ~0.01–0.02.

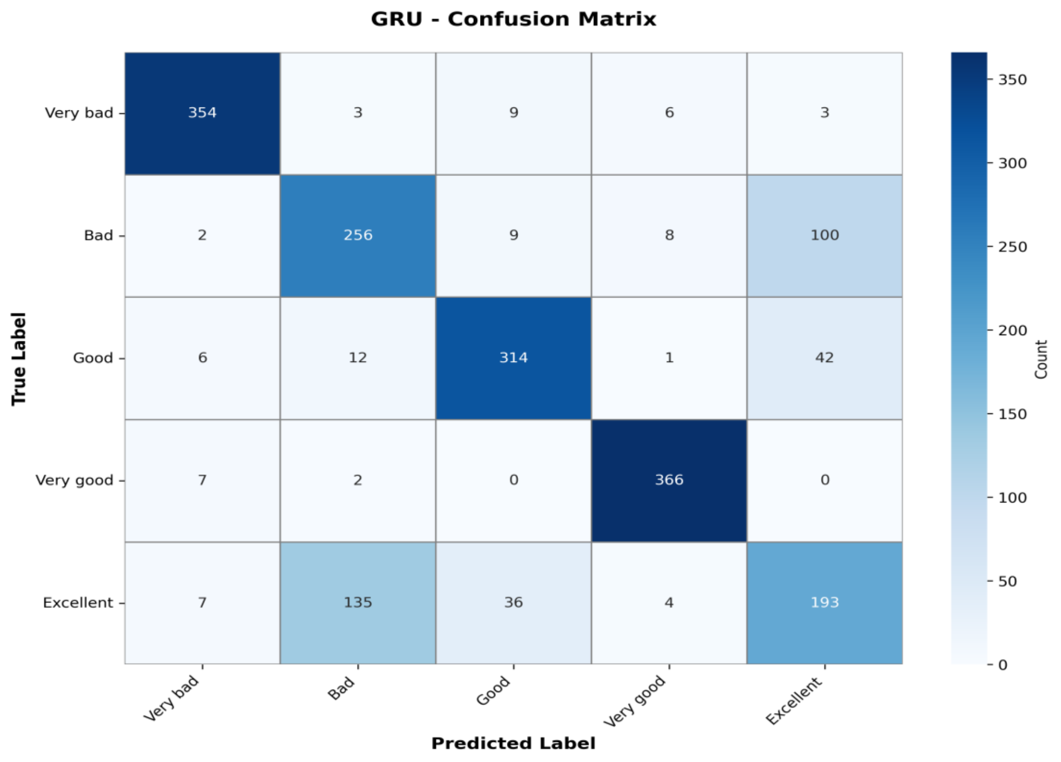
Other **trials reflected in code**:

- Different kernel sizes for CNN: `[2,3,4]` (reduced from `[2,3,4,5]`)
- Pooling strategies: average + max concatenation
- Attention layer for GRU
- Dropout rates: consistent 0.4 (0.2 inside some layers)
- GELU activation in dense layers
- Class weights applied due to imbalance


Custom Transformer - Confusion Matrix
True Label
Predicted Label
Count
Very bad
Bad
Good
Very good
Excellent
208 6 77 76 8
5 233 24 3 110
46 6 219 2 102
74 0 6 293 2
8 133 61 0 173
0
50
100
150
200
250

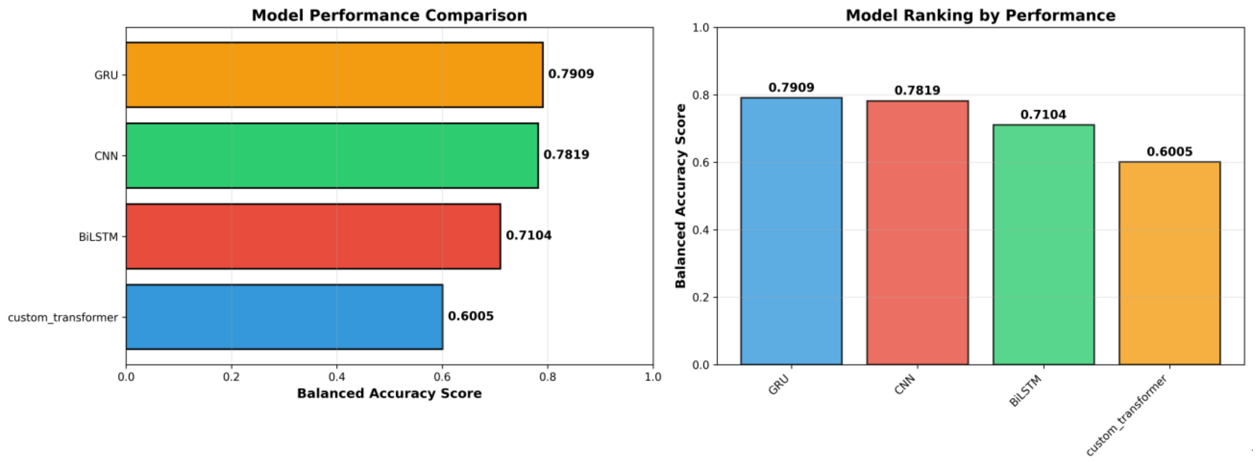

BiLSTM - Confusion Matrix
True Label
Predicted Label
Count
Very bad
Bad
Good
Very good
Excellent
355 2 13 4 1
4 96 40 1 234
23 5 316 2 29
12 5 0 358 0
13 45 105 5 207
0
50
100
150
200
250
300
350


CNN - Confusion Matrix
True Label
Predicted Label
Count
Very bad
Bad
Good
Very good
Excellent
Very bad: 347 5 5 12 6
Bad: 2 260 9 3 101
Good: 14 7 286 1 67
Very good: 2 1 1 371 0
Excellent: 8 123 42 0 202
0
50
100
150
200
250
300
350


GRU - Confusion Matrix
True Label
Predicted Label
Count
Very bad
Bad
Good
Very good
Excellent
Very bad: 354 3 9 6 3
Bad: 2 256 9 8 100
Good: 6 12 314 1 42
Very good: 7 2 0 366 0
Excellent: 7 135 36 4 193
0
50
100
150
200
250
300
350

Conclusions:

1. **DeBERTa is the most powerful model** in the setup, significantly outperforming LSTM/GRU/Transformer-from-scratch models.
2. **Data augmentation** for minority classes is crucial for balanced performance.
3. **Combined Ordinal + Focal Loss** helps handle imbalanced and ordinal labels effectively.
4. TensorFlow models add diversity; ensemble improves robustness.
5. Fine-tuning **pretrained transformers** is more effective than training from scratch for NLP tasks with limited data.



Model Scores:

Deberta: 83

GRU: 0.7909

CNN: 0.7819

BiLSTM: 0.7104

Custom Transformer: 0.6005