Amazon Review Opinion Search Engine – Final Report

Alexander Yue

2079436

azyue

COSC4397

Arjun Mukherjee

Summer 2025

1. Overview

This project implements an opinion-aware retrieval system for customer reviews using a real-world Amazon dataset. The system supports structured aspect-opinion queries and outputs review IDs most relevant to the query semantics and sentiment. Three models were developed:

- **Baseline:** Boolean matching
- **Method 1:** Boolean + rating-aligned sentiment filtering
- **Method 2:** Sentence-BERT + semantic similarity + rating alignment

Evaluation is based on precision of retrieval across five fixed queries:

- *Audio Quality:Poor*
- *Wifi Signal:Strong*
- *Mouse Button:Click Problem*
- *GPS Map:Useful*
- *Image Quality:Sharp*

2. Background

In classical IR systems, exact keyword matches via Boolean operators limit the system's ability to understand nuanced human language. This is especially problematic for opinion retrieval, where context, polarity, and paraphrasing all influence relevance.

Given a dataset of **210,761** reviews, this project explores layered strategies:

- *Lexical matching - Matching based on exact words or phrases in the text.*
- *Heuristic sentiment scoring - Assigning sentiment (positive/negative/neutral) to text using rule-based or statistical techniques.*
- *Semantic embeddings - Mapping words, sentences, or documents into high-dimensional vector space where meaning is captured.*

The reviews are preprocessed and stored as a Pickle file (reviews_segment.pkl) for efficient retrieval.

3. Query Format

All queries follow this format:

[aspect: opinion]

Where:

- aspect: max 2 tokens (e.g., "audio quality")
- opinion: 1–2 tokens (e.g., "poor", "click problem")

Fixed benchmark queries used:

- audio quality: poor
- wifi signal: strong
- mouse button: click problem
- gps map: useful
- image quality: sharp

4. Methodology

4.1 Baseline – Boolean Matching

Implementation:

The Boolean baseline uses simple keyword-based matching after preprocessing the review text through lowercasing and stopword removal. Three different matching strategies were implemented to evaluate varying levels of strictness in aspect and opinion detection:

Test 1: Aspect-Only Matching

- Matches reviews that contain any word from the aspect term (e.g., for "mouse button", a review containing either "mouse" or "button" would qualify). This approach casts the widest net, focusing solely on detecting the aspect without considering opinion sentiment.

Test 2: Aspect and Opinion Matching

- Matches reviews that contain both the full aspect and opinion terms, requiring that both be present (after stopword removal) in the cleaned review text. This stricter approach reduces false positives but may miss reworded or implied expressions.

Test 3: Aspect OR Opinion Matching

- Matches reviews that contain either any word from the aspect term or all words from the opinion term. This balances between broader recall and partial sentiment relevance.

Each test outputs a list of matching review IDs for a given (aspect, opinion) query and logs the number of matches and processing time.

Strengths:

- Fast and interpretable rule-based system
- Easy to implement and extend for new aspect-opinion pairs

Limitations:

- Cannot detect semantically reworded phrases
- Example: For the opinion “click problem,” a review stating “right mouse button becomes stuck and doesn’t click at all” would be missed.
- Lacks sentiment understanding or disambiguation
- Susceptible to false positives due to subword or partial token matches (e.g., “useful” inside “useless”)

4.2 Method 1 – Boolean Matching + Rating Filter

Enhancement: Add sentiment polarity filtering using review star ratings.

Pipeline Logic:

1. Execute Boolean AND matching as in baseline.
2. Determine if the query opinion is positive or negative.
3. Filter matched reviews:
    - Positive opinion -> keep if rating > 3
    - Negative opinion -> keep if rating ≤ 3

Advantages:

- Aligns opinion sentiment with actual review rating
- Reduces sentiment-inverted matches

Limitations:

- Relies on predefined opinion lexicon
- Still limited by lexical overlap

4.3 Method 2 – Sentence-BERT + Semantic Filtering + Rating

Architecture:

- Uses sentence-transformers with the all-MiniLM-L6-v2 model
- Embeds both query and review texts using Sentence-BERT
- Computes cosine similarity; filters with a threshold of 0.6
- Applies rating filter just like in Method 1

Technical Details:

