Mini AI Spam Filter

Kyoungjin Oh (2022148070)

1 Task Definition

- **Task description:** What are you trying to do?

  The goal is to build a binary text classification system that automatically categorizes SMS messages into two classes: Ham (legitimate) and Spam (unwanted/malicious).

- **Motivation:** Why is this task interesting or useful?

  Spam is not just annoying; it poses security risks like phishing. Manual filtering is inefficient, so an automated AI filter is essential to improve user experience and security.

- **Input / Output:** What does the model see, and what should it produce?
  - **Input:** A raw SMS text string.
  - **Output:** A binary label (0 for Ham, 1 for Spam).

- **Success criteria:** How do you know if your system is "good"?

  Since the dataset is imbalanced (mostly Ham), High Precision is critical to avoid blocking important legitimate messages. The system is considered successful if the AI pipeline achieves a significantly higher F1-score than the rule-based baseline.

2 Methods

This section includes both the naïve baseline and the improved AI pipeline.

2.1 Naïve Baseline

- **Method description:**

  Instead of manually guessing keywords, I implemented a data-driven heuristic using frequency analysis tools from Scikit-Learn [1]. By analyzing the training data, I selected the top 10 words that appear frequently in spam but rarely in ham (e.g., free, txt, claim). If a message contains any of these keywords, it is classified as spam.

- **Why naïve:**

  It treats words in isolation and ignores context or sentence structure. It cannot understand multiple meanings of a word or conversational nuance.

- **Likely failure modes:**
  - **Contextual False Positives:** Normal messages using trigger words (e.g., "Are you free tonight?") are wrongly marked as spam.
  - **Keyword Evasion (False Negatives):** Spammers can easily bypass this by using synonyms or misspellings (e.g., "F-R-E-E").

2.2 AI Pipeline

- **Models used:**
  - **Embedding:** sentence-transformers/all-MiniLM-L6-v2 [2, 3]
  - **Classifier:** Logistic Regression [1]
- **Pipeline stages:**
  1. **Preprocessing:** The raw input text is handled by the SentenceTransformer tokenizer [4]. This step automatically performs lowercasing, tokenization, and adds special tokens required by the BERT architecture.
  2. **Embedding:** The pre-trained all-MiniLM-L6-v2 model [3] processes the tokenized input. It converts the variable-length text into a fixed-size 384-dimensional dense vector.
  3. **Decision Component:** The extracted vectors are fed into a Logistic Regression classifier [1], which calculates the probability of the vector belonging to the spam class.
  4. **Post-processing:** The system applies a standard threshold (default 0.5) to determine the final label.
- **Design choices and justification:**
  - **Efficiency:** I selected all-MiniLM-L6-v2 because it offers an optimal balance between speed and accuracy. It is significantly faster and smaller (approx. 80MB) than full-sized BERT models [4], making it suitable for a lightweight pipeline.
  - **Transfer Learning Approach:** Instead of training from scratch, I used the pre-trained model as a feature extractor. Since the model already knows English semantics well, combining it with a simple linear classifier is sufficient to achieve high performance without overfitting.

3 Experiments

3.1 Datasets

- **Source:**

  The 'sms_spam' dataset from Hugging Face [5], containing 5,574 labeled SMS messages.
- **Total examples:**

  It has one collection composed by 5,574 English, real and non-encoded messages, tagged according being legitimate (ham) or spam.
- **Train/Test split:**

  Training set (80%) and Test set (20%) using a random seed of 42.
- **Preprocessing steps:**
  - **Naïve Baseline:** Lowercasing and keyword extraction based on frequency.
  - **AI Pipeline:** Automatic tokenization and vectorization using the Transformer model.

3.2 Metrics

Selected metrics align with the goal of prioritizing user trust:

- **Precision:** The most critical metric. It ensures legitimate emails are not blocked (False Positives).

- **Recall:** Measures how many spam messages are successfully caught.
- **F1-Score:** The harmonic mean of Precision and Recall, used as the primary benchmark for fair comparison.

3.3 Results

Table 1 summarizes the results. While the naïve baseline achieved reasonable Recall by catching obvious keywords, it suffered from low Precision due to frequent false alarms. In contrast, the AI Pipeline achieved perfect Precision (1.00) and a significantly higher F1-score (0.95). This demonstrates that MiniLM's semantic understanding effectively distinguishes spam even when they share common vocabulary.

Table 1: Performance comparison results

Method	**Precision**	**Recall**	**F1-score**
Baseline	0.79	0.81	0.80
AI Pipeline	1.00	0.91	0.95

Figure 1 [6, 7] visualizes the metric comparison. The bar chart clearly highlights the gap in F1-score, which is the most reliable metric for this imbalanced dataset.

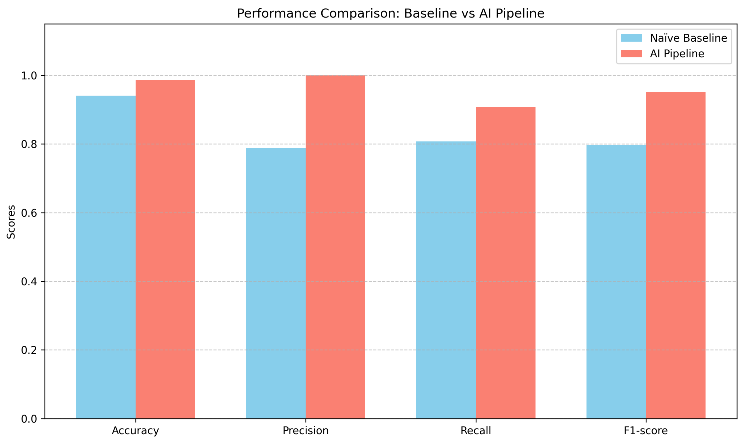

Figure 1: Performance comparison between naïve baseline and AI pipeline.

