# Project Walkthrough

COSC 4397

Presented by Ziming Zhao and Yang Lu

Summer 2025

## Overview

- Amazon review opinion search engine implementation
- Project description: https://www2.cs.uh.edu/~arjun/courses/nlp_ugrad/hw_proj/res_proj/res_proj.pdf
- 40% of your total credit!
- Submittable:
    - a PDF report,
    - a folder named “codes” with one “readme.txt” for detailed instructions on how to run the code,
    - a folder named “outputs” with your generated outputs.
    - Compress all files into a zip file. Name is lastname_PID.zip, e.g. smith_1234.zip
- Due date: 8/1/2025

# In brief

- Input format (Query): 2-word aspect + opinion
  - Aspect is always two words
  - Opinion is always 1 or 2 words
  - Connotation is required, e.g "The sound was incredibly muffled and lacked any clarity"
- Output – related review ids in a .txt file

- Dataset
  - Source: http://www2.cs.uh.edu/%7Earjun/courses/nlp_ugrad/hw_proj/res_proj/res_proj.7z
  - Total reviews: 210,761
  - Rating distribution: 1 ⋆ (28611), 2 ⋆ (15561), 3 ⋆ (19224), 4 ⋆ (38709), 5 ⋆ (108656)

## Step 1 – Implement the baseline: Boolean Search

- Will substring containment check with Python in statement work?

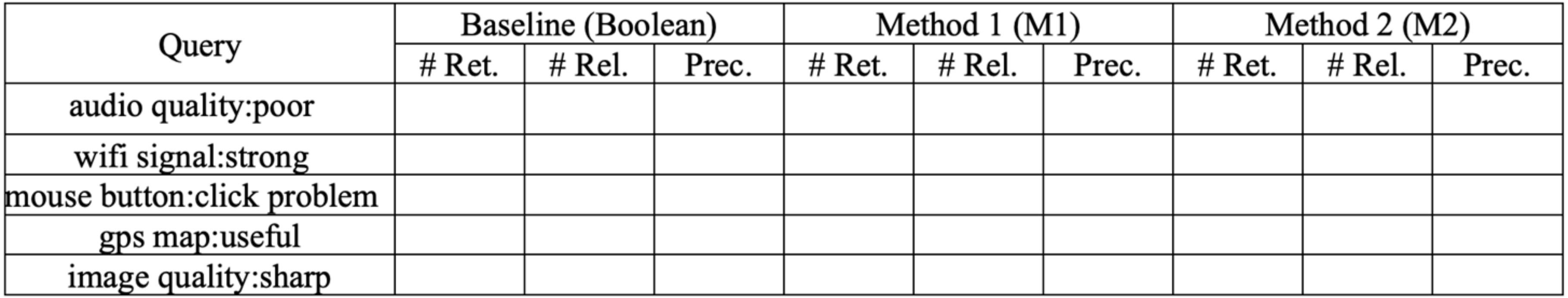
  No, think about subwords: weak in tweak. Think about regex or other SQL based implementation.

- Query results are reviews that contain at least one of the aspect words.
  Query examples that may work:

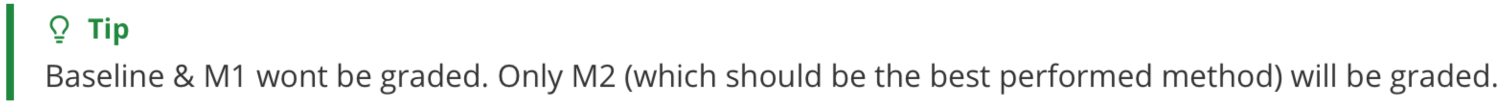
  - “audio quality: poor”, the return reviews should include at least “audio” or “quality” or both of them
  - (audio AND poor) OR (quality AND poor)
  - (audio OR quality) AND poor
  - “mouse button: click problem”, (mouse AND click AND problem) OR (button AND click AND problem)

## Step 2 – Implement two advanced methods

- Compare with the baseline using precision as evaluation metrics
- Turn in the evaluation result of Baseline & Method1 & Method 2 in the report with the following table:



| Query | Baseline (Boolean) | | | Method 1 (M1) | | | Method 2 (M2) | | |
|---|---|---|---|---|---|---|---|---|---|
| | # Ret. | # Rel. | Prec. | # Ret. | # Rel. | Prec. | # Ret. | # Rel. | Prec. |
| audio quality:poor | | | | | | | | | |
| wifi signal:strong | | | | | | | | | |
| mouse button:click problem | | | | | | | | | |
| gps map:useful | | | | | | | | | |
| image quality:sharp | | | | | | | | | |