- Embedding dimension: 384
  - Each sentence or text input is converted into a 384-dimensional vector.
- Cosine similarity via `util.cos_sim`
  - a function that measures how **similar** two vectors are by the **angle** between them.
- Batch inference supported with GPU acceleration
  - Grouping multiple inputs together to process them simultaneously, improving efficiency and speed. Used for speed optimization with CUDA GPU's

Advantages:

- Captures paraphrasing and semantic similarity
- Outperforms prior methods in nuanced queries

Limitations:

- High compute/memory cost
  - ~830s for this method compared to ~85s for method1
- Threshold tuning required
  - Thresholds tested:
    - .3, .45, .6 (best results), .7

5. Evaluation Strategy

Each method was evaluated on its ability to retrieve relevant review IDs for the five predefined aspect-opinion queries. The primary metric used for comparison was precision, defined as:

Precision = (Relevant Matches) / (Total Matches Returned)

Relevant reviews for baseline models (results>1000) were identified by copying 50 random queried reviews, while the rest were from the entire sample size. An LLM was then asked which ones were relevant to each request. Example prompt:

- "Out of the following reviews, which one is relevant to Audio Quality:Poor"

Because the accuracy of LLM's are not 100%, there may be missed or false positive reviews, particularly for larger review sets. In fact, during review of the results, some false positive/negative reviews were identified and corrected. While there is a likely a high number of false positive/negatives that were missed, since relevant reviews will be graded by an LLM, and it would be near impossible to assess all results by hand, it was decided this was a good choice for finding relevant reviews

6. Results Summary (Baseline)

Query	Baseline ASPECT 48.6 seconds			Boolean AND 73.4 seconds			Boolean OR 60.88 seconds		
	#Ret.	*#Rel/50*	*Prec.*	*#Ret.*	*#Rel/50*	*Prec.*	*#Ret.*	*#Rel/50*	*Prec.*
Audio quality: poor	23164	6	.12	1882	24	.47	27680	10	.2
Wifi signal: strong	3373	5	.1	304	3	.06	8635	4	.08
Mouse button: click problem	9696	6	.12	502	3	.06	10325	3	.06

Gps map: useful	3891	4	.08	340	2	.004	8948	5	.1
Image quality: sharp	23436	5	.1	1103	4	.08	25394	7	.14

Query	Method1 (Boolean + Rating Filter) Time Taken: 86.94 seconds			Method2 (Semantic Similarity with BERT) Time Taken: 830.57		
	#Ret.	*#Rel*	*Prec.*	*#Ret.*	*#Rel*	*Prec.*
Audio quality:poor	23	20	.8696	7	6	.8571
Wifi signal:strong	4	4	1	2	2	1
Mouse button:click problem	0	0	n/a	9	7	.7778
Gps map:useful	5	4	.80	154	92	.5794
Image quality:sharp	82	2	.0244	11	7	.6363

Tests were run on a machine with the following OS and Hardware:

- OS: Windows 10 Pro
- AMD Ryzen 5 1600, 6 Cores, 3700Mhz clock
- 24GB Vram
- NVDIA RTX 2080 6GB Vram

Discussion:

- **Baseline** often returned many irrelevant results due to shallow matching.
- **Method 1** showed improved filtering for sentiment alignment, especially for queries like “poor audio quality.”
- **Method 2** returned reviews that were exactly relevant, rather than keyword searching that boolean applies.
- **Semantic Models Matter:** Queries like "gps map:useful" retrieved diverse phrasing like “helpful for navigation” or “perfect trip companion” under Method 2 but were missed by Boolean methods.
- **False Positives in Baseline:** For "wifi signal:strong", the baseline incorrectly matched “weak signal” due to only checking for "signal".
- **Rating Filtering Reduces Noise:** Several irrelevant matches were removed in Method 1 and Method 2 through filtering based on rating number.

8. Conclusion

This project demonstrates how progressively sophisticated retrieval techniques improve results and relevance of opinion queries. While the baseline Boolean technique is viable as a simple, quick, easy to implement starting point, semantic models such as Sentence-BERT meaningfully boosts precision, as well as identifying reviews potentially missed by baseline boolean models. However, it comes at a massive computational, and time cost, as Baseline Boolean techniques each took between 48 and 61 seconds, while method 1 took ~86 seconds on CPU, while Semantic Models took ~850+ seconds on a CUDA core system. Ultimately, this project highlights the power of transformer-based sentence encoders, proving their usefulness in real-world IR tasks, as well as showing the benefits and value of hybrid pipelines that combine semantic similarity, with sentiment alignment heuristics.