Additionally, Figure 2 (Confusion Matrices) [8, 9] confirms that the AI pipeline drastically reduced False Positives compared to the baseline, which is critical for maintaining user trust.

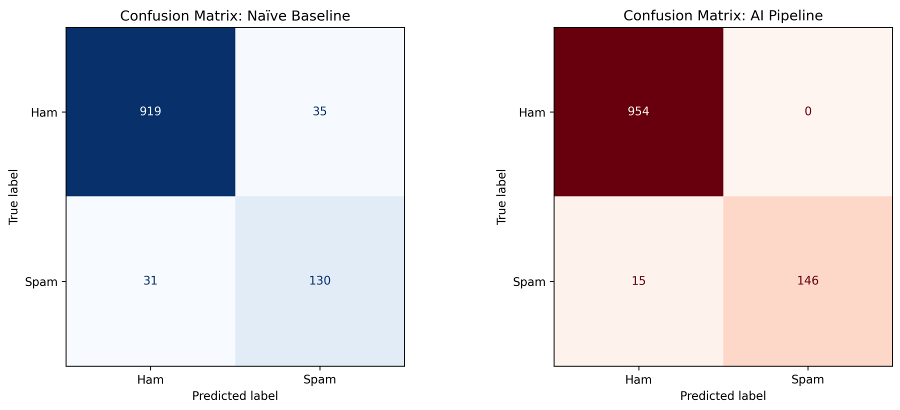

Figure 2: Confusion matrices for baseline and AI pipeline.

To better understand the limitations of the baseline and the strengths of the AI pipeline, I analyzed specific cases where the two methods produced different predictions.

- **Example 1 (Context)**
  - **Text:** "K, wen ur **free** come to my home and also tel vikky i hav sent mail to him also.. Better come evening il be free today aftr 6pm..:-)"
  - **Baseline:** Spam (Error due to "free").
  - **AI Pipeline:** Ham (Correct). It understood the context of making plans.
- **Example 2 (Unseen words)**
  - **Text:** "Call Germany for only 1 pence per minute! Call from a fixed line via access number 0844 861 85 85. No prepayment. Direct access!"
  - **Baseline:** Ham (Error). The specific words were not in the keyword list.
  - **AI Pipeline:** Spam (Correct). It captured the semantic meaning of a promotional offer.
- **Example 3 (Ambiguity)**
  - **Text:** "Dear,shall **mail** tonite.busy in the street,shall update you tonite.things are looking ok.varunnathu edukkukayee raksha ollu.but a good one in real sense."
  - **Baseline:** Spam (Error due to "mail").
  - **AI Pipeline:** Ham (Correct). It recognized the informal, personal tone.

4 Reflection and Limitations

The AI pipeline worked better than expected, achieving perfect Precision. This proves that even a small pre-trained model can capture semantics that keywords miss. However, I faced technical challenges with library version conflicts during implementation, which required troubleshooting standard Hugging Face instructions. Precision and F1-score proved to be much better indicators of quality than accuracy for this imbalanced dataset.

The primary limitation of the Naïve Baseline was its inability to understand context, leading to failures where benign words like "free" triggered false alarms. Regarding the AI pipeline, although it uses a frozen pre-trained encoder with a linear classifier, it achieved a perfect Precision of 1.00. However, there is still room for improvement in Recall (0.91), indicating that the model occasionally misses subtle spam messages. Additionally, the model might be vulnerable to adversarial attacks (e.g., intentional misspellings like "F-R-E-E"). If I had more time and compute resources, I would try fine-tuning the entire Transformer model end-to-end to capture these missed cases and boost Recall, while also exploring data augmentation techniques to improve robustness against such evasion tactics.

References

[1] F. et al. Pedregosa. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://scikit-learn.org/stable/.

[2] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://arxiv.org/abs/1908.10084.

[3] Sentence-Transformers. all-minilm-l6-v2 model. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed: 2025-12-09.

[4] Thomas Wolf et al. Transformers: State-of-the-art natural language processing. https://huggingface.co/docs/transformers, 2020. Accessed: 2025-12-09.

[5] Tiago Almeida and Jos Hidalgo. SMS Spam Collection. UCI Machine Learning Repository, 2011. DOI: https://doi.org/10.24432/C5CC84.

[6] Prashant. Matplotlib Tutorial for Beginners. Kaggle Notebook, 2023. URL https://www.kaggle.com/code/prashant111/matplotlib-tutorial-for-beginners. Accessed: 2025-12-08.

[7] Matplotlib Development Team. Matplotlib Gallery: Grouped bar chart with labels. Matplotlib Documentation, 2025. URL https://matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html. Accessed: 2025-12-08.

[8] Scikit-learn Developers. Scikit-learn Documentation: ConfusionMatrixDisplay. Scikit-learn Documentation, 2025. URL https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html. Accessed: 2025-12-08.

[9] Matplotlib Development Team. Matplotlib Pyplot Tutorial. Matplotlib Documentation, 2025. URL https://matplotlib.org/stable/tutorials/pyplot.html. Accessed: 2025-12-08.