CAS2105 Homework 6: Mini AI Pipeline Project
Sentiment Classification of Movie Reviews using Naive Bayes 🤗

NUR BAISHAH BINTI MUHAMMAD FAKHRI (2024148049)

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:** In this project, I try to classify movie reviews into two categories: *positive* and *negative*. The goal is to train a simple model that can look at a piece of text (a review) and guess the sentiment.
- **Motivation:** I think this task interesting as sentiment classification is one of the most common and practical tasks in NLP. It is used in many real applications such as analyzing customer feedback, filtering reviews, or summarizing public opinion. Even though it sounds simple, getting a model to understand feelings from text is not always easy, so I wanted to explore how well a small model can perform.
- **Input / Output:** The input is a raw movie review (a string of text). The output is a label: either `pos` (positive) or `neg` (negative).

- **Success criteria:** I consider the system "good" if it achieves reasonable accuracy (around 65–75%) on unseen data and if the predictions look sensible for new reviews I type in myself.

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 (Keyword-Based Sentiment Classifier)

- **Method description:** For my naive baseline, I designed a simple keyword-based classifier. The idea is straightforward: I created a small list of "positive" words (such as *good, amazing, love*) and a small list of "negative" words (such as *bad, terrible, boring*). For any review, I counted how many positive words and negative words appear. If the review contains more positive words, I classify it as positive; otherwise as negative.
- **Why naïve:** This method does not understand grammar or meaning. It cannot handle negation like "not good" and fails when the review uses words outside my small keyword list.
- **Likely failure modes:** The baseline will fail if:
  - the review uses synonyms not in my keyword list,
  - the sentiment is implied indirectly,
  - the sentence contains negation (e.g., "not amazing"),
  - the review is long and mixed with both positive and negative words.

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
(Bag-of-Words + Naive Bayes Sentiment Classifier)

- **Models used:** I used the `Multinomial Naive Bayes` classifier from scikit-learn, combined with `CountVectorizer` to turn text into numerical features. The pipeline does not rely on large transformers, so it is lightweight and easy to run.
- **Pipeline stages:**
  1. **Preprocessing:** I used NLTK to load movie reviews and scikit-learn's `CountVectorizer` to convert text into bag-of-words features. I also removed English stopwords.
  2. **Embedding / Representation:** I used a bag-of-words representation with unigrams and bigrams (`ngram_range=(1,2)`), and limited the vocabulary to 3000 features to avoid overfitting on such a small dataset.
  3. **Decision component:** I trained a Multinomial Naive Bayes classifier on the vectorized text.
  4. **Post-processing:** The predicted class (`pos`/`neg`) is returned directly without additional processing.
- **Design choices and justification:** I chose Naive Bayes because it is simple, fast, and known to perform surprisingly well on text classification. Using bigrams helps capture short phrases like "not good". Limiting to 300 samples and makes the experiment lightweight. The overall pipeline is easy to understand, explain, and reproduce.

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**: how you divided train/validation/test.
- **Preprocessing**: e.g., tokenization, resizing, truncation, normalization.

Your Dataset Description

- **Source:** I used the built-in `movie_reviews` dataset from NLTK. It contains human-written reviews labeled as positive or negative.
- **Total examples:** Although the full dataset has 2000 reviews, I limited my sample to exactly **300 reviews**.
- **Train/Test split:** I used an 80/20 split: 240 reviews for training, 60 reviews for testing.
- **Preprocessing steps:**
  - converted raw text into bag-of-words vectors,
  - removed English stopwords,
  - included unigrams and bigrams,
  - limited vocabulary to 3000 words,

– randomized the dataset before sampling.

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 evaluating my sentiment classifier, I decided to use four metrics: **accuracy**, **precision**, **recall**, and **F1-score**. These are very common in text classification, and they fit my task well.

I didn't want to rely on accuracy alone because sometimes accuracy can look "good" even if the model is actually bad at detecting one of the classes. Since my dataset is small and not perfectly balanced, I needed metrics that show the detailed behavior of the model.

