
CAS2105 Homework 6: Mini AI Pipeline Project hugging-face

Nishanthi Rengasamy, (2022148038)

1 Introduction

This project provides a gentle introduction to designing simple **AI pipelines**. Rather than training large models or reading extensive research literature, you will:

- Choose a small, concrete problem solvable with an AI pipeline (e.g., text classification, retrieval, simple QA, image classification).
- Choose or collect a small dataset (e.g., from **datasets**).
- Implement a simple *naïve baseline* (e.g., rule-based or heuristic).
- Build an improved pipeline using existing pre-trained models.
- Evaluate both approaches using appropriate metrics.
- Reflect on what worked, what failed, and what you learned.

The emphasis is on the *process* of AI work: problem definition, pipeline design, evaluation, iteration, and writing. Your problem should be small enough to run comfortably on a single GPU (e.g., RTX 3090) or CPU.

Your tasks. Please replace all the sections to complete your report, starting from adding your project title, your name, and student ID above! Feel free to reorganize the sections. Then, submit a **public** github repository link that includes your code (including Jupyter notebook with results) and report **in PDF**, via LearnUs.

You can use tables and figures, and cite references using `\citep` ([1]) or `\citet` (Wei et al. [1]).

2 Task Definition

- **Task description:** Classify a short SMS message into one of two classes: {spam, ham}.
- **Motivation:** Spam filtering is important for email, SMS, and messaging platforms. Even a simple classifier can greatly improve user safety and reduce unwanted content.
- **Input / Output:** Input: a raw SMS message (string).
Output: a label **spam** or **ham**.
- **Success criteria:** The system should outperform the naive keyword baseline and spam F1 score on a held-out test set of 50 messages.

3 Methods

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

3.1 Naïve Baseline

Implement a simple method that does not rely on heavy models. Examples include:

- Keyword-based text classification,
- Simple color/shape heuristics for image tasks,
- String-overlap-based retrieval.

In your report, explain:

- How the baseline works,
- Why it is considered naïve,
- Expected failure cases.

Your Baseline (TODO)

- **Method description:** The baseline predicts a message as spam if it contains any of the spam-related keywords such as "free", "win", "offer", "cash", or "urgent". If none are present, it predicts ham.
- **Why naïve:** It relies purely on surface-level word matching. It cannot handle paraphrases, synonyms, or misspellings (e.g., "fr33" instead of "free").
- **Likely failure modes:**
 - False positives when legitimate messages contain promotional words.
 - False negatives when spam messages avoid common keywords.
 - Inability to understand context or intent.

3.2 AI Pipeline

Design a small pipeline using one or more pre-trained models. Examples include:

- **Text:** embedding encoder + classifier, or a small transformer model,
- **Retrieval:** embedding model + nearest-neighbor search,
- **Vision:** pre-trained classifier (e.g., ViT-tiny).

A typical pipeline contains:

1. Preprocessing,
2. Embedding or representation,
3. Decision/ranking component,
4. Optional post-processing.

Fine-tuning large models is not required; inference-only usage is sufficient.

Your Pipeline (TODO)

- **Models used:** The pipeline uses a TF-IDF vectorizer combined with a Logistic Regression classifier. This represents a classical machine learning approach to text classification.
- **Pipeline stages:**
 1. **Preprocessing:** Lowercasing and simple whitespace cleaning.

2. **Feature Extraction:** TF-IDF converts text messages into numerical vectors based on term frequency and inverse document frequency.
 3. **Classification:** A Logistic Regression model predicts whether a message is spam or ham.
 4. **Post-processing:** Argmax over predicted class probabilities.
- **Design choices and justification:** TF-IDF + Logistic Regression is lightweight, fast to train, and often performs well on small datasets. This model is more context-aware than a keyword baseline because it considers the entire vocabulary distribution.

4 Experiments

4.1 Datasets

You may use a small public dataset (e.g., from `datasets`) or construct your own. In this section, describe:

- **Dataset source:** where it comes from.
- **Size:** number of examples used.
- **Splits:** 150 for training, 50 for testing
- **Preprocessing:** e.g., tokenization, resizing, truncation, normalization.

Your Dataset Description (TODO)

- **Source:** SMS Spam Collection Dataset (UCI/Kaggle)
- **Total examples:** 200 (first 200 rows selected for reproducibility)
- **Train/Test split:** 150 for training, 50 for test examples
- **Preprocessing steps:**
 - Lowercasing all text.
 - Stripping leading/trailing whitespace
 - Converting labels (`ham/spam`) to numerical form,
 - Vectorization through TF-IDF for the machine learning pipeline.