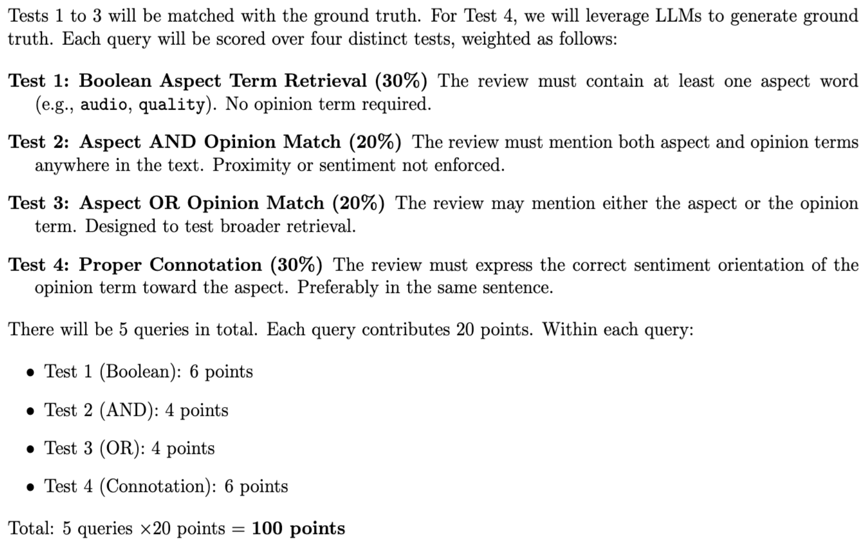


**Tip**

Baseline & M1 wont be graded. Only M2 (which should be the best performed method) will be graded.

# Project Evaluation: 50% Report, 50% Performance Judge

- We will only evaluate the retrieval result from your best method M2 (so try implement and test various SOTAs for better result).
- We will match your result with the ground truth based on 4 tests as below (from grading scheme)



Tests 1 to 3 will be matched with the ground truth. For Test 4, we will leverage LLMs to generate ground truth. Each query will be scored over four distinct tests, weighted as follows:

**Test 1: Boolean Aspect Term Retrieval (30%)** The review must contain at least one aspect word (e.g., `audio`, `quality`). No opinion term required.

**Test 2: Aspect AND Opinion Match (20%)** The review must mention both aspect and opinion terms anywhere in the text. Proximity or sentiment not enforced.

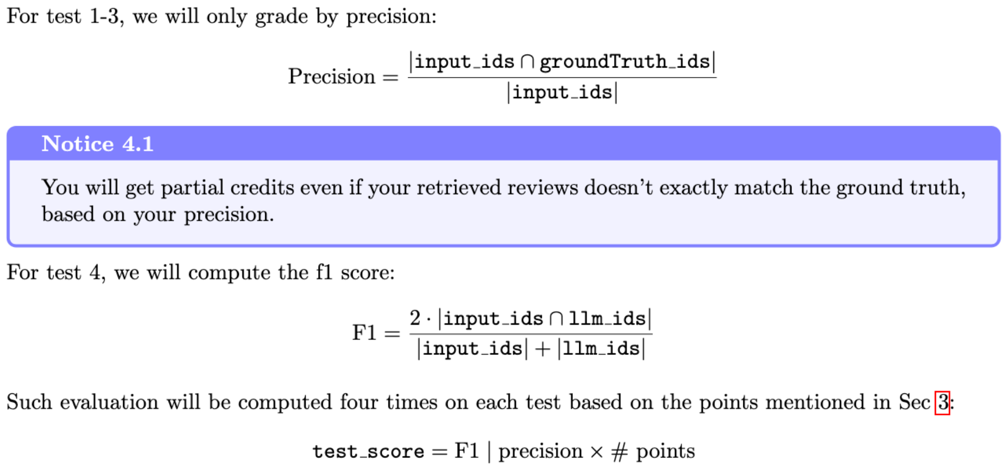
**Test 3: Aspect OR Opinion Match (20%)** The review may mention either the aspect or the opinion term. Designed to test broader retrieval.

**Test 4: Proper Connotation (30%)** The review must express the correct sentiment orientation of the opinion term toward the aspect. Preferably in the same sentence.

