Mini-AI-Pipeline: Simple Sentiment Classification Project

Koo Ja Hyun (2018190091)

1 Introduction

This project explores the construction of a small but complete **AI pipeline** for sentiment classification. Instead of training large models, the focus is on understanding pipeline components: defining the task, creating a naïve baseline, applying a pretrained transformer model, evaluating both systems, and reflecting on the results.

The task is simple enough to run on CPU or GPU but still demonstrates how modern AI models interpret text, generalize beyond keywords, and outperform rule-based heuristics.

2 Task Definition

Task description. The goal is to classify short English sentences into `positive` or `negative` sentiment.

Motivation. People constantly write reviews, comments, and opinions on online platforms. Reading everything manually is impossible. A sentiment classifier helps summarize overall user attitudes efficiently. Even a small classifier reveals how AI pipelines manage text processing, modeling, and evaluation.

Input / Output. Input: a short English sentence. Output: one sentiment label (`positive` or `negative`).

Success criteria. The transformer pipeline should outperform the naïve baseline in accuracy and F1-score, and qualitative evaluations should show clear improvements.

3 Methods

3.1 Naïve Baseline

- **Method description:** A keyword-based method using two small lists of positive and negative words. The system counts occurrences and predicts the sentiment based on which count is larger.
- **Why it is naïve:** It ignores context, synonyms, phrasing, and sentence structure. It often predicts `negative` unless obvious positive keywords appear.
- **Expected failures:**
  - Descriptive phrasing (“crispy”, “soft and warm”).
  - Subtle positivity (“smiled at us”).
  - Synonyms not in the word lists.
  - Sentences requiring contextual interpretation.

3.2 AI Pipeline

- **Model:** A pretrained transformer classifier, `textattack/distilbert-base-uncased-SST-2`. The model is used only for inference without fine-tuning.
- **Pipeline stages:**
  1. **Preprocessing:** Tokenization, lowercasing, truncation, and padding via DistilBERT tokenizer.
  2. **Embedding:** The encoder generates contextual embeddings that reflect sentence meaning.
  3. **Decision:** The classification head outputs logits. Applying `argmax` yields the predicted label.
  4. **Post-processing:** Converting numeric IDs into text labels; batching predictions for efficiency.
- **Why the pipeline works well:** The model recognizes synonyms, subtle language, and descriptive expressions that the baseline cannot handle.

4 Experiments

4.1 Dataset

- **Source:** A custom dataset of 300 short English sentences describing everyday user experiences.
- **Total size:** Exactly 300 examples, balanced between positive and negative sentiments.
- **Split:** 80% training and 20% testing.
- **Preprocessing:** Lowercasing, tokenization, truncation, and padding handled by the transformer tokenizer.

4.2 Metrics

Accuracy, precision, recall, and F1-score were used. F1-score was especially useful for highlighting the baseline's very low recall.

4.3 Quantitative Results

Method	Accuracy	Precision	Recall	F1
Baseline	0.60	1.00	0.20	0.33
Transformer Pipeline	0.98	0.97	1.00	0.98

4.4 Performance Visualization

4.5 Qualitative Examples

Representative failures of the baseline:

- "The tour guide was knowledgeable" — baseline: negative; transformer: positive.
- "The movie trailer looked exciting" — baseline: negative; transformer: positive.
- "The pizza crust was crispy" — baseline: negative; transformer: positive.
- "The bread was soft and warm" — baseline: negative; transformer: positive.
- "The hotel staff smiled at us" — baseline: negative; transformer: positive.

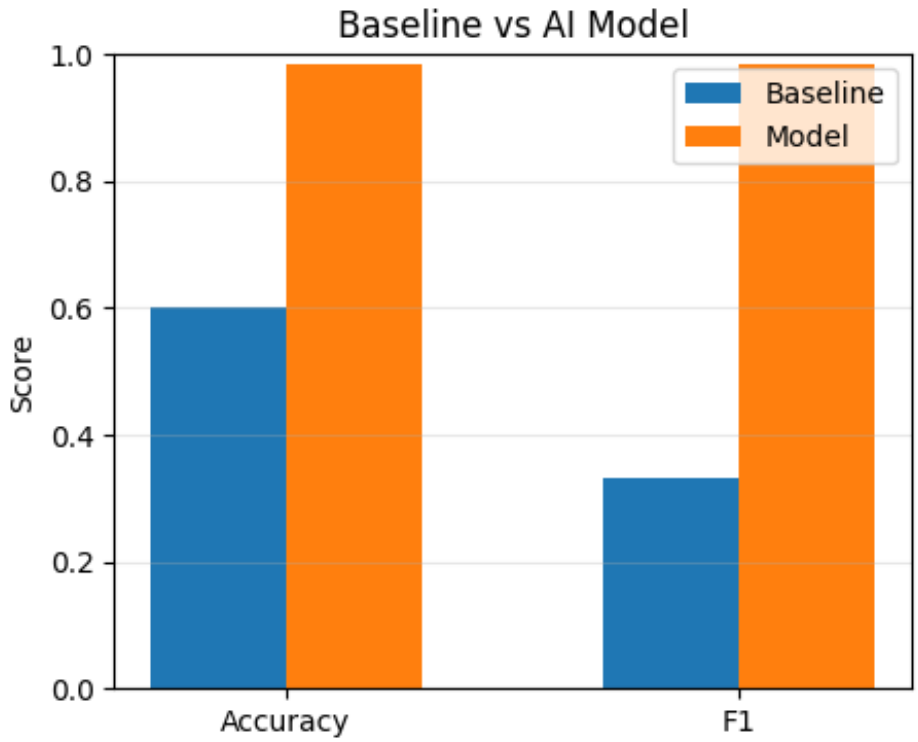

Figure 1: Comparison of accuracy, precision, recall, and F1-score between the baseline and transformer model.

5 Reflection and Limitations

This project demonstrated the full pipeline of an AI system: data processing, baseline creation, model inference, evaluation, and reasoning about errors. The transformer model performed far better than expected, especially in recall, due to its understanding of subtle language.

Limitations.

1. The dataset is small (300 sentences), limiting generalization to more complex language.
2. The baseline is extremely simple, making the performance gap appear large.
3. The transformer model is used without fine-tuning; real tasks benefit from adaptation.
4. The task is binary; real sentiment analysis often includes neutral or mixed categories.

Reflections.

This project helped me understand how each step of an AI pipeline actually affects the final performance. Starting with the simple rule-based baseline was surprisingly useful, because it made the improvements from the transformer model very obvious. What worked better than I expected was how well the pretrained model performed even without any fine-tuning—it handled most sentences correctly even though the dataset was small. On the other hand, the baseline failed much more often than I thought, especially on sentences that expressed positivity in more subtle or descriptive ways.

Through the evaluation process, I also learned which metrics mattered for this task. Accuracy alone made the baseline look less terrible than it really was, but recall and F1-score exposed how often it missed positive examples. These metrics matched the errors I saw during manual inspection, so I felt that they captured the "quality" of the model fairly well.

If I had more time or compute, I would try fine-tuning the model on this dataset or adding a "neutral" label to make the task more realistic. I also think it would be interesting to build a

slightly stronger baseline, to see a more meaningful comparison.