4.2 Metrics

Use at least one quantitative metric appropriate for your task:

- **Classification:** accuracy, precision, recall, F1,
- **Retrieval:** precision@k, recall@k,
- **Simple generation:** exact match, ROUGE-1.

It's worth considering how the metrics you select align with your tasks.

For classification tasks, the following metrics are used:

- Accuracy
- Precision (spam)
- Recall (spam)
- F1 score (spam)

The F1 score is particularly important because spam messages are often less frequent than ham messages.

4.3 Results

Report:

- Metric values for baseline vs. pipeline,
- A results table,
- At least three qualitative examples.

Method	Metric 1	Metric 2 (optional)
Baseline	TODO	TODO
AI Pipeline	TODO	TODO

If you want to visualize results with any other than tables, refer to below links

- [Matplotlib official tutorial: Introduction to pyplot](#)
- [Matplotlib example gallery \(many bar/line/scatter plots with source code\)](#)
- [Kaggle notebook: Matplotlib tutorial for beginners \(interactive code\)](#)

5 My Results

Table 1 reports the quantitative performance of the naive keyword baseline and the improved TF-IDF + Logistic Regression pipeline. The baseline achieves 80% accuracy, but its precision and recall for the spam class are low because it relies heavily on a small set of surface keywords. The TF-IDF model achieves higher accuracy (86%), but completely fails to predict any spam messages, showing the limitations of accuracy in imbalanced datasets.

Method	Accuracy	Precision (spam)	Recall (spam)	F1 (spam)
Baseline (keywords)	0.80	0.33	0.43	0.38
TF-IDF + Logistic Regression	0.86	0.00	0.00	0.00

Table 1: Quantitative comparison of the baseline and improved pipeline.

In addition to quantitative scores, three qualitative examples were analyzed from the test set to better understand model behaviour.

Example 1 (Baseline error, TF-IDF correct). *“The wine is flowing and I’m I have never-ing...”*

True label: ham **Baseline:** spam **TF-IDF:** ham

The baseline misclassifies the message because it contains the substring “win” inside “wine.” TF-IDF correctly classifies it by considering larger context patterns.

Example 2 (Both models miss subtle spam). *“Customer service announcement. You have a New Years delivery waiting for you. Please call to arrange delivery.”*

True label: spam **Baseline:** ham **TF-IDF:** ham

This spam message contains no strong promotional cues, so both models interpret it as a normal notification.

Example 3 (TF-IDF misses obvious spam). “*You are a winner! You have been specially selected to receive £1000 cash...*”

True label: spam **Baseline:** spam **TF-IDF:** ham

The baseline identifies this spam correctly through keywords such as “winner” and “receive.” TF-IDF incorrectly predicts ham because it underfits the minority spam class and rarely outputs a positive prediction.

These examples show the strengths and weaknesses of both approaches. The baseline captures explicit spam signals but overgeneralizes, while TF-IDF captures context better but fails on minority-class detection.

6 Reflection and Limitations

Write approximately 6–10 sentences reflecting on:

- What worked better than expected,
- What failed or was difficult,
- How well your metric captured “quality”,
- What you would try next with more time or compute.

Your Reflection (TODO)

The baseline performed reasonably well, achieving 80% accuracy, but it struggled with keyword oversensitivity. It frequently labeled harmless messages as spam whenever words like “win” or “free” appeared inside longer words such as “wine.” The TF - IDF + Logistic Regression model performed worse on spam detection, predicting all test messages as ham. This happened because the dataset is imbalanced in the first 200 rows, and the model did not learn the minority class. Despite this, TF-IDF still produced the highest accuracy, showing that accuracy alone is not a reliable metric for this task. F1 score for spam was more meaningful because it penalized the model for ignoring more positive cases. If more time were available, I would balance the dataset, use stratified sampling, and run the DistilBERT fine-tuning step for a stronger comparison.

References

- [1] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*, 2022. URL <https://openreview.net/forum?id=gEZrGCozdqR>.
- [2] Tiago A. Almeida and José María Gómez Hidalgo. SMS Spam Collection Dataset. UCI Machine Learning Repository, 2011. URL <https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>.
- [3] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008. URL <https://nlp.stanford.edu/IR-book/>.