There will be 5 queries in total. Each query contributes 20 points. Within each query:

- Test 1 (Boolean): 6 points
- Test 2 (AND): 4 points
- Test 3 (OR): 4 points
- Test 4 (Connotation): 6 points

Total: 5 queries ×20 points = **100 points**



For test 1-3, we will only grade by precision:

$$\text{Precision} = \frac{|\texttt{input\_ids} \cap \texttt{groundTruth\_ids}|}{|\texttt{input\_ids}|}$$

**Notice 4.1**

You will get partial credits even if your retrieved reviews doesn't exactly match the ground truth, based on your precision.

For test 4, we will compute the f1 score:

$$\text{F1} = \frac{2 \cdot |\texttt{input\_ids} \cap \texttt{llm\_ids}|}{|\texttt{input\_ids}| + |\texttt{llm\_ids}|}$$

Such evaluation will be computed four times on each test based on the points mentioned in Sec 3:

$$\texttt{test\_score} = \text{F1} \mid \text{precision} \times \# \text{ points}$$

- The llm_ids are considered as ground truth for test 4. You will get partial credit based on your

# Output Files

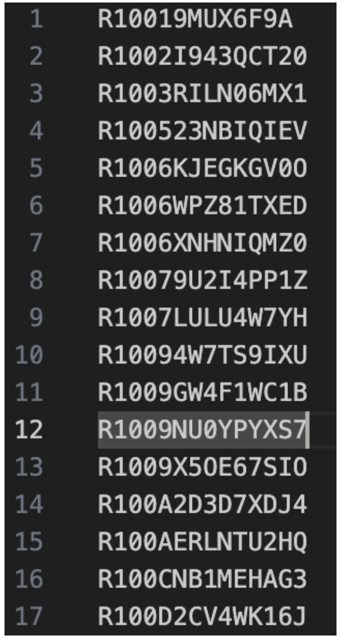
- 📁Baseline model (folder)
  - audio_quality_test1.txt
  - audio_quality_test2.txt
  - audio_quality_test3.txt
  - wifi_signal_test1.txt
  - wifi_signal_test2.txt
  - wifi_signal_test3.txt
  - mouse_button_test1.txt
  - mouse_button_test2.txt
  - mouse_button_test3.txt
  - gps_map_test1.txt
  - gps_map_test2.txt
  - gps_map_test3.txt
  - image_quality_test1.txt
  - image_quality_test2txt
  - image_quality_test3.txt

- 📁Advanced models (folder)
  - audio_quality_test4.txt
  - wifi_signal_test4.txt
  - mouse_button_test4.txt
  - gps_map_test4.txt
  - image_quality_test4.txt

## Step 3 – Format the output

- The output should be in a txt file, with each line representing a review id only.
- The file name should be {aspect}.txt, e.g., audio_quality.txt.
- Use your best model to retrieve the most related reviews!

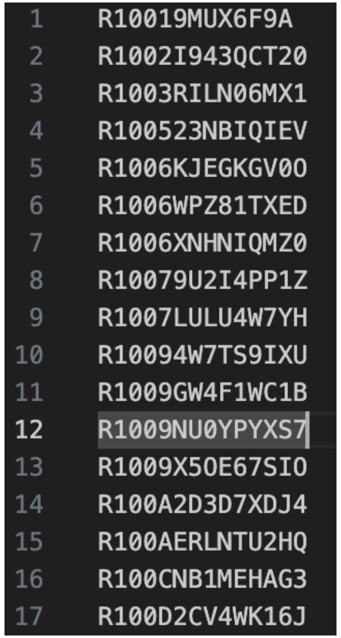
audio_quality_test1.txt:



```
R10019MUX6F9A
R1002I943QCT20
R1003RILN06MX1
R100523NBIQIEV
R1006KJEGKGV00
R1006WPZ81TXED
R1006XNHNIQMZ0
R10079U2I4PP1Z
R1007LULU4W7YH
R10094W7TS9IXU
R1009GW4F1WC1B
R1009NU0YPYXS7
R1009X50E67SIO
R100A2D3D7XDJ4
R100AERLNTU2HQ
R100CNB1MEHAG3
R100D2CV4WK16J
```