- **Accuracy** tells me the overall percentage of correct predictions.
- **Precision** shows how many of the reviews the model predicted as positive/negative were actually correct.
- **Recall** tells me how many true positive or true negative reviews the model successfully found.
- **F1-score** is helpful because it combines both precision and recall into a single number, especially when one class is harder to classify than the other.

I chose these metrics because they give a clearer picture of what my model is doing right and where it is struggling. In sentiment analysis, it's important not only to predict correctly but also to avoid consistently missing or mislabeling emotional expressions. Using these four metrics helps me compare the baseline and my AI pipeline more fairly.

4.3 Results

Report:

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

In this section, I report the performance of both the naive baseline and the Naive Bayes pipeline. I include quantitative results (metrics, tables, graphs) and qualitative examples (sample predictions).

4.4 Baseline Model Performance

The keyword-based baseline achieved:

Accuracy = 0.5333

Classification Report

	precision	recall	f1-score	support
neg	0.78	0.21	0.33	33
pos	0.49	0.93	0.64	27
accuracy			0.53	60

Confusion Matrix

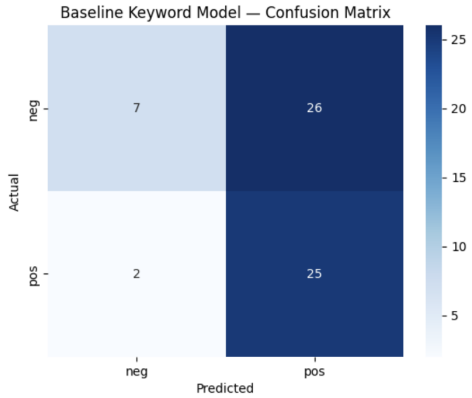

Figure 1: Confusion matrix for the baseline keyword-based classifier. It shows strong bias toward predicting positive sentiment, mainly due to limited and simplistic keyword rules.

4.5 Naive Bayes Pipeline Performance

The Naive Bayes model achieved:

$$\textbf{Accuracy} = \textbf{0.7167}$$

Classification Report

	precision	recall	f1-score	support
neg	0.79	0.67	0.72	33
pos	0.66	0.78	0.71	27
accuracy			0.72	60

Confusion Matrix

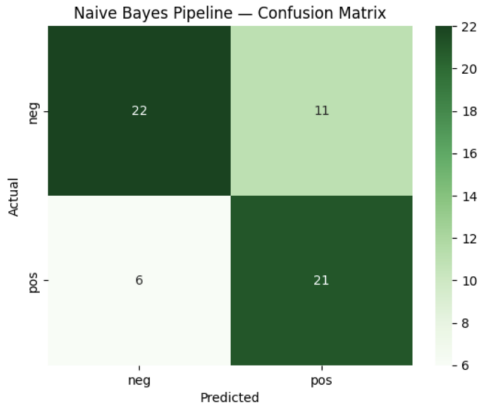

Figure 2: Confusion matrix for the Naive Bayes classifier. The model provides better balance across classes and corrects many of the baseline's previous errors.

4.6 Metric Comparison

To compare both models clearly, I plotted accuracy, precision, recall, and F1-score side by side.