✓

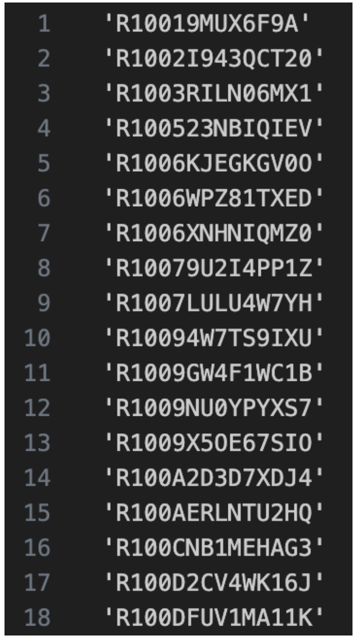
audio.txt:



```
R10019MUX6F9A
R1002I943QCT20
R1003RILN06MX1
R100523NBIQIEV
R1006KJEGKGV00
R1006WPZ81TXED
R1006XNHNIQMZ0
R10079U2I4PP1Z
R1007LULU4W7YH
R10094W7TS9IXU
R1009GW4F1WC1B
R1009NU0YPYXS7
R1009X50E67SIO
R100A2D3D7XDJ4
R100AERLNTU2HQ
R100CNB1MEHAG3
R100D2CV4WK16J
```

✗

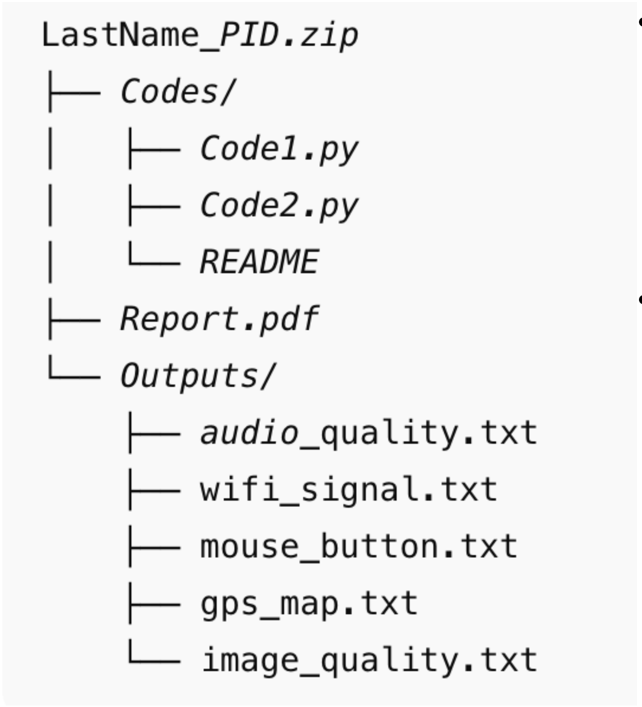
audio_quality.txt:



```
'R10019MUX6F9A'
'R1002I943QCT20'
'R1003RILN06MX1'
'R100523NBIQIEV'
'R1006KJEGKGV00'
'R1006WPZ81TXED'
'R1006XNHNIQMZ0'
'R10079U2I4PP1Z'
'R1007LULU4W7YH'
'R10094W7TS9IXU'
'R1009GW4F1WC1B'
'R1009NU0YPYXS7'
'R1009X50E67SIO'
'R100A2D3D7XDJ4'
'R100AERLNTU2HQ'
'R100CNB1MEHAG3'
'R100D2CV4WK16J'
'R100DFUV1MA11K'
```

✗

## What to submit?



```
LastName_PID.zip
├── Codes/
│    ├── Code1.py
│    ├── Code2.py
│    └── README
├── Report.pdf
└── Outputs/
     ├── audio_quality.txt
     ├── wifi_signal.txt
     ├── mouse_button.txt
     ├── gps_map.txt
     └── image_quality.txt
```

- Name your folder “Codes/” for your code
  - Implementation files
  - README: instructions on how to run your code.
  - If your code for M2 does not run or generate the expected output, you will be penalized.
- Name your folder “Outputs” for your output
  - Only need to submit the retrieval result by your best method M2.
  - You do need to implement both the Baseline and M1 models—it's not possible to evaluate M2 effectively without comparing against them.
- 1 PDF file of your report
  - Name it Report.pdf
- No need to submit any dataset/compiled codes

## Project Report Template

- Overview
- Background
- Method
- Design
- Result
- Discussion
- Conclusion

## Code Demo: Handle Connotation

We will now go through a simple method named sentence-BERT that deals with connotation.

Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." *arXiv preprint arXiv:1908.10084* (2019).