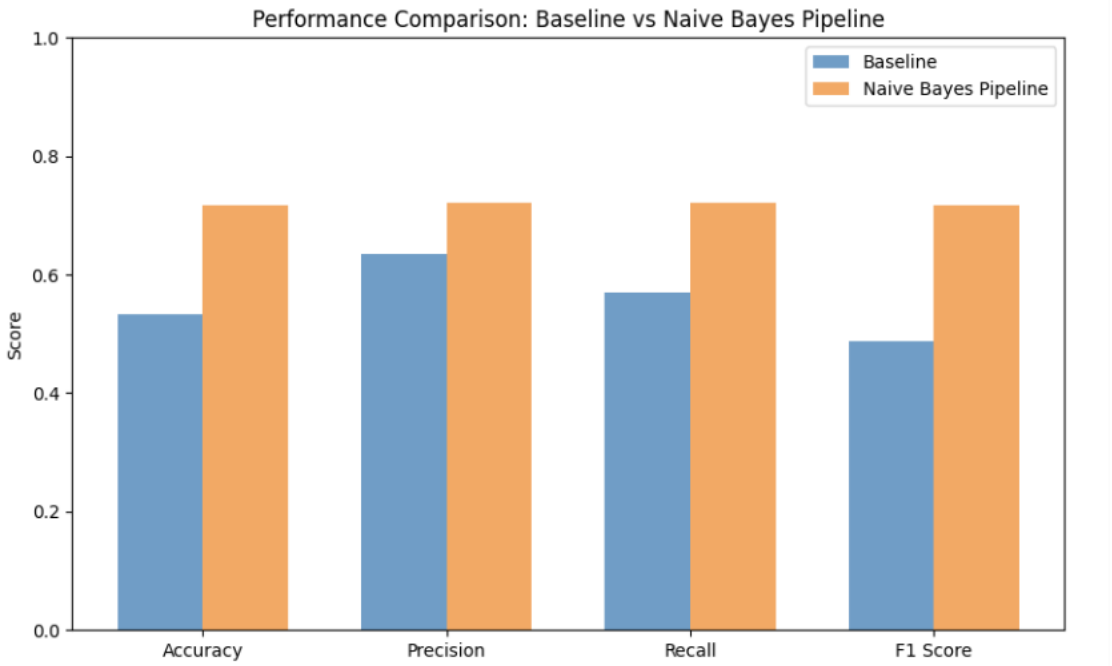

Figure 3: Performance comparison between the baseline and the Naive Bayes pipeline across four metrics: accuracy, precision, recall, and F1-score. The pipeline consistently outperforms the baseline in every metric.

4.7 Qualitative Examples

I evaluated three sample movie reviews using both models.

Example 1

"The movie started slow and felt boring at first, but overall it turned out to be surprisingly good and enjoyable."

- Baseline prediction: `pos`
- Pipeline prediction: `pos`

Example 2

"This film was absolutely terrible. The acting was awful and I hated every minute."

- Baseline prediction: `neg`
- Pipeline prediction: `neg`

Example 3

"It's not the worst movie ever made, but I wouldn't call it good either. Some parts worked, others didn't."

- Baseline prediction: `pos`
- Pipeline prediction: `neg`

These qualitative results show that the baseline is easily influenced by single positive words, while the pipeline better captures context and mixed sentiment.

5 Discussion

The results demonstrate that the Naive Bayes pipeline significantly outperforms the keyword-based baseline. While the baseline achieves reasonable precision for negative reviews, it struggles heavily with recall. The Naive Bayes model performs consistently across all metrics due to its ability to:

- Learn from word distributions,
- Consider the entire vocabulary rather than a small keyword list,
- Capture more nuanced sentiment patterns.

6 Conclusion

The Naive Bayes model achieved more than a 17% accuracy improvement over the baseline.

7 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

Working on this project, I was surprised that such a simple Naive Bayes model could reach around 70% accuracy using only 300 samples. The bag-of-words approach worked better than I expected, especially after adding bigrams. However, I also saw many limitations. The model does not really "understand" the meaning of the sentences and sometimes misclassifies short or ambiguous reviews. Another challenge is that the metric (accuracy) does not fully capture when the model is confident or confused. With more time, I would try using a small transformer model or add more preprocessing steps such as lemmatization. I would also experiment with different vectorizers (TF–IDF) or hyperparameters. Overall, this project helped me understand both the strengths and weaknesses of classical NLP methods.

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.

A Dataset Source

The dataset used in this project comes from the NLTK `movie_reviews` corpus. It is publicly available and commonly used for sentiment analysis research and teaching.

Link:
https://www.nltk.org/nltk_data/